Universidad Nacional Autónoma de México

Facultad de Ciencias

Programación, procesamiento y visualización de grandes volúmenes de datos dentro de un sistema manejador de base de datos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:
R O B E R T O Á V I L A S Á N C H E Z

DIRECTOR DE TESIS:
DR. JORGE LUIS ORTEGA ARJONA

2015

Ciudad Universitaria, D. F.

A mis padres Nora y Roberto
A mi hermana Pamela
A mi novia Yazmín

Agradecimientos

A mis padres ***Nora*** y ***Roberto***, gracias por todo el apoyo, esfuerzo y sacrificio que han realizado a lo largo de sus vidas para que nunca me faltará nada en ningún momento, no solo de mi carrera, sino de mi vida. Gracias a ustedes por inculcarme el respeto, el amor y la unión familiar, esas enseñanzas y valores que solo ustedes mis mejores maestros me podían enseñar.

Gracias ***Mami*** por dedicar tu vida a cuidarnos y educarnos día a día, gracias por escucharme y apoyarme en los momentos más dificiles de mi vida, sin tu apoyo jamás hubiera logrado terminar está tesis.

Papá gracias por proveerme de todo lo necesario en mi vida, tu sentaste en mi las bases de la respondabilidad, la honradez y el trabajo duro, para realizar mis sueños y deseos de superación, tus virtudes infinitas y tu gran corazón te hacen ser una persona muy especial.

Pame que te puedo decir, tu eres la mejor hermana que me pudo haber tocado, nunca dejes de sonreir a la vida porque la vida siempre te sonríe a ti. Gracias por brindarme tu apoyo y cariño incondicionalmente y nunca olvides que siempre nos tendremos el uno al otro.

Mi hermosa y pequeña ***Yaz***, estos doce años han sido más que perfectos y maravillosos a tu lado, gracias por tantas aventuras e historia vividas, tu apoyo, amor y comprensión son las principales razones por la que logré terminar esta tesis.

Agradezco muy especialmente a mi asesor, ***Jorge Luis Ortega Arjona***, por su apoyo, su tiempo y confianza para realizar este trabajo, sin sus grandes consejos y sabiduría este trabajo no se hubiera logrado.

A la Maestra ***Guadalupe Ibargüengoitia*** y al Dr. ***Gustavo de la Cruz*** por su ayuda en la revisión y corrección del trabajo, sus consejos fueron una gran ayuda.

A mis grandes amigos de la facultad por su amistad incondicional, ***Omar, Roger, Rodro, Cavazos, Luis Enrique***, gracias por las risas y aventuras a su lado sin ustedes la universidad no habria sido igual, ***Rebe*** la reina de las bases de datos y la normalización, gracias por tu infinita ayuda.

Por último a la ***Universidad Nacional Autónoma de México*** por otorgame una educación de primer nivel sin pedir nunca nada a cambio, por eso y mucho más ***GRACIAS***.

"Soló los libros sacarán de la barbarie a este país."

Resumen

El presente trabajo muestra el desarrollo y programación de estimadores y métodos estadísticos en un Sistema Manejador de Base de Datos (SMBD). El objetivo principal es ejecutar los estimadores y métodos estadísticos y puedan procesar una gran cantidad de datos almacenados en una base de datos dentro del SMBD, y al recuperar la información se puedan visualizar resultados electorales de una manera eficiente.

La ejecución de los estimadores y métodos estadísticos requieren de un tiempo determinado de procesamiento, debido a que se tiene que operar cada estimador y método sobre una gran cantidad de datos. Para optimizar las operaciones y poder visualizarlos (a través de un programa objetivo), los estimadores y métodos estadísticos se programan como funciones dentro de un sistema manejador de base de datos.

El problema a tratar es poder lograr que la programación de los estimadores y métodos estadísticos, dentro de un sistema de base de datos, permita procesar grandes cantidades de datos y a su vez otorgar información en un tiempo delimitado.

Los métodos estadísticos que se desarrollan son los siguientes:

- Probabilidad proporcional al tamaño.
- Muestreo aleatorio simple.

Los estimadores estadísticos que se desarrollan son los siguientes:

- Razón.
- Proporción.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto

Los sistemas manejadores de bases de datos son una colección de datos interrelacionados, así como un conjunto de programas diseñados para acceder a esos datos. Dicha colección de datos es definida como base de datos.
La función principal de los sistemas manejadores de base de datos es proveer una manera conveniente y eficiente de almacenar y recuperar información de una base de datos.[11]
Un sistema de base de datos está diseñado para manejar grandes cantidades de información. La gestión de datos implica la definición de estructuras de almacenamiento, además de proporcionar mecanismos para la manipulación de información.
Algunos beneficios que se obtienen al usar un sistema de base de datos son:

- Abstracción de datos.
- Instancias y esquemas.
- Modelos de datos.
- Lenguaje de definición de datos.
- Lenguaje de manipulación de datos.

Al utilizar un sistema manejador de base de datos para este trabajo, se espera tener una solución más eficiente para el problema de obtener resultados electorales mediante métodos y estimadores estadísticos .

Para poder visualizar los resultados electorales procesados por los métodos y estimadores estadísticos, se realiza una programa objetivo basado en una

arquitectura web, utilizando el lenguaje Java en su versión J2EE, que servirá para interactuar con el sistema de base de datos.

1.2. Problema

Obtener información de una elección para calcular resultados electorales en tiempo real, por medio de estimadores y métodos estadísticos.
El cálculo de estimadores y métodos estadísticos implica una serie de operaciones algebraicas para deducir inferencias acerca de una población o fenómeno en estudio. En este trabajo los estimadores y métodos estadísticos calculan las preferencias del votante el día de la elección. Dado este problema, se busca una forma de poder obtener información precisa, rápida y confiable. Es por eso que surge la necesidad de crear un sistema para poder estimar resultados electorales donde se minimicen los errores, así como los sesgos de información, alcanzando un alto rendimiento en el tiempo de respuesta.

1.3. Objetivo

Analizar y desarrollar las diversas etapas que caracterizan la creación de un sistema web para realizar el cálculo de resultados electorales por medio de los estimadores estadísticos razón y proporción, así como los métodos de muestreo aleatorio simple y muestreo por probabilidad al tamaño. El propósito de esta tesis no es mejorar los algoritmos estadísticos usados, sino adaptarlos para que se pueda llevar a cabo el cálculo de resultados electorales dadas las normas otorgadas por el Instituto Federal Electoral.
El enfoque de este trabajo es implementar los algoritmos en lenguaje SQL y ejecutarlos como una función, en el sistema manejador de base de datos, así como realizar la programación del sistema web con el patrón de diseño MVC para la visualización de los resultados electorales.

1.4. Hipótesis

Procesar los estimadores de razón y proporción, así como los métodos de muestreo aleatorio simple y muestreo por probabilidad al tamaño simultáneamente sobre una gran cantidad de datos, para poder visualizar resultados electorales confiables en tiempo real, mediante un sistema web.

1.5. Aproximación

Para los propósitos de esta tesis se elige el manejador de base de datos PostgreSQL por ser un manejador de código abierto[1] multiplataforma, diseñado para trabajar en entornos que usan grandes volúmenes de datos, así como sus bajos requerimientos de administración y mantenimiento respecto a bases de datos comerciales. Este manejador tiene la ventaja de procesar tareas simultáneamente y ajustarse al número de CPU's[2] disponibles, soportando grandes cantidades de peticiones a la vez.

La implementación del Sistema Web para la visualización de resultados electorales se implementa con el servidor Glassfish, dicho servidor es capaz de soportar la especificación de desarrollo JavaEE usada para este trabajo.
La especificación de JavaEE permite utilizar una arquitectura de N capas distribuidas y apoyadas en componentes de software modulares, que se ejecutan dentro de un servidor de aplicaciones. Para este trabajo se utilizan las especificaciones de JDBC, Enterprise Java Beans y Java Server Faces.

Se usa Java como lenguaje de desarrollo usando su especificación JavaEE, para realizar un Sistema Web que obtenga resultados en tiempo real, haciendo las peticiones de información al manejador de base de datos, a través de un conector JDBC.[3] El framework JSF[4] nos ayuda a la presentación de resultados obtenidos de la base de datos.

[1] Se refiere al software que se puede usar, modificar y redistribuir gratuitamente.

[2] Se refiere a Unidad Central de Procesamiento del inglés Central Processing Unit, CPU.

[3] JDBC se refiere a Java Database Connectivity, una especificación de Java para realizar operaciones sobre una base de datos.

[4] JSF se refiere a Java Server Faces, es una tecnología que establece un estándar para crear interfaces de usuarios del lado del servidor.

1.6. Estructura de Tesis

El capítulo 2 contiene una breve descripción de los temas y tecnologías que se usan para desarrollar el proyecto.

El capítulo 3 habla sobre el trabajo y resultado relacionado con el tema de esta tesis, comparando los diferentes métodos de resolución.

El capítulo 4 desarrolla los distintos estimadores y métodos estadísticos en SQL, para el cálculo de resultados electorales.

El capítulo 5 explica a detalle la implementación del sistema, y se evalúan los resultados obtenidos.

El capítulo 6 presenta las conclusiones del trabajo realizado, así como los trabajos futuros que puedan desprenderse del mismo.

Capítulo 2

Antecedentes

En este capítulo se da una breve introducción al tema de Base de Datos y los Sistemas Manejadores de Base de Datos. Se enfoca en los puntos más importantes para la realización de este trabajo, en los cuáles se proveen conceptos y definiciones acerca de una base de datos, modelos de datos y su funcionamiento en un Sistema Manejador de Base de Datos, así como sus componentes principales.
Además, se presenta una breve descripción matemática de los métodos estadísticos de muestreo aleatorio simple, muestreo por probabilidad al tamaño y los estimadores estadísticos de razón y proporción implementados para este trabajo.

2.1. Base de Datos

Un dato es la representación simbólica, un atributo o característica de una entidad. Un dato por sí mismo no tiene ningún valor semántico, pero en conjunto y al ser procesados pueden proporcionar información. Una base de datos es un conjunto de datos interrelacionados y organizados en una computadora. Detrás de la estructura de una base de datos se encuentra el modelo de datos, el cuál es una colección de herramientas conceptuales para describir los datos, como la relación que existe entre los datos, la semántica de los datos así como las restricciones de consistencia. Un modelo de datos proporciona una forma de describir el diseño de una base de datos en sus niveles físicos, lógicos y de visión [11].

Los modelos de datos pueden ser clasificados dentro de cuatro diferentes categorías:

- **Modelo relacional**. El modelo relacional contempla a una relación

como un conjunto de tuplas, que es una secuencia finita y ordenada de datos. Una relación se representa como una tabla.

- **Modelo Entidad-Relación**. El modelo de entidad-relación (ER) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades y relaciones entre estos objetos. Una entidad es una cosa u objeto en el mundo real que es distinguible de otros objetos aún siendo del mismo tipo. Una entidad se representa por características o atributos.

- **Modelo de datos basado en objetos**. El paradigma Orientado a Objetos se ha convertido en la metodología predominante en el desarrollo de software, esto llevó a desarrollar un modelo de datos orientado a objetos que es visto como la extensión del modelo entidad-relación con nociones de encapsulación, métodos ó funciones.

- **Modelo de datos semiestructurado**. El modelo semiestructurado permite la especificación de datos donde los elementos individuales de datos pueden tener diferentes conjuntos de atributos[11], es decir, los datos no tienen una estructura rígida y mucho menos predefinida[5].

El modelo de datos utilizado para este trabajo es el modelo relacional, se utiliza PostgreSQL como SMBD por ser un sistema de gestión de base de datos relacional, la decisión de usar PostgreSQL se debe a que es un SMBD de código abierto, además dicho manejador utiliza un modelo cliente/servidor.

2.1.1. Modelo de Datos

El modelo de datos es un conjunto de conceptos que pueden usarse para describir la estructura de una base de datos [5], ya que provee una visión del problema a resolver. El diseño de una base de datos se centra en poder tener un esquema conceptual para el correcto almacenamiento y manejo de datos. A continuación se explica a detalle el modelo relacional, que es el que se usa para realizar este trabajo.

Modelo Relacional

El modelo relacional fue propuesto por Edgar Frank Codd en 1970. Dicho modelo se basa en la lógica de predicados y la teoría de conjuntos. La idea fundamental de este modelo es el uso de "relaciones". El termino de relación es usado para referirse a una tabla, mientras el termino de tupla [1] se refiere a

[1] En terminología matemática tupla es una secuencia o lista de valores.

un renglón, cada dato de la tupla forma parte de una columna, dicha columna puede ser nombrada "atributo" [11].
En la figura 2.1 se ejemplifica la representación de una relación.

ID	NOMBRE	ÁREA	SALARIO
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figura 2.1: Ejemplo de una relación en el modelo relacional

La principal ventaja del modelo relacional es que reduce la complejidad de estructurar una base de datos, ya que el modelo específica un marco conceptual en el que provee qué atributos se necesitan para los datos.

Normalización

El objetivo de la normalización en una base de datos relacional es generar un esquema que permita almacenar información evitando redundancias innecesarias, pero también que permita recuperar información fácilmente [11]. La Normalización o Formas Normales son aplicadas a las tablas de una base de datos, para que una tabla se encuentre en alguna forma normal se debe satisfacer un cierto número de criterios, existen seis tipos de formas normales que son [10]:

1. Primera Forma Normal (1FN). Para que una tabla se encuentre en 1FN se debe satisfacer lo siguiente:

   a) Todos sus atributos deben ser atómicos.

   b) La tabla debe de contener una llave[2] única.

   c) La llave única no debe contener atributos nulos.

[2]Una llave es aquella columna (o conjunto de columnas) que identifica únicamente a una tupla.

d) La tabla no puede tener múltiples valores en cada columna.

2. Segunda Forma Normal (2FN). Una tabla se encuentra en 2FN si se encuentra en 1FN y los atributos que no forman parte de ninguna llave y dependen de forma completa de la llave principal.

3. Tercera Forma Normal (3FN). Una tabla se encuentra en 3FN si se encuentra en 2FN y si no existe ninguna dependencia funcional transitiva entre los atributos que no son llave.

4. Forma Normal de Boyce-Codd (FNBC). La FNBC requiere que no existan dependencias funcionales no triviales de los atributos que no sean un conjunto de una llave candidata.

5. Cuarta Forma Normal (4FN). Una tabla se encuentra en 4FN si y sólo sí está en 3FN o en FNBC (cualquiera de ambas) y no posee dependencias multivaluadas no triviales.

6. Quinta forma normal (5FN). Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo sí, cada relación de dependencia se encuentra definida por llaves candidatas.

Para este trabajo sólo se requiere de usar las primeras tres formas normales, con dichas reglas se cubren las necesidades de la base de datos para el cálculo de resultados electorales.

2.2. Sistema Manejador de Base de Datos

Un sistema manejador de base de datos (SMBD) es un conjunto de programas especializados que permiten a los usuarios almacenar, modificar y acceder a datos contenidos en una base de datos[11]. El SMBD actúa como interfaz entre los usuarios del sistema y el modelo de datos, en la figura 2.2 se muestra un diagrama que ejemplifica el funcionamiento de un SMBD.

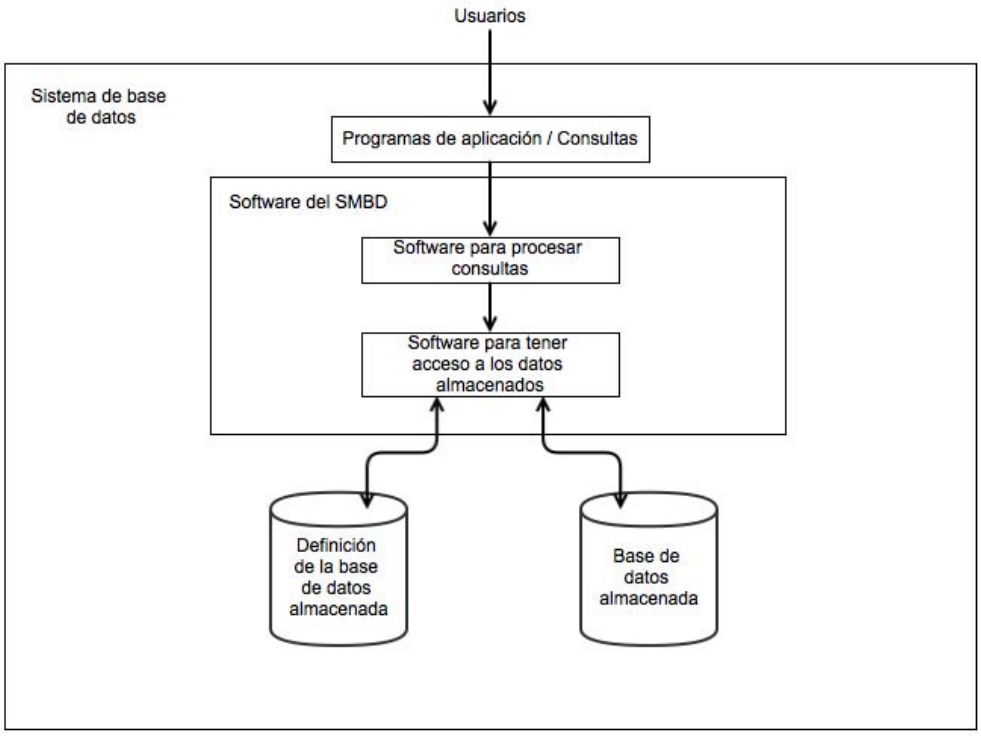

Figura 2.2: Diagrama conceptual de un SMBD [5]

La arquitectura de un SMBD consta de tres capas como se muestra en la figura 2.3, las capas que intervienen en el funcionamiento de un SMBD son [11]:

- **Capa Física**. En la capa física o de almacenamiento se describe cómo se almacenan los datos a nivel de disco, en ella se describen las complejas estructuras de datos en un bajo nivel.
- **Capa Lógica**. En la capa lógica se definen qué datos están guardados en la base de datos y que relaciones existen entre esos datos. Aunque

la capa lógica describa la base de datos en estructuras relativamente simples, eso puede implicar que a nivel físico las estructuras sean muy complejas, el usuario del nivel lógico no necesita estar al tanto de la complejidad que existe en la capa física.

- **Capa de Vista**. En la capa de vista el usuario del sistema accede a los datos a través de programas o aplicaciones, para simplificar la interacción con el sistema.

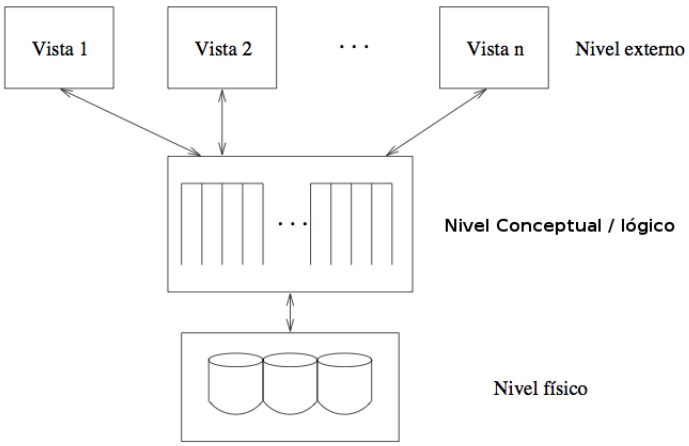

Figura 2.3: Diagrama conceptual de la arquitectura en un SMBD [5]

Los datos son el material esencial para derivar información. Dado lo anterior, el SMBD nos provee de tres lenguajes que son [5]:

- Lenguaje de definición de datos LDD.
- Lenguaje para la manipulación de datos LMD.
- Lenguaje para la consulta de datos SQL.

2.2.1. Lenguaje de Consultas

Álgebra Relacional

El álgebra relacional es un lenguaje de consultas procedimental. Esté consiste en un conjunto de operaciones que toman una o dos relaciones o tablas como entrada y producen una nueva relación ó tabla como resultado. Las operaciones fundamentales en el álgebra relacional son [11]:

- **Selección**. La operación selección permite seleccionar tuplas de una relación para satisfacer un predicado dado.

- **Proyección**. La operación proyección es una operación unaria que devuelve atributos de una relación, algunos atributos pueden omitirse o quedar fuera de la operación. Las tuplas duplicadas se eliminan.
- **Unión**. La operación unión es una operación binaria que permite unir atributos de dos relaciones distintas, es decir regresa el conjunto de tuplas que están en la relación r, o en la relación s o en ambas. Para que la operación sea válida r y s requieren de cumplir dos condiciones:
  1. La relación r y s deben ser del mismo rango, es decir, deben de contener el mismo número de atributos.
  2. Los tipos de los atributos de r deben ser los mismo que los atributos de s.
- **Diferencia**. La operación diferencia permite encontrar las tuplas que están en una relación pero no en otra.
- **Producto Cartesiano**. La operación de producto cartesiano permite combinar la información de dos relaciones distintas.
- **Renombrar**. A diferencia de las relaciones en la base de datos, los resultados que generan las expresiones del álgebra relacional no tienen un nombre, es por eso que se puede usar la operación renombrar para poder referirse a ellos.

Con dichas operaciones el álgebra relacional describe cómo se pueden manipular los datos. Debido a las propiedades algebraicas que proporciona el álgebra relacional se pueden definir operaciones más complejas mediante la composición de operaciones.

Cálculo Relacional de Tuplas

Cuando se escribe una expresión del álgebra relacional, se realiza una serie de procedimientos que genera la respuesta a una pregunta. El cálculo relacional de tuplas, por el contrario, es un lenguaje de consulta no procedimental. En él se describe la información deseada sin dar un procedimiento específico para la obtención de información.
Una consulta en el cálculo relacional de tuplas es expresada como:

$$\{t|P(t)\} \tag{2.1}$$

Es decir, el conjunto de todas las tuplas t tal que el predicado P es cierto para t. Con la notación anterior, se puede expresar a $t[A]$ para indicar el valor de la tupla t para el atributo A, y $t \in r$ para denotar que la tupla t pertenece

a la relación r. Una fórmula de cálculo relacional de tuplas se construye a partir de átomos.
Se contruyen fórmulas de átomos usando las siguientes reglas:

- Un átomo es una fórmula.
- Si P_1 es una fórmula, entonces también lo son $\neg P_1$ y (P_1).
- Si P_1 y P_2 son fórmulas, también lo son $P1 \vee P2$, $P1 \wedge P2$ y $P1 \Rightarrow P2$.
- Si P_1(s) es una fórmula que contiene tuplas libres, y r es una relación, entonces $\exists s \in r(P_1(s))$ y $\forall s \in r(P_1(s))$ también son fórmulas.

Con estás reglas el cálculo relacional de tuplas pueden generar una infinidad de relaciones, es así que, para cada expresión de álgebra relacional utilizando sólo las operaciones básicas hay una expresión equivalente en el cálculo relacional de tuplas, y por cada expresión del cálculo relacional de tuplas, hay una expresión del álgebra relacional equivalente [11].

2.2.2. SQL

Los Sistemas Manejadores de Base de Datos proveen un lenguaje para poder realizar consultas de información denominado SQL.
SQL[3] es un lenguaje estándar para trabajar con bases de datos relacionales que está basado principalmente en una combinación de álgebra relacional y cálculo relacional [5]. Dicho lenguaje permite la definición, acceso y control de datos en una base de datos relacional [5].

El lenguaje de SQL está conformado por las siguientes partes [11]:

- *Lenguaje de definición de datos (LDD).* El SQL LDD proporciona instrucciones para definir los esquemas de las relaciones, borrar relaciones, y modificar relaciones.
- *Lenguaje de manipulación de datos (LMD).* El SQL LMD ofrece la posibilidad de consultar información de la base de datos y poder insertar, borrar y modificar tuplas existentes en la base de datos.
- *Integridad.* El SQL LDD incluye instrucciones para especificar restricciones de integridad a los datos almacenados en la base de datos. Las actualizaciones que violen dichas restricciones de integridad no permiten realizarse.

[3]Su significado es Structured Query Language o Lenguaje de Consultas pero por convención lo llamaremos SQL.

- *Definición de vistas.* El SQL LDD incluye instrucciones para definir vistas.
- *Control de Transacciones.* Permite especificar el inicio y el termino de una transacción, así como bloques explícitos de datos para controlar la concurrencia.
- *Autorización.* El SQL LDD incluye instrucciones para especificar accesos a relaciones y vistas.

Lenguaje de Definición de Datos (LDD)

El conjunto de relaciones en una base de datos deben ser especificadas por el lenguaje de definición de datos proporcionados por SQL, dicho lenguaje permite especificar información sobre cada relación como lo es:

- El esquema para cada relación.
- El tipo de valores asociado a cada atributo.
- Las reglas de integridad.
- Conjuntos de índices que se mantiene en cada relación.
- La seguridad y autorización para cada relación.
- La estructura de almacenamiento físico para cada relación en el disco duro.

El LDD incluye instrucciones para crear tablas, índices, vistas y definición de los accesos a las relaciones de la base de datos. Las instrucciones para la definición de estructuras se presentan en el cuadro 2.1 y en el cuadro 2.2 las restricciones que se pueden aplicar a las relaciones o atributos. [4].

INSTRUCCIÓN	DESCRIPCIÓN
CREATE SCHEMA	Crea un nuevo esquema para una base de datos.
CREATE TABLE	Crea una nueva tabla en el esquema del usuario de la base de datos.
CREATE TABLE AS	Crea una nueva tabla basada en una consulta hecha por el usuario.
CREATE VIEW	Crea la vista de una consulta pero dicha vista no es materializada, siempre se encuentra en memoria.
CREATE INDEX	Crea un índice a una tabla.
DROP TABLE	Borra permanentemente una tabla (y los datos contenidos en la misma).
DROP INDEX	Borra permanentemente un índice.
DROP VIEW	Borra permanentemente una vista.

Cuadro 2.1: Instrucciones LDD

INSTRUCCIÓN	DESCRIPCIÓN
NOT NULL	Asegura que la columna no puede contener valores nulos.
UNIQUE	Asegura que la columna no puede contener valores duplicados.
PRIMARY KEY	Define una llave primaria para una tabla.
FOREIGN KEY	Define una llave foránea para una tabla.
DEFAULT	Define un valor por omisión a una columna.
CHECK	Se le permite especificar que el valor de un determinado atributo debe satisfacer una expresión booleana
ALTER TABLE	Modifica la definición de una tabla (agrega, modifica o borra atributos).

Cuadro 2.2: Instrucciones de restricción LDD

Lenguaje de Manipulación de Datos (LMD)

La estructura básica de una consulta SQL consiste en tres cláusulas: **SELECT, FROM y WHERE** . La consulta toma como entrada las relaciones que se encuentran después de la cláusula FROM, opera sobre los atributos

como se específica en la cláusula WHERE y SELECT, produciendo una relación como resultado [11].
Un ejemplo de una consulta típica en SQL tiene la siguiente forma:

$$\textbf{SELECT}\ A_1, A_2, ..., A_n$$
$$\textbf{FROM}\ r_1, r_2, ..., r_m$$
$$\textbf{WHERE}\ P$$

Cada A_i representa un atributo y cada r_i una relación, P es un operación.
SQL forma el producto cartesiano de las relaciones incluidas en la cláusula FROM, lleva a cabo las operaciones de la cláusula WHERE y proyecta el resultado sobre los atributos de la cláusula SELECT [11].
El LMD permite consultar o manipular datos, incluye instrucciones para insertar, actualizar, borrar y recuperar datos contenidos en una relación en la base de datos.
Las instrucciones para la manipulación de datos se presentan en el Cuadro 2.3 [4].

COMANDO	DESCRIPCIÓN
INSERT	Inserta renglones dentro de una tabla.
SELECT	Selecciona atributos de los renglones de una o más tablas o vistas.
WHERE	Restringe la selección de renglones basados en una expresión condicional.
GROUP BY	Agrupa la selección de renglones basada en uno o más atributos.
HAVING	Restringe la selección de un grupo de renglones basada en uno o más atributos.
ORDER BY	Ordena los renglones seleccionados en base a uno o más atributos.
UPDATE	Modifica el valor de uno o más atributos en una o más tablas.
DELETE	Borra uno o más renglones de una tabla.
COMMIT	Guarda permanentemente los datos que sufrieron algún cambio.
ROLLBACK	Restaura los datos a sus valores originales.
BETWEEN	Checa que el valor de un atributo se encuentre en un rango correcto.
IS NULL	Checa si el valor de un atributo es nulo.
LIKE	Checa que el valor de un atributo coincida con un patrón dada una palabra.
IN	Checa que el valor de un atributo coincida con una lista de valores.
EXIST	Checa que el resultado de una subconsulta regrese un renglón.
DISTINCT	Limita a que la consulta regrese valores únicos.

Cuadro 2.3: Instrucciones de LMD

Dentro de SQL existen funciones denominadas funciones de agregación, dichas funciones toman un conjunto de valores como entrada y devuelven un solo valor. SQL integra cinco funciones principales de agregación que son [11]:

1. Promedio: *avg*.
2. Mínimo: *min*.
3. Máximo: *max*.
4. Total: *sum*.
5. Conteo: *count*.

En este trabajo se utilizan las funciones de agregación para realizar las operaciones que los modelos estádisticos necesitan. Las funciones de agregación nos facilitan operaciones sobre las relaciones, realizándolas de manera eficiente.

Dialectos SQL

El ANSI[4] publicó su primer estándar SQL en 1986 y un segundo estándar ampliamente adoptado por los proveedores en 1989. ANSI realizó actualizaciones en su estándar en el año de 1992, y fue nombrado como SQL92 ó SQL2, y en el año de 1999 se volvió a actualizar el estándar, y fue nombrado como SQL99 ó SQL3. Cada actualización que realiza la ANSI añade nuevas características y se incorporan nuevas instrucciones y capacidades en el lenguaje. En el estándar SQL99 se manejan extensiones de datos orientadas a objetos, dicho estándar representa la forma ideal de hacer un dialecto de SQL pero la gran mayoría de los proveedores no cumplen formalmente los requisitos que marca SQL99 [8]. Hoy en día los distintos proveedores de SMBD manejan sus propios dialectos de SQL, estos dialectos cambian constantemente debido a que la comunidad de usuarios de los SMBD requieren de nuevas funcionalidades, muchas de estas funcionalidades no se encuentran en los estándares de SQL dados por la ANSI. Las diferencias entre cada dialecto son mínimas, ya que en general todos se basan en el estándar de SQL99. Las diferencias se encuentran principalmente en instrucciones que apoyan a que la funcionalidad sea más completa, algunas de las diferencias pueden ser instrucciones para controlar errores, instrucciones condicionales, manejo de arreglos variables, etc.

[4]Acrónimo de American National Standards Institute.

2.2.3. Rendimiento en una base de datos.

Una de las principales funciones de los SMBD es proveer información oportuna a los usuarios. Las consultas que se realizan a la base de datos pueden ejecutarse mucho más rápido realizando ajustes de rendimiento en el SMBD, que consiste en encontrar y eliminar los cuellos de botella, así como añadir hardware apropiado como puede ser la memoria RAM o discos duros más rápidos. El tiempo de respuesta de una consulta depende de varios factores, hay aspectos que se consideran de alto nivel, como el diseño del esquema, así como las transacciónes a los parámetros de la base de datos, otros tales pueden aparecer en el tamaño de buffer[5], hasta llegar a los problemas de hardware, todo esto puede afectar críticamente el rendimiento de una aplicación. Cada uno de estos aspectos se pueden ajustar de modo que se obtenga una mejora en el rendimiento. Para asegurar un buen rendimiento se puede realizar un conjunto de actividades y procedimientos, a fin de garantizar que una consulta sea procesada por el SMBD en la mínima cantidad de tiempo posible. Algunas acciones pueden ser las siguientes:

- Al realizar consultas a la base de datos hay que evitar usar *SELECT **, evitando así llamar datos que probablemente no se necesiten, lo mejor es limitar las consultas a las columnas que se necesiten.

- Otra técnica utilizada ampliamente en los sistemas cliente-servidor para reducir el costo de las comunicaciónes es utilizar procedimientos almacenados, las consultas a la base de datos se almacenan en el SMBD en forma de procedimientos. Los clientes pueden invocar estos procedimientos almacenados, en lugar de enviar la petición de una serie de consultas.

- Evitar escribir ó reescribir consultas con subconsultas anidadas. Las consultas complejas que contienen subconsultas anidadas no son correctamente optimizadas, es mejor usar la instrucción *JOIN* ya que casi siempre el rendimiento es mejor.

Algunas acciones que se pueden realizar a nivel de hardware puede ser agregar discos o usar un sistema RAID, sí el disco es un cuello de botella, agregar más memoria RAM, sí el tamaño de buffer de disco es un cuello de botella, o el reemplazar el procesador por uno más rápido, sí el uso de la CPU es un cuello de botella. La siguiente acción a realizar consiste en ajustar los parámetros del SMBD, tales como el tamaño del buffer y los intervalos de los puntos de control. El conjunto exacto de los parámetros del SMBD que

[5]Un buffer es un espacio de memoria en una computadora, en el que se almacenan datos de manera temporal, normalmente para un uso único.

pueden ser ajustados depende del sistema de base de datos con el que se trabaja. La mayoría de los SMBD proporcionan manuales de información sobre los parámetros que se pueden ajustar, y cómo se deben elegir los valores para un mejor rendimiento.[11, 4].

Teniendo en cuenta estos factores, y realizando todas las mejoras de rendimiento posibles se espera que el SMBD utlilice correctamente el optimizador de consultas, dicho optimizador usa reglas de equivalencia para generar de manera sistemática expresiones equivalentes a una consulta dada en el álgebra relacional pero de ejecución más eficiente. Así se puede otorgar de la manera más eficiente posible la petición de información al SMBD. Además de que el sistema pueda crear un plan de evaluación que minimice el costo de ejecución[11].

En la figura 2.4 se ejemplifica de forma básica cómo se realiza el proceso de petición de consultas a un SMBD.

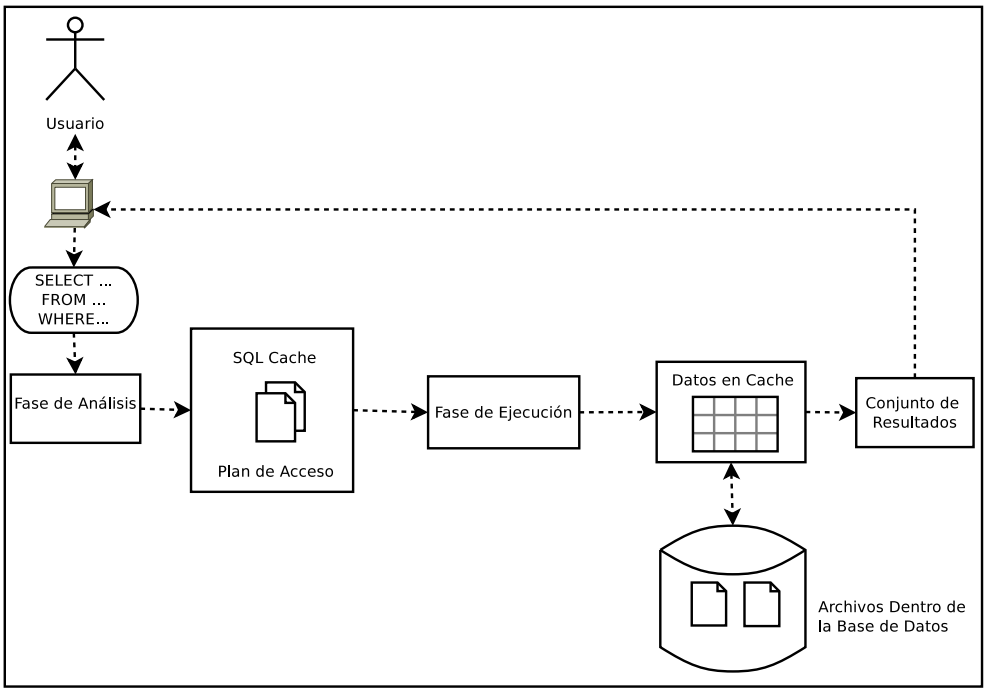

Figura 2.4: Proceso de petición de consultas.

Antes de empezar el procesamiento de una consulta, la fase de análisis del sistema manejador debe de traducir la consulta a un formato interno utilizable. Durante la generación del formato interno, el analizador comprueba la sintaxis del usuario, y verifica que el nombre de las relaciones existan en la base de datos. En la fase del plan de acceso, el sistema manejador construye un árbol para transformar la consulta en una expresión de álgebra relacional y otorgar un plan de ejecución. En la fase de ejecución, el manejador toma el plan de ejecución, lo ejecuta y accede a las relaciones para obtener los datos que fueron pedidos en la consulta para devolver la respuesta en una nueva relación [11].

2.3. Definición de Estimadores y Métodos Estadísticos

La Estadística se divide en dos grandes áreas [12]:

- Estadística descriptiva. Se dedica a la descripción, visualización y resumir datos a partir de un fenómeno de estudio.
- Estadística inferencial. Se dedica a la generación de métodos, estimadores, inferencias y predicciones asociadas a los fenómenos en cuestión teniendo en cuenta la aleatoriedad de las observaciones.

En Estadística, una *muestra* es un subconjunto de casos o individuos de una población. La muestra se obtienen con la intención de realizar inferencias acerca de una población, por lo cual debe ser representativa de la misma.
Un *estimador estadístico* es una medida estadística que específica cómo utilizar datos de una muestra para estimar un parámetro desconocido de la población [12].
Un *método estadístico* se asume como un proceso de obtención, representación, simplificación, análisis, interpretación y proyección de una característica, variables o valores numéricos de un estudio, para una mejor comprensión de la realidad y poder realizar toma de decisiones [12].
Para este trabajo se utiliza la estadística inferencial.

2.3.1. Propiedades de un Estimador Estadístico

Los estimadores estadísticos poseen las siguientes propiedades [12]:

1. Sesgo. Se denomina sesgo a la diferencia entre el valor esperado del estimador y el verdadero valor del parámetro a estimar.
2. Eficiencia. Se dice que un estimador es más eficiente o más preciso, si la varianza[6] es menor. La eficiencia de un estimador está limitada por las características de la distribución de probabilidad de la muestra de la que proceden.
3. Consistencia. Si no es posible emplear un estimador de mínima varianza, el requisito mínimo para un estimador es que a medida que el tamaño de la muestra crece, el valor del estimador tienda a ser el valor del parámetro.

[6]La varianza es una medida de dispersión definida como la esperanza del cuadrado de la desviación de una variable respecto a su media.

4. Robustez. Se dice que un estimador es robusto, si al atribuir a la población un determinado tipo de función que, en realidad no es la correcta, no altera de manera significativa los resultados que este proporciona.

A continuación, se muestran las definiciones formales de los estimadores y métodos estadísticos inferenciales usados.

Estimador Razón

Supóngase una población formada por N unidades, $\{U_1, ..., U_n\}$, y se fijan en dos características (X, Y) para cada unidad. Siendo X la variable objeto de estudio y Y una variable auxiliar correlacionada con X [9].
Se llama ***razón*** a $R = \frac{X}{Y}$ y su estimador viene dado por la expresión:

$$\hat{R} = \frac{\sum_{i=1}^{n} X_i}{\sum_{i=1}^{n} Y_i} = \frac{\hat{X}}{\hat{Y}} = \frac{\overline{x}}{\overline{y}} \tag{2.2}$$

El estimador razón resulta muy apropiado cuando se presenta una relación de proporcionalidad directa entre la variable de estudio (X) y otras variables auxiliares (Y).

Estimador de Proporción

Una proporción es la media de una variable dicótoma[7], en donde los miembros de una clase reciben el valor $Y_i = 1$ y las no miembros el valor $Y_i = 0$. A esta variable se le llama binomial. En una binomial, el valor asignado a la variable es 1 si el elemento pertenece a la clase definida, y 0 si no pertenece a ella.

$$NP = Y = \sum^{N} Y_i \tag{2.3}$$

Es el número de elementos en la población que pertenecen a la clase, y $P = \overline{Y} = Y/N$ es su proporción entre los elementos de su población. Entonces $Q = 1 - P$ es la proporción en la población de los que no pertenecen, y $NQ = N(1 - P) = N - NP$ es su número en la población [7].

[7]Una varible dicótoma es aquélla que permite sólo dos opciones a elegir por ejemplo, blanco o negro, positivo o negativo.

2.3.2. Muestreo Estadístico

El muestreo estadístico es un método de investigación donde se seleccionan sub-grupos de datos de un grupo más grande conocido como población objetivo. El muestreo estadístico se conforma por dos aspectos importantes que son:

1. **Proceso de selección**. Consiste en las reglas y operaciones mediante las cuáles se incluyen en la muestra algunos miembros de la población objetivo.

2. **Proceso de estimación**. Calcula la estadística de la muestra, estas son estimaciones muestrales de valores de la población.

Los elementos de una población son las unidades de muestreo de las que se busca información; pero son los individuos, las unidades elementales que forman la población acerca de la cual se requieren hacer inferencias. La población se define como el agregado de los elementos, y los elementos son las unidades básicas que forman y definen una población. Dado lo anterior las unidades elementales de muestreo contienen a los elementos de una población y se usan para seleccionar la muestra[7].

Método de Muestreo por Probabilidad Proporcional al Tamaño

Se supone que en la población hay N unidades de muestreo compuestas que contienen un total de M unidades elementales, es decir: [9]

$$M = \sum_{i=1}^{N} M_i \tag{2.4}$$

Mediante este método, se selecciona con reposición de la población para la muestra la unidad compuesta U_i de tamaño M_i. Como se trata del modelo clásico sin reposición, se procede a retirar de la población las M_i unidades elementales que componen la unidad de muestreo compuesta U_i antes de proceder a la selección para la muestra de la siguiente unidad de muestreo compuesta. Cuando se realiza la siguiente selección ya faltan de la población M_i unidades elementales. Con este método la probabilidad de seleccionar la unidad U_i es $P_i = M_i/M = p(u_i)$ con $i = 1, 2, ..., N$. Dado lo anterior se cumple que :

$$P_i = \frac{M_i}{M} = \frac{M_i}{\sum_{i=1}^{N} M_i} \Rightarrow \sum_{i=1}^{N} P_i = \sum_{i=1}^{N} \frac{M_i}{M} = \frac{\sum_{i=1}^{N} M_i}{M} = \frac{M}{M} = 1 \tag{2.5}$$

con lo que la probabilidad del método esta definido correctamente.

Método de Muestreo Aleatorio Simple

El muestreo aleatorio simple con reposición es un procedimiento de selección con probabilidades iguales, que consiste en obtener la muestra unidad a unidad de forma aleatoria, la muestra se selecciona con reposición, es decir se reponen las unidades elementales extraídas a la población. Suponiendo en todo momento que el tamaño de la población es de N y el tamaño de la muestra es n y con probabilidades iguales, se realiza la selección de la siguiente unidad en la muestra con probabilidad $P_i = 1/N$, y para cada unidad elemental la muestra es equiprobable como se muestra en la siguiente ecuación:

$$P(u_1, u_2, ..., u_n) = P(u_1)P(u_2)...P(u_n) = (1/N)(1/N)...(1/N) = 1/(N^n) \tag{2.6}$$

2.4. Resumen

En este capítulo se introdujeron algunos conceptos y definiciones acerca de los sistemas de base de datos, así como las definiciones matemáticas formales de los métodos estadísticos de muestreo aleatorio simple y muestreo por probabilidad al tamaño, así como de los estimadores estadísticos de razón y proporción.

Para implementar correctamente los métodos y estimadores estadísticos tratados en este capítulo, es importante conocer el funcionamiento y las características que posee un sistema de base de datos. La primera fase consiste en el diseño de la base de datos, para realizar dicho objetivo se utiliza el modelo relacional, que es el más usado dentro de los SMBD. Al tener el diseño de la base de datos, es necesario implementar las estructuras de la base de datos que contienen los datos para la consulta de información, para realizar estas acciones se usa el lenguaje que provee el SMBD llamado SQL. El lenguaje SQL está dividido en dos categorías que son:

- ***Lenguaje de definición de datos (LDD)***. Este lenguaje proporciona todas las instrucciones necesarias para la creación de las estructuras de la base de datos como tablas, vistas, índices etc.
- ***Lenguaje de manipulación de datos (LMD)***. Este lenguaje proporciona todas las instrucciones necesarias para la manipulación y extracción de información dentro de una base de datos.

La última fase del desarrollo es lograr que las consultas realizadas a la base de datos para extraer información sean lo más eficientes posibles. Para poder lograr esta última fase, se requiere realizar ajustes en el hardware de la computadora, si se es posible, además de realizar ajustes en los parámetros de configuración de el SMBD. Al terminar estas acciónes la última fase de tuning o mejora de rendimiento en una base de datos es optimizar las consultas que se realizan a la base de datos, donde el analizador del sistema manejador pueda ejecutar eficientemente las peticiones que se realizan.

Por otro lado se presentan las características que conforman un método y un estimador estadístico, para explicar de forma general su uso y en que casos se pueden utilizar. Los métodos y estimadores estadísticos desarrollados en este trabajo pertenecen a la rama de la estadística inferencial, que es aquélla que estudia las inferencias y predicciones de algún fenómeno en una población.

Capítulo 3

Trabajo Relacionado

En este capítulo se presenta el artículo *Set-Oriented Mining for Association Rules in Relational Databases* [6], cuyo trabajo está relacionado con la ejecución de algoritmos para determinar reglas de asociación, dichos algoritmos se expresan en consultas SQL, se discute la eficiencia y optimización de dichos algoritmos. En este caso, el artículo presenta un algoritmo denominado SETM. El algoritmo SETM usa funciones básicas del sistema manejador de base de datos, como ordenaciones y uniones. El algoritmo SETM es simple, rápido y estable mostrando que es fácil desarrollar algoritmos de minado de datos usando consultas de SQL .

3.1. Introducción

La competitividad de las empresas hoy en día depende de la calidad en la toma de decisiones. Por lo tanto, no es de extrañar que las empresas a menudo traten de aprender de las transacciones y decisiones tomadas en el pasado, con el fin de mejorar la calidad de las decisiones tomadas en el presente o el futuro. Con el fin de apoyar este proceso se recolectan, almacenan y procesan grandes cantidades de datos. Estos datos son analizados para obtener información relevante [6]. A este proceso se le denomina *minería de datos*. El término minería de datos se refiere al proceso de análisis de forma semiautomática sobre grandes conjuntos de datos para encontrar patrones que resulten útiles. La minería de datos es muy importante para la toma de decisiones en diferentes ámbitos, un proceso de minería de datos en general consta de los siguientes pasos:

1. **Selección del conjunto de datos**. Se refiere a las variables objetivo, aquellas que se quiere predecir, calcular o inferir, así como a las variables independientes que sirven para hacer el cálculo o proceso.

2. **Análisis de las propiedades de los datos**. Se refiere a identificar la presencia de valores atípicos.

3. **Transformación del conjunto de datos de entrada**. Este paso es el preprocesamiento de datos, y tiene el objetivo de realizar un análisis previo para adaptar la mejor técnica de minado de datos para resolver el problema.

4. **Técnica de minería de datos**. Se aplican los algoritmos dados para la resolución del problema.

5. **Análisis de resultados**. Determina si la información aporta un conocimiento nuevo para tomar correctas decisiones.

6. **Evaluación**. Determina si las conclusiones son suficientemente satisfactorias. En el caso de haber obtenido varios modelos mediante el uso de distintas técnicas, se deben comparar los modelos en busca de aquel que se ajuste mejor al problema.

La técnica de minería de datos, aporta un modelo de conocimiento, dicho modelo representa patrones de comportamiento observados en los valores de las variables del problema o relaciones de asociación entre dichas variables. Si al realizar técnicas de minería de datos no se tienen los resultados esperados, se debe alterar alguno o todos los pasos anteriores para generar nuevos modelos de conocimiento y así obtener resultados válidos y satisfactorios (no se profundizará en el tema de minería de datos, ya que, está fuera de los objetivos de este trabajo).

3.2. Definición del problema

El artículo *Set-Oriented Mining for Association Rules in Relational Databases* del autor Arun Swami considera el problema de encontrar reglas de asociación en una base de datos, él toma como muestra una base de datos que registra todas las compras que realizan los clientes a una tienda. Los datos recolectados se almacenan en una base de datos utilizando una tabla llamada SALES (trans_id , item), para cada compra que realiza un cliente se inserta una tupla que corresponde a la venta de los artículos que compró.
Para encontrar una regla de asociación, se necesita explorar las transacciones en la tabla SALES para encontrar patrones recurrentes. Para determinar un patrón en la tabla SALES se define de la siguiente manera:

- Si los clientes de la tienda compran los artículos A,B y C en una transacción y ocurre frecuentemente se considera que dicha compra es un patrón.

De esta observación, se puede concluir que la regla de asociación es la siguiente:

$$A \wedge B \Rightarrow C \tag{3.1}$$

Es decir, si un cliente compra el artículo A y el artículo B por consecuencia se comprará el artículo C, dada la regla de asociación anterior a los artículos A y B se les denomina antecedentes de la regla y el artículo C es la consecuencia de la regla. Para decidir si una regla de asociación se puede tomar como válida es necesario verificar que cumpla con ciertas restricciones. Las restricciones que debe cumplir una regla de asociación son el *apoyo* y la *confianza*. Se define al apoyo para una regla de asociación, a la frecuencia con la que aparece un patrón en una tabla de base de datos. El factor de confianza para una regla de asociación indica el número de veces que se cumple tanto el antecedente como la consecuencia. Dada la regla de asociación $A \wedge B \Rightarrow C$, se calcula el factor de confianza como $|ABC|/|AB|$, donde $|ABC|$ denota el apoyo para el patrón ABC. Para delimitar las reglas de asociación que son útiles se toman en cuenta el apoyo mínimo y el factor de confianza, estos deben de ser mayor a un cierto valor determinado.
Un ejemplo dado por el autor del artículo para generar todos los patrones de dos artículos (x, y) se expresa de la siguiente manera en SQL:

$$\begin{aligned}&\textbf{SELECT}\, r_1.transid, r_1.item, r_2.item\\ &\textbf{FROM SALES}\, r_1, \textbf{SALES}\, r_2\\ &\textbf{WHERE}\, r_1.transid = r2.transid\end{aligned}$$

Para cada par de artículos (x, y) se cuenta el número de id de las transacciones a fin de encontrar el número de transacciones que cumplan el apoyo mínimo para un patrón, cabe mencionar que (y, x) es equivalente. Para generar la regla de asociación de tres elementos, sólo basta unir los resultados de la consulta anterior con la tabla SALES, y así sucesivamente para las reglas de asociación de cuatro,cinco, seis elementos [6].

3.3. Algoritmos de solución

Existen diferentes maneras de resolver el problema para encontrar reglas de asociación en una base de datos. La diferencia entre cada solución es el rendimiento que nos proporcionan dichas soluciones, a continuación se explican brevemente los dos tipos de soluciones que en efecto nos ayudan a resolver el problema pero su eficiencia no es la adecuada.

3.3.1. Uniones en ciclos anidados

Las transacciones de los clientes están guardadas en la relación SALES(trans_id ,item). Usando esta relación primero se genera el conteo para cada objeto x, es decir, el número de transacciones que contienen al objeto x. Se comprueba que dicho objeto cumple con el soporte mínimo y se almacena el resultado en la relación C_1(item,count), se expresa de la siguiente manera en SQL [6]:

INSERT INTO C1
SELECT $r_1.item,$ **COUNT (*)**
FROM SALES r_1
GROUP BY $r_1.item$
HAVING COUNT(*) $>= min\_support$

El siguiente paso es generar todos los patrones (x,y) y checar si cumplen el criterio de apoyo mínimo. Para un artículo específico A es fácil expresarlo. Por ejemplo todos los patrones (A,y) se pueden generar al volver a unir la tabla SALES, y se puede expresar en SQL de la siguiente manera [6]:

SELECT $r_1.item,\ r_2.item,$ **COUNT (*)**
FROM SALES r_1, **SALES** r_2
WHERE $r_1.trans_id = r2.trans_id$
AND $r_1.item =' A'$
AND $r_2.item <>' A'$
GROUP BY $r_1.item, r2.item$
HAVING COUNT(*) $>= min\_support$

Esta consulta solo genera patrones con un artículo específico en la primera posición. La consulta debe ser generalizada y ordenada para generar patrones arbitrarios. La siguiente consulta en SQL genera cualquier patrón en cualquier orden cumpliendo el apoyo mínimo [6]:

INSERT INTO C_k
SELECT $r_1.item, ...,\ r_k.item,$ **COUNT (*)**
FROM $C_{k-1}\ c$, **SALES** $r_1, ...,$ **SALES** r_k
WHERE $r_1.trans_id =, ..., rk.trans_id$
AND $r_1.item = c.item_1$
AND $r_{k-1}.item = c.item_{k-1}$ **AND** $r_k.item > r_k.item$
GROUP BY $r_1.item, ..., rk.item$
HAVING COUNT(*) $>= min\_support$

Con la consulta anterior se encuentran todas las posibles reglas de asociación generadas de longitud 1 a k.

El problema que se encuentra al utilizar ésta solución es el tiempo que tarda en encontrar todas las reglas de asociación, suponiendo que la tabla contiene cerca de dos millones de tuplas, el tiempo total es de once horas aproximadamente en encontrar dichas reglas de asociación[6].

3.3.2. Uniones con ordenamientos mezclados

Esta estrategia de solución requiere de generar relaciones intermedias de la forma R_i(trans_id, $item_1$,...,$item_i$), cada relación que se genera a su vez se ordena y se obtienen todos los patrones no importando su orden. Se expresa de la siguiente forma con SQL [6]:

INSERT INTO Rk
SELECT $p.trans\_id, p.item1, ..., p.itemk - 1, q.item$
FROM Rk-1 p, **SALES** q
WHERE $q.trans_id = p.trans_idAND$
$q.item > p.itemk - 1$

Una vez generados todos los patrones de longitud k en R_k ahora sólo es necesario contar que patrones en R_k cumplen con el apoyo mínimo. Se obtiene de la siguiente manera [6]:

INSERT INTO Ck
SELECT $p.item1, ..., p.itemk$ **COUNT (*)**
FROM Rk p
GROUP BY $p.item1, ..., p.itemk$
HAVING COUNT(*) >=: $min\_support$

Sólo falta realizar las ordenaciones sobre los campos (trans_id, $item_1$,...,$item_k$), se implementa en SQL con la siguiente consulta [6]:

INSERT INTO Rk
SELECT $p.trans\_id, p.item1, ..., p.itemk$
FROM Rk p, **Ck** q
WHERE $p.item_1 = q.item_1AND$
.
.
.
$p.item_k = q.item_k$
ORDER BY $p.trans_id, p.item_1, ..., p.item_k$

Con esta solución el tiempo que tarda en encontrar las reglas de asociación de tres elementos es de diez minutos, ya que el crear las relaciones intermedias es de orden secuencial, mostrando así que la solución de uniones con ordenamientos mezclados es más eficiente que la de ciclos anidados[6].

3.4. Algoritmo SETM

Aplicando el algoritmo SETM, al problema de la generación de reglas de asociación, se explica de la siguiente manera [6]:

- El algoritmo consiste de un simple ciclo, en el cual se realizan dos operaciones de ordenación y una combinación.
- La primera ordenación se necesita para poder realizar la combinación.
- La segunda ordenación se realiza para generar los recuentos de manera eficiente.
- Los recuentos de cuántas veces se repite cierta regla de asociación, implica solamente un recorrido secuencial de las relaciones.

En el algoritmo SETM para encontrar los patrones de longitud k, se deben de considerar todas las posible combinaciones de $k-1$ objetos en el antecedente. El elemento restante no utilizado en las combinaciones es el consecuente. Para cada combinación del antecedente y el consecuente se checa si el factor de confianza cumple o excede el mínimo factor de confianza. Si el factor de confianza es alto entonces se guarda dicha regla. El algoritmo SETM se muestra en el siguiente pseudocódigo [6]:

```
k := 1;
sort R1on item;
C1 := generate counts from R1 ;
repeat
k := k + 1;
sort Rk-1 on trans_id, item1,...,itemk-1;
Rk := merge-scan Rk-1, R1;
sort Rk on item1,..., itemk;
Ck := generate counts from Rk;
Rk := filter Rk to retain supported patterns;
until Rk ={}
```

Al ejecutar el algoritmo SETM aplicado a la tabla SALES que contiene dos millones de tuplas, con la diferencia de variar el valor de apoyo mínimo los tiempos de ejecución se muestran en el cuadro 3.1.

Los siguientes tiempos muestran que al comparar las tres soluciones para minar datos el algoritmo SETM es el más estable.

Apoyo Mínimo (%)	Tiempo de Ejecución (segundos)
0.1	6.90
0.5	5.30
1	4.64
2	4.22
5	3.97

Cuadro 3.1: Tiempos de Ejecución [6]

3.5. Conclusiones

La mayor contribución de este trabajo es mostrar que el algoritmo para resolver el problema de encontrar reglas de asociación puede ser hecho con funciones básicas como ordenaciones y uniones de SQL; y esto puede ser fácilmente implementado en un sistema manejador de base de datos. Un aspecto a tomar en cuenta es la forma de desarrollar el algoritmo en lenguaje SQL dentro del sistema de base de datos, ya que si la consulta realizada en SQL es correctamente implementada se obtiene un desempeño eficiente.

3.6. Resumen

En el artículo mencionado [6] se define el problema de como poder obtener reglas de asociación, las reglas de asociación son patrones de repetición que se pueden encontrar al analizar los datos contenidos en una base de datos.

Para la solución de este problema, se propone un algoritmo llamado SETM, el cual es implementado dentro del sistema manejador de base de datos y programado en SQL, utilizando funciones básicas como ordenaciones, uniones y combinaciones.

El algoritmo SETM identifica las reglas de asociación y las discrimina dependiendo su apoyo mínimo y su factor de confianza. El apoyo mínimo es la frecuencia con la que aparece un patrón en una tabla de base de datos, mientras que el factor de confianza calcula que tan confiable es una regla de asociación para poder ser tomada en cuenta.

En conclusión, se observa que se pueden desarrollar algoritmos en lenguaje SQL, con el que se pueden procesar grandes cantidades de datos, a fin de obtener un buen rendimiento y estabilidad.

Capítulo 4

Análisis de estimadores y métodos estadísticos

En este capítulo, se describe el diseño de la base de datos y la implementación en SQL de los métodos estadísticos de probabilidad proporcional al tamaño y muestreo aleatorio simple, así como los estimadores estadísticos de razón y proporción, dentro de un sistema manejador de base de datos.

4.1. Marcos Muestrales

El diseño de la base de datos para el cálculo de resultados electorales se basa en el marco muestral, definido por el Instituto Federal Electoral (IFE). El marco muestral es la información que puede referirse a cierta área geográfica, una lista de unidades de viviendas o personas de una población. Los marcos de la geografía electoral se construyen por un conjunto de reglas político-geográfica. Dichas reglas definen las siguientes áreas geográficas [1]:

- Circunscripciones Plurinominales. Área geográfica integrada por un grupo de entidades federativas.
- Entidad Federativa. Cada una de las porciones geográficas que integran las treinta y dos entidades federativas.
- Municipio. Constituye la unidad política administrativa básica de la división territorial y de la organización de las entidades federativas del país.
- Distritos Federales Electorales. Espacio geográfico delimitado para fines electorales. De acuerdo con la ley, el país se divide en trescientos distritos electorales uninominales. A cada distrito electoral se le asigna una clave de dos dígitos.

- Sección Electoral. Es la fracción básica territorial de los distritos electorales uninominales, para la inscripción de los ciudadanos en el padrón electoral.

- Sección Mixta. Área geográfica conformada por un grupo de manzanas que forman parte de una localidad urbana, además de presentar una o más localidades rurales.

- Sección Urbana. Es aquélla conformada por un conjunto de manzanas bien definidas y que forman parte de una localidad urbana, misma que normalmente presentan nomenclatura de calles.

Teniendo en cuenta las reglas anteriores, se construyen cuatro marcos muestrales que son:

- Marco por entidad federativa. Se reconoce a cada una de las treinta dos entidades federativas que componen los Estados Unidos Mexicanos, por separado.

- Marco por municipios. Se toman los municipios más poblados de cada entidad federativa como los más representativos y los municipios con menor población se agrupan para formar un municipio.

- Marco por secciones federales. La entidad federativa se divide en secciones federales que designa el IFE dependiedo de estándares como el aumento poblacional en el listado nominal.

- Marco de votantes por secciones. Las secciones federales se componen por casillas de votación, dichas casillas recolectan el voto emitido por las personas.

Los marcos muestrales abarcan todas las unidades de muestreo. Dichos marcos se actualizan anualmente con información proporcionada por el IFE, así como las tasas de participación de las votaciones anteriores.

4.1.1. Estratificación

La estratificación es un método estadístico que divide un conjunto de datos en subconjuntos homogéneos, a cada subconjunto se le denomina estrato. La división de los datos se efectúa en base a diversos factores que se identifican al momento de obtener los datos [3].
En este problema la estratificación delimita las divisiones electorales que se tienen. Por lo tanto se hacen dos tipos de estratificación: geográfica y política. La geográfica toma en cuenta únicamente las divisiones geográficas dadas por el IFE.
La política, consiste en observar las preferencias partidistas, agrupando a las secciones que tienen comportamientos similares.

4.1.2. Notación

R : Número total de regiones (estratos).

S_m : Número de secciones totales en el estrato m.

s_m : Número de secciones, en muestra[1], en el estrato m.

LN : Lista nominal otorgada por el IFE.[2] k.

f^T : Factor total de expansión.

En base a está notación se generan nuevas variables

V_k^j es el número de votos, del partido j en la sección k.

$V_k := \sum_j V_k^j$ es el número de votos, en la sección k.

$VP_k^j := V_k^j / V_k$ es la proporción de votos, del partido j en la sección k.

$\overline{VP^j} = \frac{\sum_{k=1}^{s} VP_k^j}{s}$ es el promedio de la proporción de votos del partido j.

$\widehat{V_m}$ es el número total de votos que se esperan en la región m.

$\widehat{V}$ es el número total de votos que se esperan.

[1]Se refiere a las secciones elegidas para realizar el cálculo

[2]La lista nominal es la relación electrónica e impresa que contiene a los ciudadanos debidamente registrados en el Padrón Electoral

para $k = 1, 2, \ldots, s$.

Para hacer diferencia entre los votos obtenidos el día de la elección, al de los votos obtenidos en elecciones pasadas (que se requieren para hacer comparaciones), se agrega una variable A para identificar los votos actuales o una variable P para los votos pasados.
Quedando entonces VA_k^j, VP_k^j, VA_k, VP_k, VAP_k^j, VPP_k^j, $\overline{VAP^j}$, $\overline{VPP^j}$.
Entonces, el coeficiente de correlación de los votos actuales con los pasados del partido j es:

$$\rho^j = \frac{\sum_k \left[\left(VAP_k^j - \overline{VAP^j}\right)\left(VPP_k^j - \overline{VPP^j}\right)\right]}{\sqrt{\sum_k \left(VAP_k^j - \overline{VAP^j}\right)^2 \sum_k \left(VPP_k^j - \overline{VPP^j}\right)^2}}$$

4.2. Diseño e implementación de la base de datos

El diseño de la base de datos, se basa en los marcos muestrales dados por el Instituto Federal Electoral, implementado con el modelo relacional, para un adecuado funcionamiento y así poder realizar la ejecución de modelos estadísticos en SQL.
La implementación para el diseño de la base de datos requiere de cuatro relaciones para demarcar la totalidad de las reglas político-geográficas requeridas por el Instituto Federal Electoral.
Las relaciones utilizadas son las siguientes:

- Votos Pasados. Contiene los siguientes atributos: estado, sección, partido1, partido2, partido3.
- Votos Actuales. Contiene los siguientes atributos: estado, sección, partido1, partido2, partido3.
- Pesos Geográficos. Contiene los siguientes atributos: estado, sección, listanominal, estrato, peso.
- Pesos Políticos. Contiene los siguientes atributos: estado, sección, listanominal, estrato, peso.

En la figura 4.1 se muestra el diseño de la base de datos.

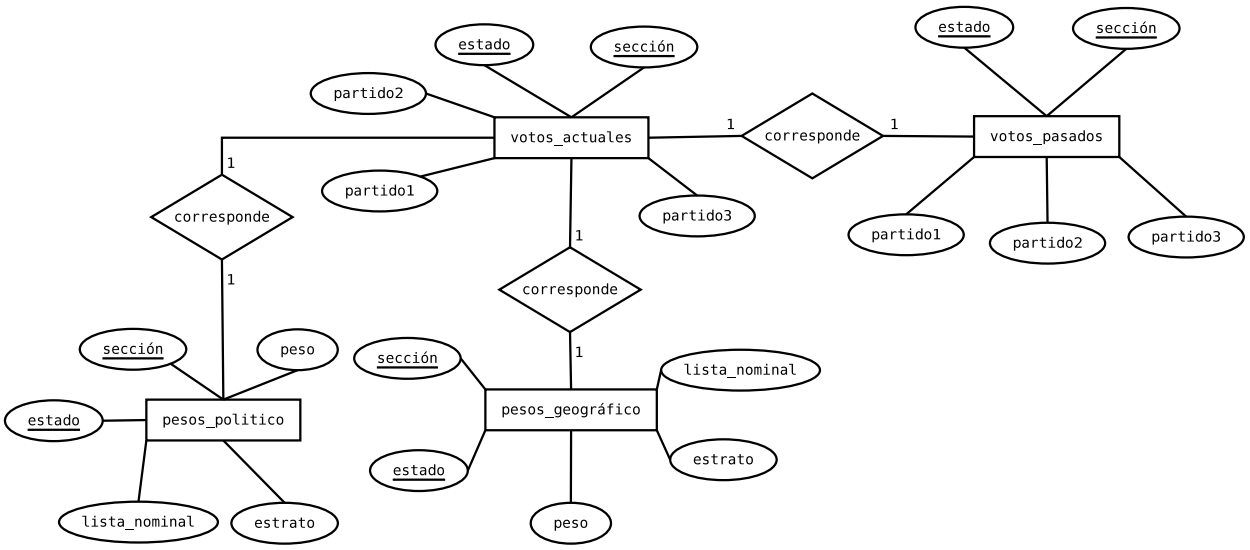

Figura 4.1: Diagrama de la base de datos para el cálculo de votaciones en México

Realizado el diseño de la base de datos, se requiere crear las relaciones y dotarlas de datos para después implementar los algoritmos en SQL y así realizar el cálculo de resultados electorales, aplicando los métodos y estimadores estadísticos descritos en este trabajo.

4.3. Implementación de Estimadores y Métodos Estadísticos en SQL

Este trabajo está enfocado en implementar algoritmos estadísticos con sentencias y funciones básicas de SQL, las cuales permiten trabajar con grandes cantidades de datos, y a su vez, obtener resultados en tiempos delimitados, ejecutando dichos algoritmos de manera simultánea.

4.3.1. Implementación de Estimador de Razón

En el modelo razón se suponen dos variables correlacionadas, de tal manera que cambian de manera proporcional. En este caso, se implementa en dos fases: En la primer fase, se estima la razón en cada estrato; en la segunda fase, se suman los resultados otorgados en la primera fase, ponderando[3] con

[3]Determinar el valor dentro de una serie de datos.

los pesos de cada estrato, para obtener los resultados.

$$\begin{aligned}\widehat{V^j} &= \sum_m \sum_{k=1}^{s} \widehat{V^j_{(k)}} * (f_k^T) \\ &= \sum_m \sum_{k \in s} \widehat{V^j_{(k)}} * (f_k^T) \\ &= \sum_m \sum_{k \in s} \widehat{VA^j_{(k,m)}} \cdot \left(\frac{LN_m}{LN_{(k,m)}} \right)\end{aligned}$$

donde $\widehat{VA^j_{(k,m)}} = \widehat{R_m} VP^j_{(k,m)}$, con $\widehat{R_m} = \frac{\overline{VA^j_m}}{\overline{VP^j_m}}$.

El Programa 4.1 muestra la codificación[4] del modelo razón con sentencias SQL.

```
1  SELECT redondear(va1*100/(va1+va2+va3),2) AS raz1,
2  redondear(va2*100/(va1+va2+va3),2) AS raz2,
3  redondear(va3*100/(va1+va2+va3),2) AS raz3
4  /*SE OBTIENE LA ULTIMA SUMA DE TODOS LOS ESTRATOS*/
5  FROM (SELECT SUM(va1) AS va1, SUM(va2) AS va2,SUM(va3) AS va3
6  /*SE CALCULA Y OBTIENE EL PROMEDIO POR ESTRATO */
7  FROM (SELECT estrato, AVG(va1) AS va1,AVG(va2) AS va2,AVG(va3) AS va3
8  FROM
9  /*SE MULTIPLICAN TODAS LAS VARIABLES A USAR*/
10 (SELECT estado, seccion, votos_pasados.estrato,
11 SUM(CASE WHEN m12!=0 THEN m1*mp12/m12 ELSE 0 END) AS va1,
12 SUM(CASE WHEN m22!=0 THEN m2*mp22/m22 ELSE 0 END) AS va2,
13 SUM(CASE WHEN m32!=0 THEN m3*mp32/m32 ELSE 0 END) AS va3
14 FROM (SELECT votos_pasados.estado, estrato, votos_pasados.seccion,
15 /*SE OBTIENE EL PROMEDIO DE LOS VOTOS PASADOS, POR SECCION Y MULTIPLICANDO
       POR
16 (LN_m / LN_(k,m)) */
17 SUM(CASE WHEN peso>0 THEN CAST(partido1 AS float)/peso END) AS mp12,
18 SUM(CASE WHEN peso>0 THEN CAST(partido2 AS float)/peso END) AS mp22,
19 SUM(CASE WHEN peso>0 THEN CAST(partido3 AS float)/peso END) AS mp32
20 FROM votos_pasados,
21 /*SE OBTIENE LA INVERSA DE (LN_m / LN_(k,m)) Y SE RENOMBRA COMO LA VARIABLE "peso" */
22 (SELECT estado, seccion, pesos2. estrato, peso FROM
23 (SELECT estrato, CAST(sum(lista_nominal) AS float)/lm AS peso FROM
       pesos_politico AS pesos,
24 (SELECT sum(lista_nominal) AS lm FROM pesos_politico) AS total GROUP BY
       estrato,lm) AS pesos2,
25 pesos_politico WHERE pesos2.estrato=pesos_politico.estrato ) AS pesos
26 WHERE votos_pasados.estado = pesos.estado AND votos_pasados.seccion = pesos.
       seccion GROUP BY votos_pasados.seccion,votos_pasados.estado,estrato) AS
       vp,
27 /*SE OBTIENE EL PROMEDIO DE VOTOS ACTUALES, POR SECCION*/
```

[4]La sentencia pesos_politicos puede ser substituida por pesos_geograficos dependiendo el caso que se requiera calcular.

```
28 (SELECT estrato, AVG(partido1) AS m1, AVG(partido2) AS m2, AVG(partido3) AS
       m3
29 FROM votos_actuales AS vc, pesos_politico AS pesos WHERE vc.estado = pesos.
       estado AND vc.seccion = pesos.seccion
30 GROUP BY estrato) AS med1,
31 /*SE OBTIENE EL PROMEDIO DE VOTOS PASADOS, POR ESTRATO*/
32 (SELECT estrato, AVG(partido1) AS m12, AVG(partido2) AS m22, AVG(partido3)
       AS m32 FROM votos_pasados AS vp, pesos_politico AS pesos WHERE vp.estado
       = pesos.estado AND vp.seccion = pesos.seccion  GROUP BY estrato
33 ) AS med2 WHERE med1.estrato = vp.estrato AND med2.estrato = vp.estrato
       GROUP BY estado, seccion, vp.estrato
34 ) AS vc GROUP BY estrato) AS total) AS razon;
```

Programa 4.1: Modelo Razón implementado en lenguaje SQL

4.3.2. Implementación de Método Probabilidad Proporcional al Tamaño

El modelo de Probabilidad Proporcional al Tamaño con reposición se realiza en dos etapas, porque en las unidades de observación (Estratos→Secciones) sólo se seleccionan las secciones, tomándose todas las regiones. De acuerdo a las secciones en muestra que se tenga de cada región.

Formación de Unidades de Muestreo

Para realizar la selección de muestra, la primera etapa consiste en la selección de Unidades Primarias de Muestreo (UPM), que se forman por secciones previamente ordenadas en forma descendente o ascendente por preferencia partidista (según indique la elección federal anterior, es decir, el que obtuvo la mayor cantidad de votos), tomándose todas las casillas de cada UPM (Sección).

$$\widehat{V^j} = \sum_{m=1}^{R} \frac{1}{s_m} \sum_{k=1}^{s_m} (f_{mk}^T \; V_k^j)$$

Sean $\{VP^j_{(1,m)}, VP^j_{(2,m)}, \ldots, VP^j_{(S,m)}\}$ una sucesión de variables ordenadas (por el porcentaje de preferencia partidista), que representan el número de votos emitidos en el pasado, en las secciones de la región m para el partido j

$$\begin{aligned} U_{1,m} &: VP^j_{(1,m)} \\ U_{2,m} &: VP^j_{(1,m)} + VP^j_{(2,m)} \\ &\vdots \\ U_{S,m} &: VP^j_{(1,m)} + VP^j_{(2,m)} + \cdots + VP^j_{(S,m)} = VP^j_m \end{aligned}$$

Una manera de extraer una muestra en el modelo de probabilidad proporcional al tamaño, es que dado un número aleatorio R tal que $1 \leq R \leq (VP/s)$, se eligen las secciones, muestra, cuyos índices (i) satisfagan la siguiente desigualdad:

$$U_{i-1} < R + j(VP/s) \leq U_i, \quad \forall j = 0, 1, \ldots, n-1. \quad U_0 = 0$$

Lo que interesa conocer no es el valor de la muestra, lo que se necesita es reproducir un resultado a nivel población, por lo cual el valor obtenido por la muestra se extenderá a un factor que nos ayude a representar dicha población, este factor es llamado factor de expansión. Para calcular el factor de expansión f^T_{mk} se necesita conocer las probabilidades de selección.

Probabilidades de selección

- La probabilidad de elegir una sección k que está dentro de la región m.

$$p(\text{“}UPM_m\text{”}) \approx \frac{LN_{(k,m)}}{LN_m}$$

Factores de expansión

El factor de expansión a nivel de sección (que es la UPM) se define como:

$$f_{(k,m)} = 1/p(\text{“}UPM_m\text{”})$$

Como se toman en muestra todas las regiones, entonces:
$P(\text{“elegir la Región } m\text{”}) = 1.$

$$\begin{aligned} f^T_{mk} &= f_{(k,m)} \cdot 1 \\ &\approx \frac{\cdot LN_m}{LN_{(k,m)}} \end{aligned}$$

Sustituyendo (4.1) en (4.3.2) se obtiene

$$\begin{aligned} \widehat{V^j} &= \sum_{m=1}^{R} \frac{1}{s_m} \sum_{k=1}^{s_m} (f^T_{mk}\ V^j_k) \\ &= \sum_{m=1}^{R} \frac{1}{s_m} \sum_{k=1}^{s_m} \left(\frac{LN_m}{LN_{(k,m)}}\ VA^j_{(k,m)} \right) \\ &= \sum_{m=1}^{R} \left[\frac{LN_m}{s_m} \sum_{k=1}^{s_m} \left(\frac{VA^j_{(k,m)}}{LN_{(k,m)}} \right) \right] \end{aligned}$$

donde LN es la última lista nominal de la que se tiene información, pues siempre vamos a preferir la información más reciente para ello, si usamos información no actualizada, la información procesada acarreará sesgos.

El Programa 4.2 implementa el modelo de probabilidad proporcional al tamaño.

```
SELECT redondear(SUM(r1)*100/SUM(r1+r2+r3),2) AS ppt1,
redondear(SUM(r2)*100/SUM(r1+r2+r3),2) AS ppt2,
redondear(SUM(r3)*100/SUM(r1+r2+r3),2) AS ppt3
FROM
/*SE OBTIENE LA SUMA Y DIVISION ENTRE EL TAMANO DE MUESTRA (sm)*/
(SELECT estrato,AVG(partido1) AS r1,AVG(partido2) AS r2,AVG(partido3) AS r3,
FROM
(SELECT votos_actuales.seccion,pesos.estrato,
/*EN LA TABLA pesos_politico YA SE TIENE LA INVERSA DE (LN^j_(m) / LN_(k,m)) QUE ES
LA VARIABLE "peso", AQUI SOLO SE OBTIENE LA MULTIPLICACION DE
(LN_m)(VA^j_(k,m) / LN_(k,m)) */
SUM(CASE WHEN peso>0 THEN CAST(partido1 AS FLOAT)/peso END) AS p1,
SUM(CASE WHEN peso>0 THEN CAST(partido2 AS FLOAT)/peso END) AS p2,
SUM(CASE WHEN peso>0 THEN CAST(partido3 AS FLOAT)/peso END) AS p3
FROM
votos_actuales, pesos_politico WHERE votos_actuales.estado=pesos_politico.
    estado AND votos_actuales.seccion=pesos_politico.seccion AND
    votos_actuales.estado=estado GROUP BY votos_actuales.seccion,
    pesos_politico.estrato
) AS respre GROUP BY estrato) AS ppt
```

Programa 4.2: Modelo Probabilidad Proporcional al Tamaño implementado en lenguaje SQL

4.3.3. Implementación del Método Muestreo Aleatorio Simple

El muestreo aleatorio simple sólo considera la votación del día de la elección y con ello, se obtiene el porcentaje correspondiente a cada partido. La representación matemática, es:

$$\widehat{VA^j} = \sum_{\forall m,k} VA^j_{(m,k)} \cdot \frac{N}{n} = N \cdot \widehat{VA^j}$$

entonces el porcentaje correspondiente para cada partido está determinado por:

$$\widehat{VAP^j} = \frac{\widehat{VA^j}}{\sum_i \widehat{VA^i}}$$

El Programa 4.3 implementa el modelo de muestreo aleatorio simple.

```
SELECT redondear(y1*100/(y1+y2+y3),2) AS mas1,
redondear(y2*100/(y1+y2+y3),2) AS mas2,redondear(y3*100/(y1+y2+y3),2) AS
    mas3
/*SE CALCULAN LAS MULTIPLICACIONES CORRESPONDIENTES A LA FORMULA DE MUESTREO
     ALEATORIO SIMPLE*/
FROM (SELECT n*p1/m AS y1, n*p2/m AS y2, n*p3/m AS y3
/*SE CALCULAN LA "N" POBLACIONAL Y LA "n" MUESTRAL, EVITANDO AMBIGUEDADES EN
     EL CODIGO SE RENOMBRAN n y m
Y TAMBIEN SE CALCULAN LAS SUMAS DE LOS VOTOS ACTUALES*/
FROM (SELECT n,COUNT(votos_actuales.seccion) as m, SUM(p1) AS p1, SUM(p2) AS
     p2, SUM(p3) AS p3
FROM (SELECT estado, seccion,SUM(CAST(partido1 AS float)) AS p1, SUM(CAST(
    partido2 AS float)) AS p2, SUM(CAST(partido3 AS float)) AS p3
FROM votos_actuales GROUP BY seccion,estado) AS VC
JOIN (SELECT estado,count(seccion) AS N FROM  pesos_politicos GROUP BY
    estado) AS N ON vc.estado=n.estado
GROUP BY vc.estado,n) AS tot) AS mas
```

Programa 4.3: Modelo de Muestreo Aleatorio Simple implementado en lenguaje SQL

4.3.4. Implementación de Estimador Proporción

El modelo de proporción calcula el porcentaje de votos del partido j (VAP_m^j) por estrato y luego hace una ponderación de este porcentaje a nivel nacional.
La representación matemática es la siguiente:

$$V_m^j = \sum_{k \in s} V_{(k)}^j$$

$$\Rightarrow VAP_m^j = \frac{V_m^j}{\sum_h V_m^h}$$

y

$$\begin{aligned} \widehat{VAP^j} &= \sum_m VAP_m^j * f_m \\ &= \sum_m VAP_m^j * \frac{L_m}{\sum L_m} \end{aligned}$$

El Programa 4.4 implementa el modelo de proporción.

```
/*SE REALIZAN LAS MULTIPLICACIONES Y LA SUMA PARA OBTENER LA PROPORCION DE
    CADA PARTIDO*/
SELECT redondear(SUM(pp1*lm)*100/SUM(lm),2) AS prop1,redondear(SUM(pp2*lm)
    *100/SUM(lm),2) AS prop2,redondear(SUM(pp3*lm)*100/SUM(lm),2) AS prop3
FROM (
/*SE CALCULA EL PROMEDIO DE VOTO POR CADA SECCION*/
SELECT COUNT(seccion),
CASE WHEN SUM(p1+p2+p3)!=0 THEN SUM(p1)/SUM(p1+p2+p3) END as pp1,
```

```
7  CASE WHEN SUM(p1+p2+p3)!=0 THEN SUM(p2)/SUM(p1+p2+p3) END as pp2,
8  CASE WHEN SUM(p1+p2+p3)!=0 THEN SUM(p3)/SUM(p1+p2+p3) END as pp3,lm
9  FROM (SELECT votos_actuales.seccion,p1,p2,p3,lm FROM (SELECT seccion,SUM(
       CAST(partido1 AS float)) AS p1, SUM(CAST(partido2 AS float)) AS p2, SUM(
       CAST(partido3 AS float)) AS p3
10 FROM votos_actuales GROUP BY seccion) AS vc,
11 /*AQUI SE OBTENE LA SUMA DEL LISTADO NOMINAL POR ESTADO Y LO ASIGNAMOS A
       CADA SECCION DEFINIDA DENTRO DEL ESTRATO*/
12 (SELECT seccion,lm
13 FROM (SELECT estrato,SUM(lista_nominal) as lm FROM pesos_politico GROUP BY
       estrato) AS unionestra, pesos_politico WHERE unionestra.estrato=
       pesos_politico.estrato and pesos_politico.estado=estado
14 ) AS inpas WHERE inpas.seccion=vc.seccion) AS uni GROUP BY lm) AS prop
```

Programa 4.4: Modelo de Proporción implementado en lenguaje SQL

4.4. Resumen

El diseño y la implementación de la base de datos se enfoca en los marcos muestrales utilizados por el Instituto Federal Electoral, los cuales se basan en reglas político-geográficas. Dichas reglas explican las divisiones electorales que se tienen en el país, y cómo se conforman los diferentes tipos de marcos muestrales para las elecciones.
La implementación de la base de datos está conformada por las siguientes entidades:

- Votos Pasados.
- Votos Actuales.
- Pesos Geográficos.
- Pesos Políticos.

Con dichas entidades, la base de datos está dotada para realizar el cálculo de resultados electorales con los distintos métodos y estimadores estadísticos.

La implementación de los métodos y estimadores estadísticos se realiza con sentencias básicas de SQL, aprovechando el rendimiento del manejador de base de datos. Utilizamos funciones como SUM, AVG que proporcionan una buena eficiencia en cálculos de sumas y promedios, necesarios en la implementación de los modelos. En los Programas 4.1, 4.2, 4.3 y 4.4 se puede observar la codificación de cada modelo estadístico.

Capítulo 5

Implementación de estimadores y métodos estadísticos así como su programa objetivo para la visualización de resultados

El programa objetivo es la herramienta que se crea para interactuar entre el usuario y la base de datos, para facilitar la petición de resultados. En este capítulo se describen las características y tecnologías usadas en el programa objetivo. En la experimentación de este trabajo, se busca conocer el tiempo que tardan en ejecutarse de manera simultánea los métodos y estimadores estadísticos implementados dentro del Sistema Manejador de Base de Datos, además se explica el porque se eligió esta forma de solución al problema dado, para posteriormente obtener conclusiones en base a ello.

5.1. Descripción

La arquitectura usada para realizar este trabajo es una arquitectura de tipo web. La arquitectura web es una típica arquitectura cliente/servidor, de un lado se encuentra el cliente el cuál realiza peticiones por medio de un navegador web. Por otro lado el servidor, cuya función es atender las peticiones del cliente para proporcionar recursos o información contenida en él. Una buena practica de programación en arquitecturas web es utilizar un modelo de programación de tres capas. El modelo que se utiliza para este trabajo es Modelo-Vista-Controlador.Dicho modelo divide el programa en tres componentes:

- **Modelo**. Es la representación de los datos con la cual el sistema opera,

dicho componente gestiona todos los accesos a los datos con los que se trabajan.

- **Vista**. La vista es la capa en donde el usuario interactua con el programa para realizar peticiones de información .
- **Controlador**. El controlador maneja los eventos que se producen en la vista y notifica al modelo la acción o petición que realiza el usuarios, lo que puede implicar un cambio en el estado del modelo. Es por eso que el controlador sirve de intermediario entre la vista y el modelo.

En la siguiente figura 5.1 se observa cómo funciona el modelo MVC.

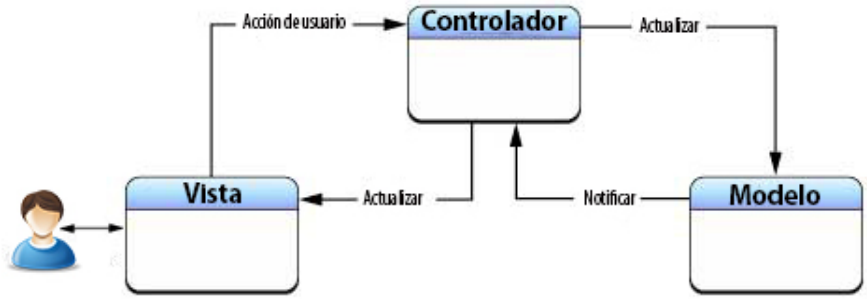

Figura 5.1: Ejemplo del funcionamiento del modelo MVC

En una arquitectura web típica, las operaciones o cálculos se realizan en la capa del controlador, pero qué sucede cuando tenemos que realizar una gran cantidad de operaciones simultáneamente sobre una gran cantidad de datos. Para éste trabajo en específico se tendría que consultar los datos existentes en las tablas de la base de datos, cada tupla que exista dentro de las tablas de la base de datos se tienen que convertir a objetos para poder ser operados dentro de la capa del controlador, pero se tiene que tomar en cuenta que la base de datos se alimenta en tiempo real para realizar el cálculo de resultados electorales, por lo tanto si analizamos el desempeño de la aplicación de está forma, en un principio con pocos datos el comportamiento sería bueno pero entre más datos se obtengan se corre el riesgo de que la memoria se vea saturada por la gran cantidad de objetos que debe contener.
Teniendo en cuenta que los recursos computacionales con que se cuentan son limitados, podemos ver el problema de forma distinta, el Sistema Manejador de Base de Datos cuenta con grandes ventajas que podemos utilizar, una de ellas es poder realizar funciones dentro del Sistema Manejador de Base de Datos para realizar las operaciones internamente sin necesidad de acceder a los datos por medio de la capa del controlador, dado esto el manejador se

encarga de realizar toda la coherencia interna sobre el manejo de los datos evitando así la sobrecarga de memoria por parte del servidor web y además realiza los cálculos necesarios para los estimadores y métodos estadísticos de una forma eficiente.
Para los objetivos de esta tesis la elección de una arquitectura web se da por la portabilidad y la necesidad de tener un programa que pueda responder en tiempo real a las peticiones hechas por los usuarios. Para realizar el programa objetivo se utilizan las siguientes herramientas:

Plataforma de ejecución. Se utiliza un equipo con las siguientes características:

- **Sistema Operativo OS X Mavericks**
  - Versión: 10.9.2.
- **Componentes Físicos**
  - Procesador: Intel Core i7 a 2.9 GHz.
  - Memoria Ram: 8 GB a 1600 MHz DDR3.
  - Disco Duro: TOSHIBA 750GB.

Lenguaje de Programación. Para realizar el programa se utiliza Java Enterprise Edition versión 1.7.

Frameworks. Los frameworks utilizados para el programa objetivo son:

- **JSF 2**. Java Server Faces es una tecnología que ayuda a establecer estándares para desarrollar programas Web. Lo más importante de este framework es simplificar el desarrollo de las interfaces de usuario, además de usar como patrón de diseño el conocido Modelo-Vista-Controlador[2].
- **Primefaces**. Es un componente para JSF 2, que cuenta con un conjunto de componentes que facilitan la creación de componentes web, enriquecidos con acciones como creación de tablas, paneles, botones, etc. Todos ellos con peticiones AJAX[1] [13].

IDE de Desarrollo. El IDE utilizado para desarrollar el programa objetivo es NetBeans 7.3, el cual contiene todas las herramientas necesarias para la ejecución del programa.

[1]AJAX es el acrónimo de Asynchronous JavaScript And XML y es una tecnología que solicita datos al servidor sin interferir en la visualización de la página.

Servidor Web. El Servidor web que se utiliza es Glassfish V3, diseñado para soportar JSF 2.0.

Manejador de Base de Datos. Para este trabajo se utiliza PostgreSQL version 9.3.

5.2. Implementación de programa objetivo

El programa objetivo es la herramienta que ayuda a interactuar al usuario y la base de datos, ya que facilita la petición de resultados ocultando la complejidad de las consultas. La implementación del programa está dividida en dos clases que son:

- DatosModelos.java
- EjecucionModelos.java

El archivo *DatosModelos.java* contiene los atributos que componen los resultados arrojados por la consulta, los atributos que los conforman son:

- NombreModelos. Este atributo está compuesto por un string para guardar el nombre del modelo calculado.
- ResultadoPartidoUno. Este atributo está compuesto por un double, el cual guarda el valor resultante de aplicar el modelo estadístico para el partido uno.
- ResultadoPartidoDos. Este atributo está compuesto por un double, el cual guarda el valor resultante de aplicar el modelo estadístico para el partido dos.
- ResultadoPartidoTres. Este atributo está compuesto por un double, el cual guarda el valor resultante de aplicar el modelo estadístico para el partido tres.

El programa 5.1 muestra la codificación de la clase DatosModelos.java.

```
1  import java.io.Serializable;
2  /**
3   *
4   * @author roberto
5   */
6  public class DatosModelos implements Serializable {
7
8      private String nombreModelo;
9      private double ResultadoPartidoUno;
10     private double ResultadoPartidoDos;
11     private double ResultadoPartidoTres;
12
13     /**
14      * @return the nombreModelo
15      */
16     public String getNombreModelo() {
17         return nombreModelo;
18     }
19
20     /**
21      * @param nombreModelo the nombreModelo to set
```

```
22          */
23         public void setNombreModelo(String nombreModelo) {
24             this.nombreModelo = nombreModelo;
25         }
26 
27         /**
28          * @return the ResultadoPartidoUno
29          */
30         public double getResultadoPartidoUno() {
31             return ResultadoPartidoUno;
32         }
33 
34         /**
35          * @param ResultadoPartidoUno the ResultadoPartidoUno to set
36          */
37         public void setResultadoPartidoUno(double ResultadoPartidoUno) {
38             this.ResultadoPartidoUno = ResultadoPartidoUno;
39         }
40 
41         /**
42          * @return the ResultadoPartidoDos
43          */
44         public double getResultadoPartidoDos() {
45             return ResultadoPartidoDos;
46         }
47 
48         /**
49          * @param ResultadoPartidoDos the ResultadoPartidoDos to set
50          */
51         public void setResultadoPartidoDos(double ResultadoPartidoDos) {
52             this.ResultadoPartidoDos = ResultadoPartidoDos;
53         }
54 
55         /**
56          * @return the ResultadoPartidoTres
57          */
58         public double getResultadoPartidoTres() {
59             return ResultadoPartidoTres;
60         }
61 
62         /**
63          * @param ResultadoPartidoTres the ResultadoPartidoTres to set
64          */
65         public void setResultadoPartidoTres(double ResultadoPartidoTres) {
66             this.ResultadoPartidoTres = ResultadoPartidoTres;
67         }
68 }
```

Programa 5.1: Codificación de la Clase DatosModelos

El archivo *EjecucionModelos.java* realiza las operaciones de petición a la base de datos. La clase contiene dos métodos que son Precarga() y Modelos(). El método Precarga() ejecuta el método EjecucionModelos() automáticamente, al iniciar el programa. Con esta instrucción ya no es necesario realizar la petición desde el navegador y automáticamente ejecuta la llamada a la base de datos para la petición de resultados.

El método Modelos() abre una conexión a la base de datos y hace la petición

de resultados por medio de una consulta. La consulta regresa los resultados y se almacenan en una estructura de tipo ResultSet. Dicha estructura asemeja una tabla de la base de datos. Después recuperamos los datos para poder ser mostrados en el navegador, y eso se realiza guardando los valores contenidos en nuestro ResultSet a objetos de tipo DatosModelos, estos objetos representan los renglones contenidos en el ResultSet.

El programa 5.2 muestra la codificación de la clase EjecucionModelos.java.

```
1
2       private int estado = 2;
3       private String tablava = "'votos_actuales'";
4       private String partidos = "'{p1,p2,p3}'";
5       private String pesos = "'pesos_politico'";
6       private String votosactuales = "'{p1,p2,p3}'";
7       private String votospasados = "'{p1,p2,p3}'";
8       private String tablavp = "'votos_pasados'";
9
10      @PostConstruct
11      public void precarga() throws SQLException {
12          Modelos();
13      }
14      /**
15       * @return Lista con los datos obtenidos de la base de datos.
16       * @throws SQLException
17       */
18      public void Modelos() throws SQLException {
19          getModelosList().clear();
20          try (Connection con = ds.getConnection();
21               PreparedStatement querymodelos = con.prepareStatement("SELECT *
                      FROM modelos(" + tablava + "," + pesos + "," + tablavp + ",
                     " + estado +  "," + votosactuales + "," + partidos + "," +
                     votospasados + ")");
22               ResultSet rs1 = querymodelos.executeQuery()){
23
24              while (rs1.next()) {
25                  DatosModelos modelos = new DatosModelos();
26                  modelos.setNombreModelo(rs1.getString("modelo"));
27                  modelos.setResultadoPartidoUno(rs1.getDouble("res1"));
28                  modelos.setResultadoPartidoDos(rs1.getDouble("res2"));
29                  modelos.setResultadoPartidoTres(rs1.getDouble("res3"));
30                  getModelosList().add(modelos);
31              }
32          } catch (SQLException e) {
33              e.printStackTrace();
34          }
35      }
36      .
37      .
38      .
39  }
```

Programa 5.2: Codificación de la Clase EjecucionModelos

Para mostrar los datos se usa el framework Primefaces, que contiene componentes que facilitan la visualización de los datos. Los datos se presentan en un componente llamado datatable, el cual muestra los datos en un tabla.

El programa 5.3 muestra la codificación del XHTML para la visualización de datos.

```
<p:dataTable id="tablaModelos" var="datos" value="#{ModelosEjec.
    modelosList}">
<f:facet name="header">
   Baja California
</f:facet>
  <p:columnGroup type="header">
   <p:row>
    <p:column/>
       <p:column>
         <f:facet name="header">
           <h:graphicImage width="35" height="35" library="images" name="
               pan.jpg"/>
             </f:facet>
               </p:column>
                 <p:column>
                   <f:facet name="header">
                     <h:graphicImage width="35" height="35" library="
                         images" name="pri.jpg"/>
                   </f:facet>
                  </p:column>
                <p:column>
                 <f:facet name="header">
                  <h:graphicImage width="35" height="35" library="images"
                      name="mc.jpg"/>
                 </f:facet>
                </p:column>
                  </p:row>
                     </p:columnGroup>
                       <p:column style="font-weight: bold;">
                           <h:outputText value="#{datos.nombreModelo}" />
                       </p:column>
                       <p:column>
                           <h:outputText value="#{datos.resultadoPartidoUno
                               }" />
                       </p:column>
                       <p:column>
                           <h:outputText value="#{datos.resultadoPartidoDos
                               }" />
                       </p:column>
                       <p:column>
                           <h:outputText value="#{datos.
                               resultadoPartidoTres}" />
                       </p:column>
                    </p:dataTable>
```

Programa 5.3: Codificación de XHTML para la visualización de Datos

En la figura 5.2 se observan los resultados otorgados por la base de datos y mostrados a través del programa objetivo.

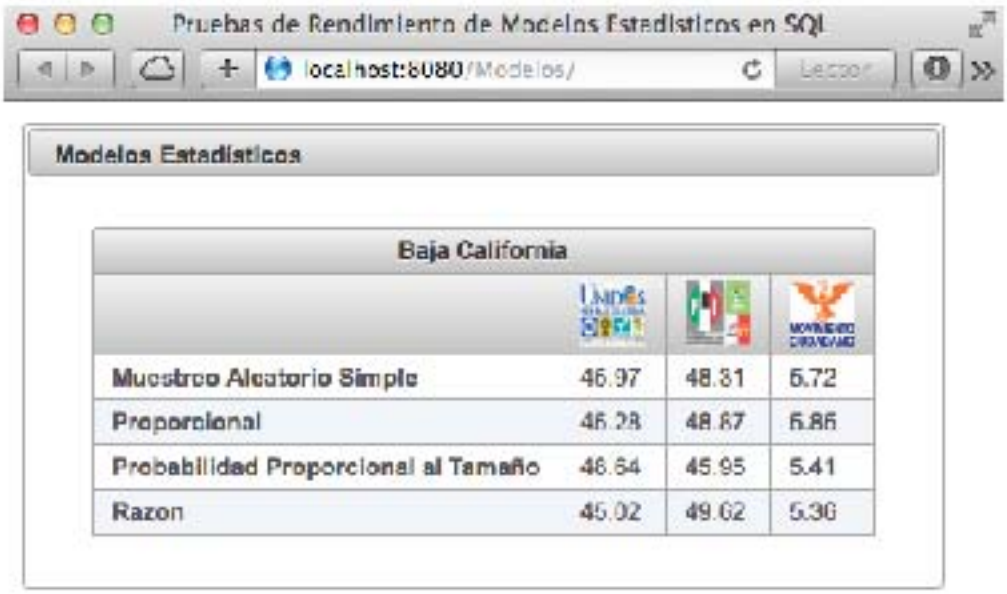

Figura 5.2: Resultados mostrados por el programa objetivo

5.3. Medición de tiempos para la obtención de resultados

El enfoque principal de este trabajo, es poder obtener resultados en un tiempo determinado al ejecutar los métodos y estimadores estadísticos en la base de datos. A continuación se muestran los tiempos obtenidos al realizar los experimentos con grandes cantidades de registros en la base de datos y se comparan con un sistema realizado en PHP. Se debe de tomar en cuenta que los métodos y estimadores se ejecutan simultáneamente y son independientes entre sí, es decir, se están ejecutando cuatro procesos (que son cada uno de los métodos y estimadores) al mismo tiempo.

En el cuadro 5.1 observamos el tiempo que la base de datos tarda en procesar los dos métodos y los dos estimadores estadísticos con un número diferente de datos, cada vez.

Número de Registros	**Tiempo total de procesamiento en SQL (ms)**	**Tiempo total de procesamiento en PHP(ms)**
500,000 Datos	≈2255.291 ms	≈60000.151 ms
1,000,000 Datos	≈4555.392 ms	≈144000.823 ms
1,500,000 Datos	≈6947.159 ms	≈302200.329 ms
2,000,000 Datos	≈9352.030 ms	≈424310.961 ms
3,000,000 Datos	≈14625.838 ms	≈610200.123 ms
4,000,000 Datos	≈18967.399 ms	≈810300.191 ms
5,000,000 Datos	≈23376.482 ms	≈972010.221 ms
6,000,000 Datos	≈28559.136 ms	≈1122090.181 ms
7,000,000 Datos	≈33533.023 ms	≈1218040.951 ms
8,000,000 Datos	≈38706.522 ms	≈1398010.941 ms
9,000,000 Datos	≈43445.529 ms	≈1459003.291 ms
10,000,000 Datos	≈49384.288 ms	≈1608030.129 ms

Cuadro 5.1: Comparativa de tiempos de procesamiento

En el cuadro 5.1 se puede observar que la diferencia de tiempos entre el sistema que realiza los cálculos en PHP y los de SQL es notoria, se tiene que tomar en cuenta que al realizar los cálculos en PHP hay que obtener los datos de la base de datos y operarlos fuera de ella, conforme la tabla de base de datos aumenta en registros el comportamiento del sistema en PHP decrece en rendimiento, esto se debe a que hay que obtener todos los datos pasados como los actuales por cada método y estimador estadístico; a diferencia de que se programen en SQL el manejador de base de datos se encarga de la petición de los datos acelerando el manejo y a su vez las operaciones para el cálculo de los métodos y estimadores.

5.4. Resumen

El programa objetivo implementa una arquitectura web, el cual funge simplemente como un intermediario entre el usuario y la base de datos para facilitar la petición de resultados. Usa como lenguaje principal de programación Java y frameworks como JSF 2 y primefaces. Utiliza como servidor Web a Glassfish y como IDE de desarrollo Netbeans, dichas herramientas incluyen todo lo necesario para implementar el programa objetivo.

Los resultados de tiempo para el procesamiento de datos (mostrados en el cuadro 5.1), muestran el tiempo que tardan en ejecutarse los métodos y estimadores estadísticos en el sistema manejador de base de datos PostgreSQL. La ventaja que obtenemos al procesar los datos de esta forma es que el manejador de base de datos estructura la información, de tal manera que acelera los procesos para poder otorgar la información rápidamente, así no tenemos que preocuparnos de tratar una gran cantidad de datos fuera del manejador.

Capítulo 6

Conclusiones

En este capítulo se analizan los resultados mostrados en el capítulo anterior, de donde se obtienen las conclusiones finales para este trabajo.

6.1. Resumen

La característica principal del presente trabajo se basa en aprovechar las ventajas que proporciona un sistema manejador de base de datos, que son el almacenamiento y manipulación de grandes cantidades de datos, para obtener resultados en tiempos delimitados. Al procesar los métodos y estimadores estadísticos, se observa que su rendimiento es mayor al procesarlos dentro del sistema manejador, dada la flexibilidad y el manejo que nos otorga sobre los datos.
Sin embargo, podemos encontrar algunos problemas que son:

- El rendimiento de la base de datos está sujeta en bastante medida al hardware con el que se trabaja. Esto se debe a que el sistema manejador se acopla al hardware del cual dispone la computadora en donde se instala. Dicho esto, los tiempos de respuesta pueden verse afectados significativamente si se emplea este método en computadoras con bajos recursos.

- Saturación de conexiones a la base de datos. El aumento en las conexiones que se tengan a la base de datos pueden provocar la disminución en el tiempo de respuesta, ya que la base de datos recibe y atiende las peticiones requeridas por los usuarios, aumentando el tiempo para el término del proceso.

El programa objetivo se utiliza para poder realizar la petición de resultados automáticamente, ocultando la complejidad de realizar consultas a la base

de datos para obtener resultados. Cada modelo se ejecuta simultáneamente dentro del sistema manejador, teniendo un rendimiento óptimo al procesar grandes cantidades de datos. El trabajo implementado se diseña únicamente para la ejecución de métodos y estimadores estadísticos. Dichos métodos y estimadores fueron adaptados para el cálculo de resultados electorales.

6.2. Contribuciones

Las contribuciones que se obtienen de este trabajo son:

- Desarrollo de métodos y estimadores estadísticos aplicados para el cálculo de resultados electorales.
- Implementación de métodos y estimadores estadísticos dentro de un manejador de base de datos, utilizando Structured Query Language.
- Implementación de programa objetivo para la visualización de resultados, aplicando patrones de diseño.
- Se muestra que es mejor el tratamiento de grandes cantidades de datos dentro de un sistema manejador, para obtener resultados en tiempos delimitados.

6.3. Trabajo Futuro

Los trabajos que pueden desprenderse son:

- Utilizar los métodos y estimadores estadísticos en ámbitos más generales. Los métodos y estimadores estadísticos son generados a partir de la necesidad del cálculo de resultados electorales. En este caso, se pueden modificar para algún problema en específico que se requiera resolver.
- Comparación de tiempos de ejecución con otro manejador de base de datos. Existen distintos manejadores, los cuales pueden trabajar de manera distinta y realizar el proceso con mayor eficiencia. La finalidad de este punto es realizar comparación de tiempos entre distintos manejadores, pues el enfoque de este trabajo es que la ejecución de los modelos sea veloz. Las principales sugerencias serían implementar los modelos en Oracle o Microsoft SQL Server ambos manejadores son considerados de uso empresarial, y al ser software de paga se tienen ventajas, como un mejor manejo de memoria y de recursos.

- Comparación de la eficiencia de los métodos y estimadores, con una computadora de mayores recursos. Para comparar dichos resultados con los obtenidos en este trabajo y determinar sí las conclusiones son iguales.

Bibliografía

[1] Instituto federal electoral-glosario de términos.

[2] Ed Burns, Chris Schalk, and Neil Griffin. *The Complete Reference: Java Server Faces 2.0*. McGraw-Hill, 2010.

[3] V. Carot. *Control Estadístico de la Calidad*. Servicio de Publicaciones Camino de Vera, 1998.

[4] Carlos Coronel, Steve Morris, and Peter Rob. *Database Systems: Design, Implementation and Management*. Cengage Learning;, 9 edition, November 23, 2009.

[5] Amparo López Gaona. Bases de datos 1 introducción.

[6] Maurice Houstma and Arun Swami. Set-oriented mining for association rules in relational databases. 1995.

[7] Leslie Kish. *Muestreo de Encuestas*. Editorial Trillas, 1979.

[8] Kevin Kline. *SQL in a Nutshell*. O Reilly, 3 edition, 2008.

[9] César Pérez López. *Muestreo Estadístico, Conceptos y Problemas Resueltos.* Pearson, 2005.

[10] Shamkant B. Navathe Ramez Elmasri. *Fundamentals of Database Systems*. Addison-Wesley Pearson, 6 edition, 2010.

[11] Abraham Silberschatz, Henry F Korth, and S Sudarshan. *Database system concepts*. McGraw-Hill Hightstown, 6 edition, 2010.

[12] David W. Stockburger. Introductory statistics: Concepts, models and applications.

[13] Oleg Varaksin and Mert Çasliskan. *PrimeFaces Cookbook*. Packtpub, 2013.

Apéndice A

Instalación y Configuración

La realización de los experimentos en este trabajo requieren de la instalación y configuración de herramientas, para ayudarnos a realizar la ejecución de los métodos y estimadores estadísticos de manera satisfactoria. La parte esencial es la instalación y configuración del manejador de base de datos PostgreSQL, la cual requiere de cambios en los parámetros de configuración por omisión, para obtener un mejor rendimiento y aprovechar al máximo los recursos computacionales disponibles.

En esta sección se explican a detalle las configuraciones necesarias de las diferentes herramientas que usaremos para la experimentación de este trabajo.

A.0.1. Instalación y configuración de PostgreSQL

La instalación de PostgreSQL está incluida en un gestor de paquetes llamado **Homebrew**; dicho paquete realiza la instalación de forma automática en OS X, a continuación explicamos la instalación de Homebrew y de la base de datos para las pruebas de experimentación.

La instalación de Homebrew necesita de la versión más reciente que exista de Xcode[1] y se instala de la siguiente manera:

```
1 robert$ xcode-select --install
```

Programa A.1: Comando de instalación Xcode

Una vez terminada la instalación de las herramientas de desarrollo de Xcode proseguimos a instalar Homebrew con el siguiente comando:

[1]Xcode son las herramientas de desarrollo de software para sistemas operativos OS X.

```
robert$ ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go/
    install)"
```

Programa A.2: Comando de instalación de Hombrew

Sólo basta instalar el sistema manejador de base de datos PostgreSQL con los siguientes comandos:

```
robert$ brew update
robert$ brew install postgresql
robert$ initdb /usr/local/var/postgres
robert$ cp /usr/local/Cellar/postgresql/9.3/homebrew.mxcl.postgresql.plist
    ~/Library/LaunchAgents/
robert$ launchctl load -w ~/Library/LaunchAgents/homebrew.mxcl.postgresql.
    plist
robert$ pg_ctl -D /usr/local/var/postgres -l /usr/local/var/postgres/server.
    log start
robert$ createdb robert
```

Programa A.3: Comandos de Instalación PostgreSQL

Para ver más detalles acerca de la instalación anterior toda la documentación se encuentra el sitio oficial de Homebrew[2].

[2] http://brew.sh/index_es.html

En la figura A.1 se puede observar la ejecución de la base de datos PostgreSQL.

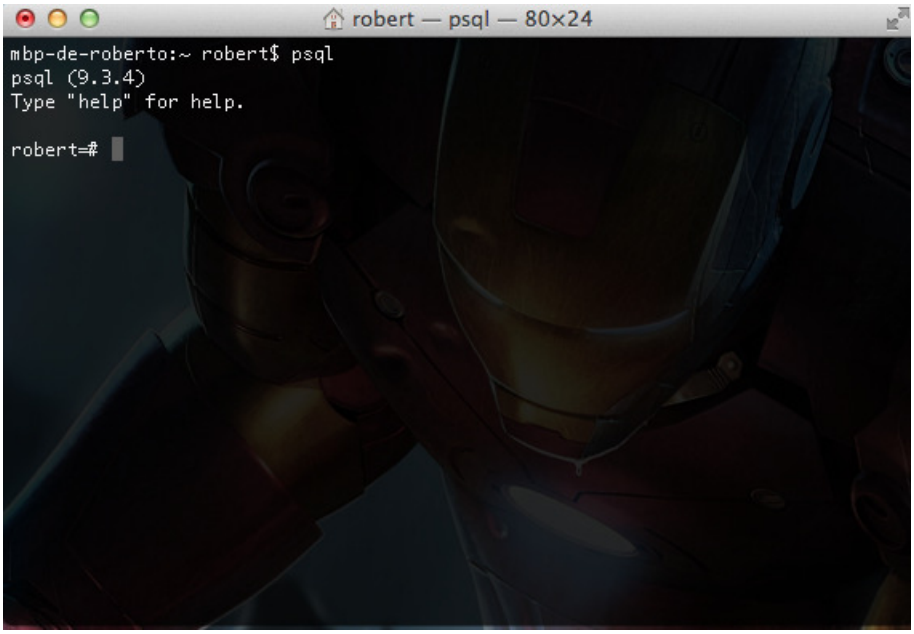

Figura A.1: Ejecución de Base de Datos PostgreSQL 9.3.4

Teniendo instalado y ejecutando el sistema manejador PostgreSQL, creamos una base de datos, para generar las tablas que contienen la información y poder ejecutar los métodos y estimadores estadísticos. Utilizamos el siguiente comando para realizar dicha tarea:

```
mbp-de-roberto:~ robert$ psql
psql (9.3.4)
Type "help" for help.

robert=> CREATE DATABASE modelos;
CREATE DATABASE
```

Programa A.4: Comando para crear base de datos.

Para verificar que la base de datos se creó satisfactoriamente sólo tecleamos el siguiente comando:

```
robert=> \l
                              List of databases
   Name    | Owner  | Encoding | Collate | Ctype | Access privileges
-----------+--------+----------+---------+-------+-------------------
 modelos   | robert | UTF8     | C       | UTF-8 |
 robert    | robert | UTF8     | C       | UTF-8 |

(2 rows)
```

Programa A.5: Listado de base de datos

Como se puede observar en la salida del Programa A.5 la base modelos está contenida en el manejador de base de datos y lista para ser manipulada.

Las configuraciones del manejador de base de datos están definidas en el archivo **postgresql.conf**, dicho archivo se encuentra en la ruta **/usr/local/var/postgres/postgresql.conf**.
En el archivo de configuraciones se modifican algunos parámetros para tener una mejora tanto en rendimiento y manejo de conexiones a la base.

Las modificaciones a la configuración[3] de PostgreSQL son las siguientes:

Parámetro	Valor por omisión	Valor Nuevo	Utilidad
port	#	5432	Puerto para realizar enlaces a la base de datos.
max connections	100	15	Reducir las conexiones para mejorar el rendimiento.
shared buffers	128 MB	1 GB	Memoria dedicada al servidor de base de datos.
work mem	1 MB	16 MB	Memoria usada para mejorar las operaciones como ordenamientos, distinciones y uniones en las consultas.

Las configuraciones de la base de datos que se tienen por omisión son para computadoras con pocos recursos o los mínimos para obtener un buen funcionamiento. El cambio en los parámetros anteriores se realizan para poder obtener el mejor rendimiento posible, adecuado a las capacidades de la computadora utilizada para la experimentación.

[3]Todos los parámetros de configuración los podemos encontrar en `www.postgresql.org/docs/9.3/static/runtime-config.html`

A.0.2. Instalación de Java

Para este trabajo utilizamos la versión de Java 1.7, primeramente obtenemos el JDK[4] que se encuentra en la siguiente dirección: `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html?ssSourceSiteId=otnes.`
Aceptamos el contrato y descargamos la versión OS X de 64 bits.

Para su instalación damos doble click sobre el archivo .dmg descargado, al terminar la instalación abrimos la terminal y tecleamos

```
robert$ java -version
```

y debe de aparecer la versión de java que acabamos de instalar.

En la figura A.2 y A.3 observamos la instalación finalizada correctamente.

Figura A.2: Instalación de Java

[4]Su significado es Java Development Kit su traducción al español es Herramientas de Desarrollo Java

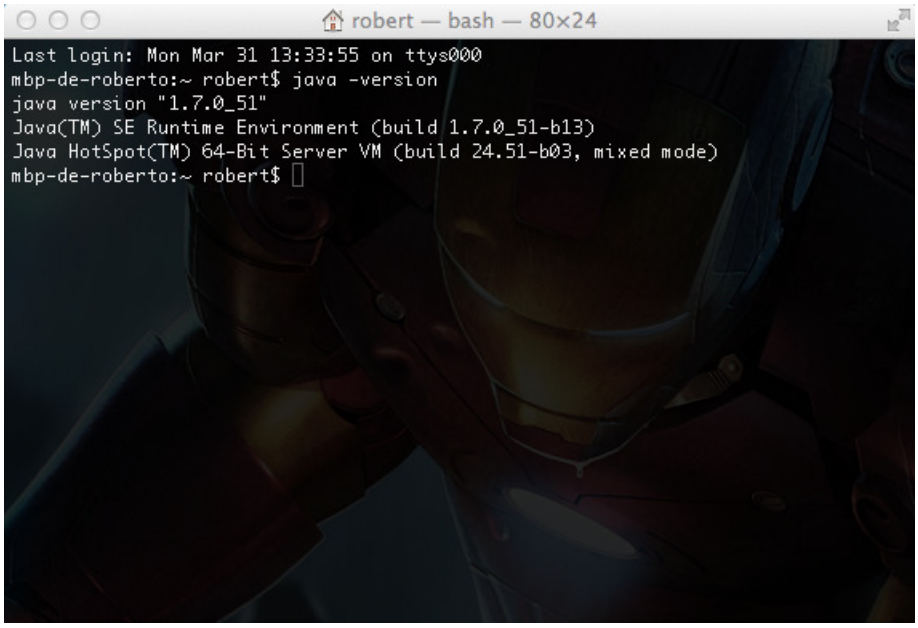

Figura A.3: Verificación de Instalación

A.0.3. Instalación de Netbeans y Servidor Web Glassfish

Netbeans es el IDE utilizado para escribir el código del programa objetivo, este programa trae incluido todo lo necesario para el desarrollo e implementación de nuestro programa.
Para la instalación se debe descargar el paquete de Java EE, que se ubica en la página: `https://netbeans.org/downloads/7.3/` para plataformas OS X.

Para la instalación basta con hacer doble click sobre el archivo .dmg, aceptar el contrato y proseguir con los pasos que nos va dando el instalador, en la figura A.4 se muestra el proceso de instalación de Netbeans.

Al terminar la instalación abrimos el programa de Netbeans, y creamos un nuevo proyecto, debemos elegir la opción Java Web. Para crear los archivos y directorios necesarios, debemos proporcionar los requerimientos del proyecto que son:

- Nombre del proyecto. Para este trabajo lo denominaremos "Modelos".
- Servidor. Se elige el servidor por omisión que es Glassfish Server 3.1.2.

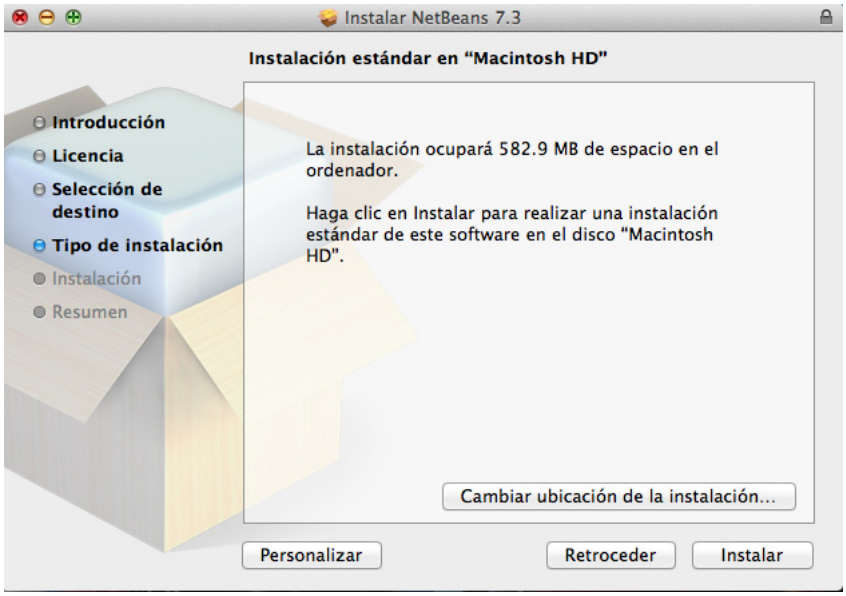

Figura A.4: Proceso de instalación de Netbeans

- Java EE versión. Se elige Java EE 6 Web.
- Frameworks. Se selecciona el apartado de JavaServer Faces.

Netbeans tiene instalado Glassfish como servidor web por omisión, por lo tanto no se requiere ninguna instalación extra para este componente.

A.0.4. Configuración de conexiones a la base de datos en Glassfish

Algo que afecta el rendimiento de la base de datos al realizar consultas son la apertura y cierre de conexiones. Esto se debe a que cada vez que se pide una consulta se tiene que abrir una conexión entre el programa objetivo y la base para intercambiar datos, una vez que la base de datos termina de enviar los resultados de la consulta se debe asegurar que esas conexiones sean cerradas, esto debido a que si no se cierran, la base de datos tendrá abiertas conexiones que no pueden ser reutilizadas.

Para este tipo de casos en una aplicación web podemos crear un pool de conexiones por medio del servidor, este denominado pool lo que hace es tener abiertas conexiones permanentemente a la base de datos pudiendo ser reutilizadas, evitando así la apertura y cierre de conexiones constantemente.

La configuración del pool de conexiones en glassfish se hace de la siguiente manera.
Iniciamos el servidor web desde Netbeans, abrimos un navegador web y tecleamos la siguiente dirección: `http://localhost:4848/common/index.jsf` está dirección nos conecta a las configuraciones del servidor web. Accedemos al menú de Recursos y seleccionamos el apartado JDBC, dentro de ese apartado encontramos dos urls que son Recursos JDBC y Pool de Conexiones JDBC. El primer paso a seguir es entrar a la url Pools de Conexiones JDBC, y creamos un nuevo pool en la figura A.5 y A.6 se muestran los campos requeridos para crear un pool de conexiones.
La configuración principal está dada por los siguientes campos:

- Tamaño de Pool Inicial y Mínimo. En este campo se fija el parámetro para el número de conexiones iniciales, para este trabajo se fija en 6 conexiones.
- Tamaño de Pool Maximo. Fija la cantidad máxima de conexiones que se pueden abrir, se fija en 10 conexiones.
- Cantidad de cambios de tamaño del pool. Fija el número de conexiones que se eliminan por inactividad, se fija en 2 conexiones.
- Tiempo de inactividad. Es el tiempo máximo que puede permanecer inactiva una conexión, se fija en 400 segundos.
- Tiempo de espera. Es el tiempo que espera el emisor de la llamada antes de que se envíe un mensaje de error en la conexión, se fija en 60000 milisegundos.

Con esta configuración nuestro pool de conexiones está listo para poder atender las peticiones que se le hagan, ahora sólo falta crear un recurso JDBC que es con el cual va a ser llamado el pool de conexiones desde nuestro código fuente, el cual simplemente nos pide un nombre e indicar al pool de conexiones que apunta en la figura A.7 observamos la configuración del recurso JDBC.

Figura A.5: Configuración de pool de conexiones

Figura A.6: Configuración de parametros de pool de conexiones

Figura A.7: Creación de Recurso JDBC

Apéndice B

Diagrama de Clases

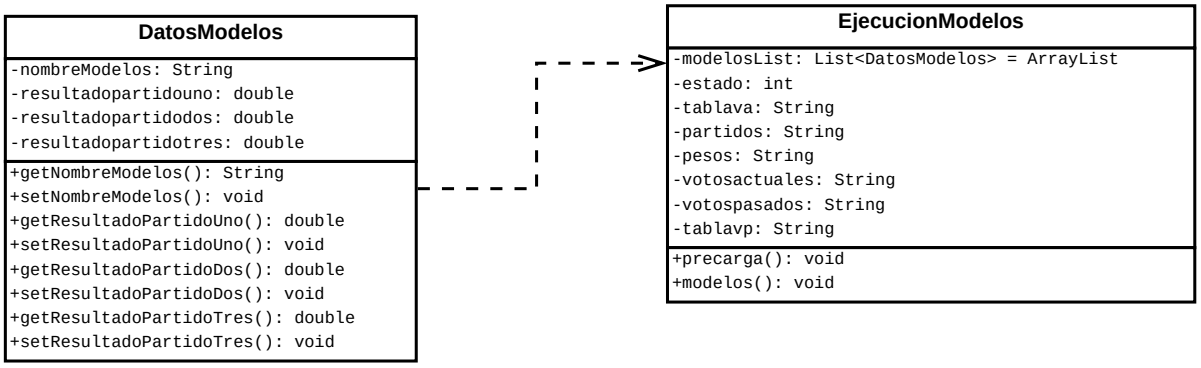

Figura B.1: Diagrama de Clases