CENTRO DE INVESTIGACIONES EN ÓPTICA, A.C.

DESARROLLO DE PLATAFORMA INTEGRAL DE MONITOREO DE PARÁMETROS ELÉCTRICOS POR MEDIO INALÁMBRICO

Como Requisito para obtener el grado de:

Maestro en Optomecatronica

Asesor:
M.I Enrique Noe Arias

Estudiante:
Ing. Fabio Vega Nieto

Mayo de 2017
León, Guanajuato, México

Resumen

Conocer el consumo energético y variables eléctricas de una empresa o bien en el hogar, es de vital importancia cuando se trata de reducir costos por pagos de luz o cuidar la vida de las máquinas y equipos, ya que la información llega fuera de tiempo, el costo de mantenimiento incrementará de manera significativa o podría producirse un daño importante en los equipos pudiendo ser prevenido. Al contar con un medidor de energía trifásico el problema para obtener la información y mantenerse al tanto del consumo de energía podría estar solucionado, pero si no es posible estar todo el tiempo monitoreando estos datos, la información podría ser obsoleta a la hora en que sea posible consultarla, o podría ser demasiado tarde para prevenir alguna falla.

La finalidad del proyecto consiste en integrar una plataforma de enlace inalámbrico a un dispositivo monitor de redes eléctricas con comunicación Modbus TCP con el objetivo de monitorear los principales parámetros eléctricos, los cuales serán almacenados en una base de datos con fines de analizar consumos, facturación, comportamiento histórico etc. Realizar alarmas de consumo como de demanda que influye directamente en la factura de algunas empresas. Con la información recabada por el Medidor de Energía Trifásica se podrá tener acceso en puntos alejados de la zona en que se encuentre el sistema de medición, lo cual permitirá además de tener la información reciente, consultar gráficas en las que se podrá observar con facilidad si hay algún motivo para considerar corregir el factor de potencia, Factor de carga y demanda de la empresa. Dicho monitoreo estará disponible para consulta desde cualquier dispositivo autorizado para acceder a ella como también para la toma de decisiones administrativas para planeación de producción y carga en el sistema eléctrico que influye directamente en la facturación.

Este proyecto consta de un sistema hardware y software para el monitoreo, guardado y análisis de parámetros eléctricos de una empresa para lo cual se usa un PM850 ó S203TA-D como medidor de parámetros eléctricos trifásicos, un software desarrollado Labview con interfaz ModBus TCP para comunicación con los medidores, un sistema de gestión de usuarios para administración del software, un sistema programable de guardado de datos de los medidores, una base de datos con en la cual se diseñaron las tablas, procedimientos y funciones de almacenado para el cálculo de tarifas, almacenamiento de usuarios, gestión de días festivos, gestión de errores, configuración del software, alertas etc. Un sistema de encriptado de datos para darle seguridad al software y a los usuarios del mismo, un sistema de

configuración de tarifas, un sistema de consulta de tarifas y generación de reportes, un sistema de configurable de alertas, y un sistema automático de actualización de tarifas de CFE.

El modelo de programación implementado en Labview es un sistema productor consumidor en configuración "continuous message data and logging" que permite al software administrar en diferentes hilos de programas (programas con ejecución en paralelo y comunicación entre ellos), la gestión de cada uno de sistemas mencionados en el párrafo anterior. El sistema se encuentra terminado e implementado en Industrias Scalini la cual también comercializa este servicio.

Palabras Claves: Monitoreo de parámetros eléctricos, Modbus RTU/TCP, Calidad de energía, Base de Datos.

Dedicatoria

A Mis Padres

Luz Marina Nieto Atencio y Fabio Vega Díaz, Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A Mis Hijos

Quienes me han enseñado el amor de dar en vez de recibir y me han impulsado a ser cada vez mejor. A mi esposa Por enseñarme a superar las dificultades.

A Mis Amigos.

Que nos apoyamos mutuamente en nuestra formación profesional y que hasta ahora, seguimos siendo amigos: Daniel Moreno, Fernanda Gonzales, Jorge Aguirre, Amanda salas, Ramon Cruz, Iosvani More.

A Mi Familia

Por haberme apoyado en todo momento, especialmente a mis hermanas Sandy y Mariangelica.

Agradecimiento

A todo el pueblo mexicano y al CONACYT por el financiamiento económico de mis estudios acá en México. Al Centro de Investigaciones en Óptica, por brindar esta maestría e instalaciones para el desarrollo de la misma.

A mi director de Tesis, Enrique Noe Arias por permitirme aportar a este proyecto con la industria, por mostrarme un ejemplo de humildad, colaboración y sus excelentes clases de Labview.

A Ricardo Valdivia por sus amables concejos, historias y apoyos en el laboratorio de electrónica del CIO.

Al doctor David Moreno por ver una cátedra de matemáticas como debe ser.

Al doctor Manuel De la Torres por sus clases de Matlab y apoyo cuando lo necesite.

Al Doctor Geminiano Martinez por su humildad y apoyo con clases extras los sábados cuando las necesitábamos.

A Mis Amigos Yaili Fernadez, Iosvani Moreno por apoyarme cuando estaba enfermo.

A Mis Amigos Fernada Gonzales y Jorge Aguirre por brindarnos estas maravillosas vacaciones, donde conocimos muchos bellos lugares con mi familia en este bello país.

A Daniel Sanpedro, quien me brindo su amistad incondicional.

A doña Angeles por brindar su hogar para algunas de nuestras reuniones.

A Annete torres y Claudia Medina por ese bello congreso en Guanajuato.

A los Colombianos compañeros en el CIO quienes me recibieron y nos apoyamos mutuamente.
A mis amigos Colombianos que con sus palabras me alentaron siempre a continuar.
A todos los Mexicanos que de alguna u otra manera me apoyaron y recibieron en sus hogares.

Contenido

Lista de Figuras

Lista de Tablas

1. Antecedentes y objetivos

1.1. Introducción

En este primer capítulo se describen algunos antecedentes para el desarrollo del proyecto como son tesis para el desarrollo del equipo de medición de parámetros eléctricos, uso de equipos para medición del mismo, sistemas SCADA para disminución del valor del consumo, empresas que brindan el servicio de monitoreo y control de parámetros eléctricos como también los principales fabricantes de equipos de medición. También se describen los objetivos del proyecto como los resultados esperados.

1.2. Antecedentes

Actualmente se han desarrollado diversas investigaciones y trabajos en el que se han propuesto una variedad de técnicas para medición y monitoreo de parámetros eléctricos como técnicas para disminuir la factura a través del control de la demanda máxima consumida por una empresa. En la tesis [1] utilizo para su trabajo un medidor de parámetros eléctricos Alpha power plus combinado con un sistema telemétrico GSM/GPRS por TCP a través de la red móvil. Para que la empresa donde se realizó el trabajo pudiera monitorear remotamente los parámetros de su interés (Voltaje, Corriente, Potencia, Demanda etc.). También se han desarrollado trabajo en el desarrollo del medidor mismo como [2,3], donde se usa como circuito principal de medición la serie (ADE7753, ADE7758) de "Analog Devices" los cuales pueden medir Voltaje, corriente, potencia activa, reactiva, aparente, factor de potencia, demanda por fase y global con normas internacionales de medición IEC 60687, IEC 61036, IEC 61268, IEC 62053-21, IEC 62053-22 y IEC 62053-23. Existen varios esfuerzos realizados a nivel nacional e internacional para atacar el problema de la eficiencia en el consumo energético, existe un trabajo muy interesante realizado en el Instituto Politécnico Nacional [4], donde se utiliza un sistema SCADA para monitorear el consumo eléctrico y a través de control de cargas con un PLC logra disminuir costos por conceptos de demanda máxima en los diferentes horarios de CFE desconectando cargas que no contribuyen al proceso de producción inmediato de la planta. Evitando la penalización por una elevada potencia consumida en ciertos instantes de tiempo. Por otro lado existen empresas Nacionales y extranjeras que brindan el servicios de estudio de parámetros eléctricos de manera local como en la web como lo son el grupo absa(www.grupoabsa.com), Suministros y Servicios Electromecánicos

SA de CV(www.syse.com.mx), codensa en Colombia (www.codensa.com.co) que brindan servicio y solución de Factor de potencia, Contenido de armónicos, Disturbios eléctricos, Redes de tierra, Fuentes de energía alternativa, Calidad de energía, Ahorro de energía, eficiencia energética etc. Por ultimo cabe destacar el desarrollo de equipos robustos para la medición de parámetros eléctricos con diferentes protocolos de comunicación entre ellos Modbus RTU y TCP, PROFINET, PROFIBUS etc. como son los de National Instruments, Schneider Electric, SIEMENS, SENECA, Allen Bradley, Phoenix, etc.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar e implementar una plataforma que sea capaz de enlazarse a un equipo medidor de variables eléctricas con comunicación Modbus RTU/TCP para la concentración de datos correspondientes a los parámetros eléctricos, para arrojar datos comparativos con la facturación de consumos eléctricos proporcionados por CFE, históricos de consumo, con la finalidad de fomentar la cultura de ahorro energético, planeación de mantenimiento y estrategias de consumo. Esta plataforma de Hardware-Software una vez desarrollada será comercializada por Industrias Scalini S.A de C.V como empresa integradora.

1.3.2. Objetivos Específicos

- Diseñar la base de datos que soporte las tarifas actuales de CFE.
- Diseñar Procedimientos de almacenado en la base de datos para cálculo de las diferentes tarifas.
- Realizar sentencias SQL para correcta comunicación entre LabVIEW y MySQL.
- Diseñar rutinas en LabVIEW para consulta de tarifas
- Diseñar rutina el LabVIEW para actualización automática de tarifas.

1.4. Resultados Esperados Del Proyecto

Desarrollo e implementación una plataforma de software que sea capaz de enlazarse a un dispositivo medidor de variables eléctricas y obtener los valores de potencia Activa, Reactiva, Aparente, Demanda, Factor de potencia, voltajes, corrientes, etc., mediante una comunicación Modbus RTU/TCP, almacene estos datos de las mediciones en una base de datos MySQL, el software deberá ser capaz de arrojar datos comparativos con la facturación de consumos eléctricos proporcionados por CFE, históricos de consumo, con la finalidad de

fomentar la cultura de ahorro energético, planeación de mantenimiento y estrategias de consumo. Esta plataforma de Hardware-Software una vez desarrollada será comercializada por Industrias SCALINI S.A. de C.V.

2. Conceptos Básicos

2.1. Parámetros eléctricos

2.1.1. Voltaje

Es una magnitud física, es la diferencia de potencial eléctrico entre dos puntos; en un circuito eléctrico impulsa los electrones a través del mismo. Su unidad de medida es el Voltio y su símbolo eléctrico V en honor a Alessandro Volta quien invento la pila eléctrica. El voltaje puede ser de corriente directa DC o de corriente alterna AC.

2.1.2. Corriente Eléctrica

Es el flujo de carga o electrones a través de un circuito eléctrico impulsado por un voltaje. En el Sistema Internacional de Unidades se expresa en C/s (culombios sobre segundo), unidad que se denomina amperio ó ampere en honor al físico francés André-Marie Ampere (1775-1836). Su símbolo eléctrico A.

2.1.3. Potencia Eléctrica

La potencia eléctrica es la relación de paso de energía de un flujo por unidad de tiempo; se calcula multiplicando el voltaje por la corriente. La unidad en el Sistema Internacional de Unidades es el vatio (watt).

2.1.4. Potencia Activa (P)

La potencia activa es la que se aprovecha como potencia útil en los dispositivos resistivos. También se llama potencia media, real o verdadera y se debe a los dispositivos resistivos. Su unidad de medida en el vatio (W).

Se calcula como:

$$P = V * I * cos(\varphi) \tag{2-1}$$

Donde φ, es el desfase entre el voltaje y la corriente alterna producidos por la presencia de bobinas y capacitores en el circuito eléctrico.

2.1.5. Potencia Reactiva (Q)

La potencia reactiva es la potencia compleja que necesitan las bobinas y los condensadores para generar campos magnéticos o eléctricos, pero que no se transforma en trabajo efectivo, sino que fluctúa por la red entre el generador y los receptores. Su unidad de medida es el voltamperio reactivo (VAR).

Se calcula como:

$$P = V * I * sen(\varphi) \tag{2-2}$$

2.1.6. Potencia Aparente (S)

Es la potencia total consumida por la carga y es el producto de los valores eficaces de tensión e intensidad. Se obtiene como la suma vectorial de las potencias activa y reactiva y representa la ocupación total de las instalaciones debidas a la conexión del receptor. Su unidad de medida es el voltamperio (VA).

Se calcula como:

$$S = \sqrt{P^2 + Q^2} \tag{2-3}$$

$$S = \tfrac{1}{2} V * I = V_R ms * I_R ms \tag{2-4}$$

Donde $V_R ms$,$I_R ms$ son los valores eficaces o RMS del voltaje y corriente. Estas tres potencias se relacionan a través del conocido triangulo de potencia compleja, como se observa a continuación.

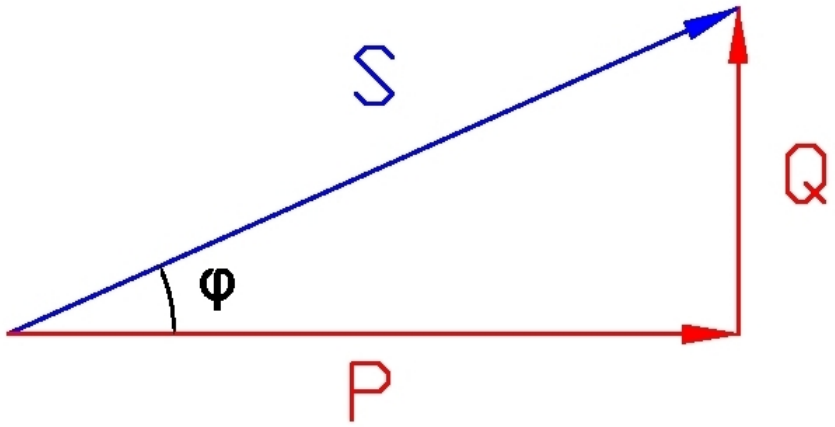


Figura 2-1.: Triangulo de potencia Compleja

2.1.7. Factor de Potencia (φ)

El factor de potencia φ, es el desfase entre el voltaje y la corriente alterna producidos por la presencia de bobinas y capacitores en el circuito eléctrico. También se puede calcular como el cociente entre la potencia reactiva Q y el valor potencia aparente Activa P. Un valor entre 0 y 1. Ver figura 1. Comisión Federal de Electricidad penaliza a las industrias por un bajo factor de potencia inferior a 0.9. Para ver el efecto de un bajo factor de potencia, supongamos dos cargas con igual consumo P=20000W, pero una con factor de potencia de 0.98 y la otra de 0.7. Con un voltaje de alimentación de 220v Por lo que la corriente en potencia activa se puede calcular en y la potencia aparente:

$$Factor(1) \mapsto I = \frac{20000W}{(220V * 0{,}98)} = 92{,}76A \tag{2-5}$$

$$Factor(0{,}7) \mapsto I = \frac{20000W}{(220V * 0{,}7)} = 129{,}8A \tag{2-6}$$

$$Factor(1) \mapsto S = 220V * 92{,}76A = 20408{,}16VA \tag{2-7}$$

$$Factor(0{,}7) \mapsto S = 220V * 129{,}8A = 28571{,}42VA \tag{2-8}$$

Como se puede observar la corriente se ha incrementado en un 33 % aproximadamente, lo

que implica el uso de conductores de mayor calibre. También a menor factor de potencia mayor potencia aparente y generadores de mayor capacidad. Por lo que implica mayor costo de operación por CFE, por esto es penalizado un factor de potencia por debajo de 0.95.

2.1.8. Energía Eléctrica

La energía eléctrica en corriente alterna es la potencia eléctrica por unidad de tiempo generalmente en KW (kilowatt/hora) y es el consumo que cobra CFE.

2.1.9. Red trifásica

Un sistema trifásico de distribución de energía y consumo eléctrico está formado por tres corrientes alternas monofásicas de igual frecuencia y amplitud, que tienen una diferencia de fase entre ellas de 120° eléctricos denominadas fase. Como se observa en la Figura **2-2**.

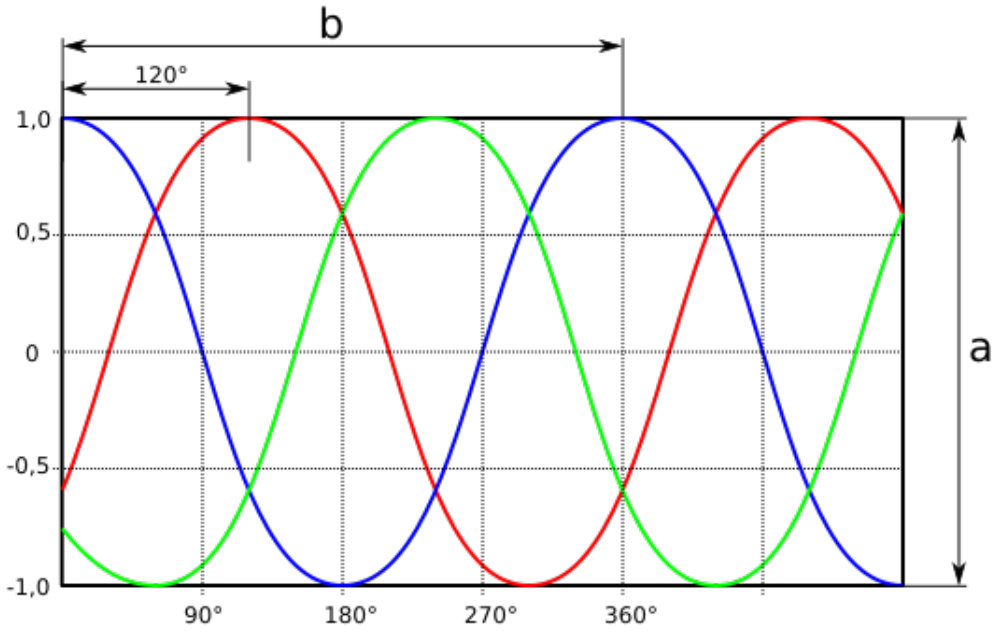


Figura 2-2.: Desfase en red trifásica.

2.1.10. Demanda Máxima

Es una medida del promedio de potencia en KW en un periodo de tiempo generalmente entre 5 y 30 minutos, para el caso de México comisión federal de electricidad tiene como periodo de medida 15 minutos. Como se observa en la siguiente ecuación.

$$DemandaMaxima = \frac{\sum PotenciasMedidas}{NumerodeMuestras} \quad (2\text{-}9)$$

Un ejemplo se puede observar en la figura 3. Donde el medidor registra la demanda de energía en un periodo de 15 minutos del instante A al B, donde a pesar de haber una valor pico de demanda la demanda se calcula tomando el promedio en dicho periodo.

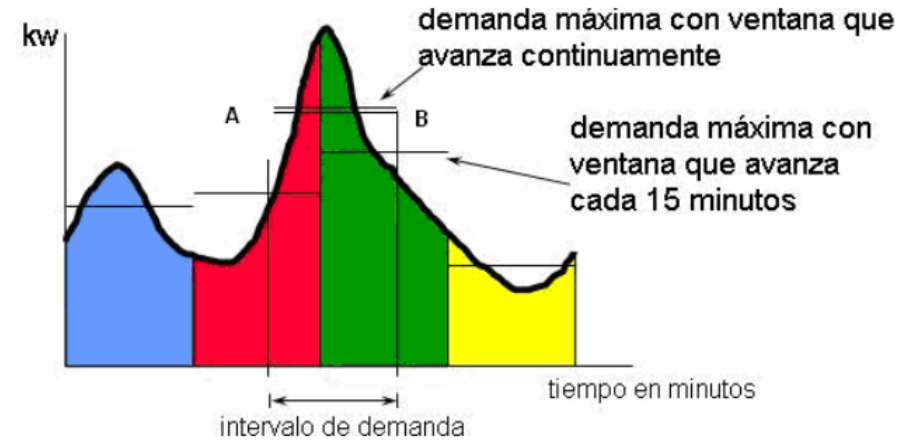


Figura 2-3.: Demanda máxima en un periodo de 15 minutos [4].

Existen diferentes formas de cálculo que implementan los equipos electrónicos para medir la demanda máxima como son:

- **Ventana fija:** en este caso la empresa que suministra la energía proporciona un pulso entre cada periodo de medición para indicar el inicio del periodo de medición de demanda máxima.
- **Ventana deslizante:** en este caso no hay pulso de sincronización pero el equipo de medición calcula la demanda máxima sobre los últimos 15 minutos.
- **Ventana de sincronización de tiempo:** es una variante de la ventana fija en este caso la empresa proporciona un pulso de sincronización al inicio del día y el equipo de medición a través de su reloj interno calcula los otros periodos.

2.1.11. Factor de Carga (FC)

Es la relación que existe entre la energía en KWh y la energía obtenida de multiplicar la demanda máxima por las horas transcurridas en emplear dicha energía. Como se observa en la ecuacion:

$$\frac{EnergiaKWh}{DemandaMaxima*Horastranscurridas} \tag{2-10}$$

Esta media implica que si una empresa tiene equipos que no usa habitualmente pero que consumen mucha energía, cada KWh consumida saldrá mucho más caro por concepto de demanda máxima ya que la demanda máxima durante el periodo de consumo será mayor respecto al promedio de consume en KWh habitual. Si el consumidor utiliza la capacidad total, o sea la demanda máxima, durante las 24 horas diariamente, se dice que está operando al 100 % de su carga o de su factor de carga. En esta forma se logrará la tarifa más baja por kilowatt-hora.

3. Marco Teórico

3.1. Bases de datos

Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Las bases de datos tradicionales se organizan por campos, registros y archivos. Lo cual permite un manejo de grandes volúmenes de información de manera eficiente. La información se organiza en tablas que a su vez tienen columnas don diferentes tipos de datos, cada tuplas de información es almacenado en forma de filas

3.2. Lenguaje SQL

SQL (Structured Query Language) es un lenguaje de programación estándar e interactiva que permite interactuar con la base de datos para la obtención de información de esta, para eliminarla y/o actualizarla. Una de sus características es el manejo del álgebra y el cálculo relacional directamente en el servidor que permiten efectuar consultas con el fin de recuperar información, de forma sencilla. Aunque SQL es a la vez un ANSI y una norma ISO, muchos productos de bases de datos soportan SQL con extensiones propietarias al lenguaje estándar.

3.2.1. Sintaxis de SQL

Comandos para definición de datos en tablas.

- **CREATE TABLE:** Se utiliza para crear una nueva relación a la que se le asigna un nombre y unos atributos
- **DROP TABLE:** Borra una relación existente, así como también sus atributos y la tupla asignada a esta relación.
- **ALTER TABLE:** Modifica la tabla, agrega un atributo a una de estas, además de cambiar la tupla del código de la Base de Datos.
- **CREATE INDEX:** Comando empleado para crear índices, estos índices se crean bajo un nombre y pueden ser eliminados cuando son innecesarios.

- **DROP INDEX:** Este comando es usado para borrar los índices de la tabla relacionada y la tupla del catalogo.

Comandos para manipulación de datos:

- **SELECT:** Esta instrucción tienen como fin, recuperar la información desde una base de datos. Existen funciones que están relacionadas con el comando SELECT, por ejemplo:
  - **DISTINCT:** Antes de ejecutar la sentencia SELECT, esta instrucción borrara todos los errores de redundancia de datos que puedan existir.
  - **COUNT:** Se utiliza para obtener el número de valores en la columna.
  - **SUM:** Suma todos los elementos de una columna, siempre y cuando estos sean numéricos.
  - **AVG:** Hace un promedio de los datos numéricos de una columna.
  - **MIN o MAX:** Se usa para obtener el mayor o menor valor de una Columna.
  - **COUNT(*):** Se implementa para contar la cantidad de elementos por filas de una tabla sin eliminación de valores duplicados.
  - **GROUP BY:** Reordena de manera virtual, lógicamente y en grupos una tabla.
  - **HAVING:** Esta sentencia se usa para eliminar grupos de datos.
  - **ORDER BY:** Ordena la tabla en un orden especifico.
  - **EXIST:** Esta función es una especie de calificador de existencia, es decir, evalúa todos los procesos lógicos y se cumple cuando el retorno de estos no son nulos.
- **UPDATE:** Se utiliza para modificar los atributos de una o más tuplas seleccionadas.
- **DELETE:** Comando utilizado para borrar las tuplas desde una relación, si se digita solo, se borran todas, pero al combinarlo con el comando WHERE, se pueden seleccionar las tuplas que se van a borrar.
- **INSERT:** Agrega una tupla a una relación, para esto se debe especificar el nombre de la relación y una lista ordenada de valores que se agregaran a la tupla.

3.3. MySQL

Es un sistema open-source de gestión de bases de datos. Es una de las plataformas más utilizadas mundialmente, hoy en día pertenece a Oracle Corporation. Ésta, incluye múltiples motores de almacenamiento, como lo son: InnoDB, MyISAM, NDB, Memory, Merge, Archive, CVS, etcétera. Además tiene distintas funcionalidades que permiten que sea una

plataforma completa, como por ejemplo, MySQL Replication, que permite la escalabilidad de la base de datos; Stored Procedures, para mejorar la productividad del desarrollador; Views, para asegurar que información importante no es comprometida; MySQL Connectors, como ODBC, JDBC, .NET, etcétera, para desarrollar en múltiples lenguajes de programación; MySQL Workbench, para visualizar, administrar y desarrollar; entre muchas otras funcionalidades. Como existe mucho soporte oficial en internet para MySQL, la curva de aprendizaje es pequeña. Además, como ya se mencionó anteriormente, existe una versión gratuita con funcionalidades completas. Estas características y otras ya mencionadas vuelven MySQL Community Edition ideal para su implementación en la Plataforma Integral de Medición de Parámetros Eléctricos por Medio Inalámbrico (PIME).

3.3.1. Tipos de datos soportados

MySQL soporta un número de tipos de columnas divididos en varias categorías: tipos numéricos, tipos de fecha y hora, y tipos de cadenas de caracteres. En esta sección se proporciona una visión general de estos tipos de datos, una descripción más detallada de las propiedades de los tipos en cada categoría, y un resumen de los requisitos de almacenamiento de tipos de datos. La primera visión de conjunto es intencionadamente breve. Las descripciones más detalladas más adelante en el capítulo deben ser consultados para obtener información adicional acerca de los tipos de datos particulares, tales como los formatos admisibles en los que se pueden especificar valores.
Convenciones:

- **M**, Indica la máxima anchura al mostrar los datos. El máximo ancho de muestra es 255.
- **D**, Se aplica a tipos de punto flotante y de coma fija e indica el número de dígitos a continuación del punto decimal. El valor máximo de D posible es 30, pero no debe ser mayor que **M-2**.
- Los corchetes ('[' y ']') indican partes de especificadores de tipos que son opcionales

3.3.2. Datos numéricos

3.3.2.1. BIT [(M)]

En un tipo de datos bit. M indica el número de bits por valor, de 1 a 64. El valor por defecto es 1 si se omite M. Este tipo de datos se añadió en MySQL 5.0.3 para MyISAM, una extensión en 5.0.5 para MEMORY, InnoDB, y BDB. Antes de 5.0.3, BIT es un sinónimo de TINYINT (1).

3.3.2.2. TINYINT [(M)] [UNSIGNED] [ZEROFILL]

Un entero muy pequeño. El rango con signo es de -128 a 127. El rango sin signo es de 0 a 255.

3.3.2.3. BOOL, BOOLEAN

Son sinónimos para TINYINT (1). Un valor de cero se considera falso. Valores distintos a cero se consideran ciertos.

3.3.2.4. SMALLINT [(M)] [UNSIGNED] [ZEROFILL]

Un entero pequeño. El rango con signo es de -32768 a 32767. El rango sin signo es de 0 a 65535.

3.3.2.5. MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]

Entero de tamaño medio. El rango con signo es de -8388608 a 8388607. El rango sin singo es de 0 a 16777215.

3.3.2.6. INT [(M)] [UNSIGNED] [ZEROFILL]

Un entero de tamaño normal. El rango con signo es de -2147483648 a 2147483647. El rango sin signo es de 0 a 4294967295.

3.3.2.7. INTEGER [(M)] [UNSIGNED] [ZEROFILL]

Es un sinónimo de INT.

3.3.2.8. BIGINT [(M)] [UNSIGNED] [ZEROFILL]

Un entero grande. El rango con signo es de -9223372036854775808 a 9223372036854775807. El rango sin signo es de 0 a 18446744073709551615. Algunos aspectos a considerar con respecto a las columnas BIGINT: Toda la aritmética se hace usando valores BIGINT o DOUBLE, así que no debe usar enteros sin signos mayores que 9223372036854775807 (63 bits) excepto con funciones bit, si se hace, algunos de los últimos dígitos en el resultado pueden ser erróneos por culpa de errores de redondeo al convertir valores BIGINT a DOUBLE.

3.3.2.9. FLOAT (p) [UNSIGNED] [ZEROFILL]

Número con coma flotante. p representa la precisión. Puede ir de 0 a 24 para números de coma flotante de precisión sencilla y de 25 a 53 para números de coma flotante con doble precisión. Estos tipos son como los tipos FLOAT y DOUBLE descritos a continuación. FLOAT(p) tiene el mismo rango que los tipos correspondientes FLOAT y DOUBLE, pero la anchura

de muestra y el número de decimales no están definidos. En MySQL 5.0, este es un valor de coma flotante auténtico. Esta sintaxis se proporciona para compatibilidad con ODBC.

3.3.2.10. FLOAT [(M,D)] [UNSIGNED] [ZEROFILL]

Un número de coma flotante pequeño (de precisión simple). Los valores permitidos son de -3.402823466E+38 a -1.175494351E-38, 0, y de 1.175494351E-38 a 3.402823466E+38. Si se especifica UNSIGNED, los valores negativos no se permiten. M es la anchura de muestra y D es el número de dígitos significativos. FLOAT sin argumentos o FLOAT(p) (donde p está en el rango de 0 a 24) es un número de coma flotante con precisión simple.

3.3.2.11. DOUBLE [(M,B)] [UNSIGNED] [ZEROFILL]

Número de coma flotante de tamaño normal (precisión doble). Los valores permitidos son de -1.7976931348623157E+308 a -2.2250738585072014E-308, 0, y de 2.2250738585072014E-308 a 1.7976931348623157E+308. Si se especifica UNSIGNED, no se permiten valores negativos. M es la anchura de muestra y B es el número de bits de precisión. DOUBLE sin parámetros o FLOAT(p) (donde p está en el rango de 25 a 53) es un número de coma flotante con doble precisión. Un número de coma flotante con precisión sencilla tiene una precisión de 7 decimales aproximadamente; un número con coma flotante de doble precisión tiene una precisión aproximada de 15 decimales.

3.3.2.12. DOUBLE PRECISION [(M,D)] [UNSIGNED] [ZEROFILL], REAL[(M,D)] [UNSIGNED] [ZEROFILL]

Son sinónimos de DOUBLE. Excepción: Si el modo del servidor SQL incluye la opción REAL_AS_FLOAT, REAL es un sinónimo para FLOAT en lugar de DOUBLE.

3.3.2.13. DECIMAL [(M[,D])] [UNSIGNED] [ZEROFILL]

A partir de MySQL 5.0.3: Número de punto fijo exacto y empaquetado. M es el número total de dígitos y D es el número de decimales. El punto decimal y (para números negativos) el signo '-' no se tiene en cuenta en M. Si D es 0, los valores no tienen punto decimal o parte fraccional. El máximo número de dígitos (M) para DECIMAL es 64. El máximo número de decimales soportados (D) es 30. Si UNSIGNED se especifica, no se permiten valores negativos. Si se omite D, el valor por defecto es 0. Si se omite M, el valor por defecto es 10. Todos los cálculos básicos (+, -, *, /) con columnas DECIMAL se hacen con precisión de 64 dígitos decimales.

3.3.2.14. DEC [(M[,D])] [UNSIGNED] [ZEROFILL], NUMERIC [(M[,D])] [UNSIGNED] [ZEROFILL], FIXED [(M[,D])] [UNSIGNED] [ZEROFILL]

Son sinónimo de DECIMAL. El sinónimo FIXED está disponible por compatibilidad con otros servidores. En resumen de tipos de datos para columnas tipo numéricos se pueden observar en la Tabla **3-1**.

Tipo	Bytes	Valor Mínimo	Valor Máximo
		(Con signo/Sin signo)	(Con signo/Sin signo)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Tabla 3-1.: Tipos de datos de columnas numéricos en MySQL [5].

3.3.3. Datos Tipo Fecha y hora

Un resumen de los tipos de columnas temporales se muestra a continuación.

3.3.3.1. DATE

Una fecha. El rango soportado es de '1000-01-01' a '9999-12-31'. MySQL muestra valores DATE en formato 'YYYY-MM-DD', pero permite asignar valores a columnas DATE usando cadenas de caracteres o números.

3.3.3.2. DATETIME

Combinación de fecha y hora. El rango soportado es de '1000-01-01 00:00:00' a '9999-12-31 23:59:59'. MySQL muestra valores DATETIME en formato 'YYYY-MM-DD HH:MM:SS', pero permite asignar valores a las columnas DATETIME usando cadenas de caracteres o números.

3.3.3.3. TIMESTAMP

Una marca temporal. El rango es de '1970-01-01 00:00:00' hasta el año 2037. Una columna TIMESTAMP es útil para registrar la fecha y hora de una operación INSERT o UPDATE.

La primera columna TIMESTAMP en una tabla se rellena automáticamente con la fecha y hora de la operación más reciente si no le asigna un valor. Puede asignar a cualquier columna TIMESTAMP la fecha y hora actual asignándole un valor NULL.

3.3.3.4. TIME

Una hora. El rango es de '-838:59:59' a '838:59:59'. MySQL muestra los valores TIME en formato 'HH:MM:SS', pero permite asignar valores a columnas TIME usando números o cadenas de caracteres. La razón por la que la parte de hora puede ser tan grande es que el tipo TIME puede usarse no sólo para representar una hora del día (que debe ser menor a 24 horas), pero también el tiempo transcurrido o un intervalo de tiempo entre dos eventos (que puede ser mucho mayor a 24 horas, o incluso negativo).

3.3.3.5. YEAR [(2—4)]

Un año en formato de dos o cuatro dígitos. El valor por defecto está en formato de cuatro dígitos. En formato de cuatro dígitos, los valores permitidos son de 1901 a 2155, y 0000. En formato de dos dígitos, los valores permitidos son de 70 a 69, representando los años de 1970 a 2069. MySQL muestra los valores YEAR en formato YYYY pero permite asignar valores a columnas YEAR usando cadenas de caracteres o números. En resumen de tipos de formatos para columnas tipo fecha se pueden observar en la tabla **3-2**.

Tipo de Columna	"Cero" Valor
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000
TIME	'00:00:00'
YEAR	0000

Tabla 3-2.: Formato para datos tipo Fecha en MySQL [5].

3.3.4. Datos Tipo Caracteres

Un resumen de los tipos de columnas para caracteres se muestra a continuación. Las definiciones de columnas para varios tipos de datos de cadenas de caracteres incluyen un atributo CHARACTER SET para especificar el conjunto de caracteres y, ocasionalmente, una colación. (CHARSET es sinónimo de CHARACTER SET.) Estos atributos se aplican a los tipos CHAR, VARCHAR, TEXT, ENUM, y SET. Por ejemplo:

```
CREATE TABLE t
( c1 CHAR(20) CHARACTER SET utf8,
 c2 CHAR(20) CHARACTER SET latin1 COLLATE latin1_bin);
```

El atributo BINARY es una abreviatura para especificar la colación binaria del conjunto de caracteres de la columna. La ordenación y comparación se basa en los valores numéricos de los caracteres. El tipo de columna CHAR BYTE es un alias para CHAR **BINARY**. Esta es una característica de compatibilidad. El atributo ASCII puede especificarse para CHAR. Asigna el conjunto de caracteres **latin1**. El atributo **UNICODE** puede especificarse en MySQL 5.0 para CHAR. Asigna el conjunto de caracteres ucs2.

3.3.4.1. CHAR

Es un sinónimo de CHAR(1).

3.3.4.2. [NATIONAL] VARCHAR (M) [BINARY]

Cadena de caracteres de longitud variable. M representa la longitud de columna máxima. En MySQL 5.0, el rango de M es de 0 a 255 antes de MySQL 5.0.3, y de 0 a 65,535 en MySQL 5.0.3 y posterior. (La longitud máxima real de un VARCHAR en MySQL 5.0 se determina por el tamaño de registro máximo y el conjunto de caracteres que use. La longitud máxima efectiva desde MySQL 5.0.3 es de 65,532 bytes.) VARCHAR es la abreviación de CHARACTER VARYING.

3.3.4.3. BINARY (M)

El tipo BINARY es similar al tipo CHAR, pero almacena cadenas de datos binarios en lugar de cadenas de caracteres no binarias.

3.3.4.4. VARBINARY (M)

El tipo BINARY es similar al tipo CHAR, pero almacena cadenas de datos binarios en lugar de cadenas de caracteres no binarias.

3.3.4.5. BLOB y TEXT

Un BLOB es un objeto binario que puede tratar una cantidad de datos variables. Los cuatro tipos BLOB son TINYBLOB, BLOB, MEDIUMBLOB, y LONGBLOB. Difieren sólo en la longitud máxima de los valores que pueden tratar. Los cuatro tipos TEXT son TINYTEXT, TEXT, MEDIUMTEXT, y LONGTEXT. Se corresponden a los cuatro tipos BLOB y tienen las mismas longitudes y requerimientos de almacenamiento. El tamaño máximo de un objeto BLOB o TEXT se determina por su tipo, pero el valor máximo que puede transmitir entre

el cliente y el servidor viene determinado por la cantidad de memoria disponible y el tamaño de los buffers de comunicación. Puede cambiar el tamaño de los buffers de comunicación cambiando el valor de la variable **max_allowed_packet**, pero debe hacerlo para el servidor y los clientes.

3.3.4.6. El tipo de columna ENUM

Un ENUM es un objeto de cadenas de caracteres con un valor elegido de una lista de valores permitidos que se enumeran explícitamente en la especificación de columna en tiempo de creación de la tabla.
El valor puede ser la cadena vacía (”) o NULL bajo ciertas circunstancias: Si inserta un valor inválido en un ENUM (esto es, una cadena de caracteres no presente en la lista de valores permitidos), la cadena vacía se inserta en lugar de un valor especial de error. Esta cadena puede distinguirse de una cadena vacía “normal” por el hecho que esta cadena tiene un valor numérico 0.
Si se declara una columna ENUM para permitir NULL, el valor NULL es un valor legal para la columna, y el valor por defecto es NULL. Si una columna ENUM se declara NOT NULL, su valor por defecto es el primer elemento de la lista de valores permitidos.

3.3.5. Requisitos de almacenamiento según el tipo de columna

En esta sección se muestra un resumen de los requerimientos de los datos soportados, Los requerimientos de almacenamiento para cada uno de los tipos de columnas soportados por MySQL se listan por categoría. El máximo tamaño de un registro en una tabla MyISAM es 65534 bytes. Cada columna BLOB y TEXT cuenta sólo de cinco a nueve bytes más allá de su tamaño.

Tipo de columna	Almacenamiento requerido
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(*p*)	4 bytes si 0 <= *p* <= 24, 8 bytes si 25 <= *p* <= 53
FLOAT	4 bytes
DOUBLE [PRECISION], objeto REAL	8 bytes
DECIMAL(*M*, *D*), NUMERIC(*M*, *D*)	Varía; consulte la siguiente explicación
BIT(*M*)	aproximadamente (*M*+7)/8 bytes

Tabla 3-3.: Requerimientos de almacenamiento para tipos numéricos [5].

Tipo de columna	Almacenamiento requerido
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

Tabla 3-4.: Requerimientos de almacenamiento para tipos fecha y hora [5].

Tipo de columna	Almacenamiento requerido
CHAR(*M*)	*M* bytes, 0 <= *M* <= 255
VARCHAR(*M*)	*L*+1 bytes, donde *L* <= *M* y 0 <= *M* <= 255
BINARY(*M*)	*M* bytes, 0 <= *M* <= 255
VARBINARY(*M*)	*L*+1 bytes, donde *L* <= *M* y 0 <= *M* <= 255
TINYBLOB, TINYTEXT	*L*+1 byte, donde *L* < 2^8
BLOB, TEXT	*L*+2 bytes, donde *L* < 2^16
MEDIUMBLOB, MEDIUMTEXT	*L*+3 bytes, donde *L* < 2^24
LONGBLOB, LONGTEXT	*L*+4 bytes, donde *L* < 2^32
ENUM('*value1*','*value2*',...)	1 o 2 bytes, dependiendo del número de valores de la enumeración (65,535 valores como máximo)
SET('*value1*','*value2*',...)	1, 2, 3, 4, o 8 bytes, dependiendo del número de miembros del conjunto (64 miembros como máximo)

Tabla 3-5.: Requerimientos de almacenamiento para tipos carácter [5].

3.3.6. Funciones y operadores

MySQL maneja cientos de funciones y procedimientos para manejo de columnas tipo numéricos, Fecha, hora y caracteres. También permite crear funciones y procedimientos directamente en la base de datos. Las funciones de MySQL se pueden acceder directamente desde una query ejemplo:

```
MySQL> Select 1 && 1
    -> 1
MySQL> Select if(3>5,2,4)
    -> 4
MySQL> Select year('2017-05-01')
    -> 2017
MySQL> Select my_procedimiento();
    -> ok
```

En este capítulo se mostrara los operadores, funciones y procedimiento que competen a este proyecto.

3.3.6.1. Operadores

A continuación se lista una lista de operadores disponibles en MySQL

Operadores Lógica binaria	| , or logical bit a bit ~ , inversion de un bit & , and lógica Bit a bit >> << !
Operadores de comparación Lógica para control de flujo	|| OR && AND XOR > < >= <= = IS NOT NULL LIKE
Operadores aritméticos	+ - * / % MOD

Tabla 3-6.: Operadores MySQL.

3.3.6.2. Year, Month, Day, Date, Hour

Estas funciones devuelven el año, mes, día, fecha en formato años-mes-dia y hora de una cadena tipo date, datetime. Ejemplo:

```
MySQL> select Year ('2017-12-31 23:59:59');
    -> 2017
MySQL> select Day ('2017-12-31 23:59:59');
    -> 12
MySQL> select Date ('2017-12-31 23:59:59');
    -> '2017-12-31'
MySQL> select Date ('2017-12-31 23:59:59');
    -> 23
```

3.3.6.3. Función AES_ENCRYPT (str, key_str)

Es una función Estándar de Encriptación Avanzada o AES (Advanced Encryption Standard), también conocido como Rijndel, especifica un algoritmo criptográfico. Fue establecido por el U.S. National Institute of Standards and Technology (NIST). Esta función encripta un dato de acuerdo a una llave de encriptación ejemplo:

```
INSERT INTO t
VALUES (1,AES_ENCRYPT('text',UNHEX('F9A0B379441B830D21A390C3')));
```

Se usa por ejemplo para guardar las claves en la base de datos donde aunque el administrador de la base de datos vea los datos de las tablas si no posee la llave de encriptación no podrá ver el contenido cifrado.

3.3.6.4. Función AES_DECRYPT(str, key_str)

Esta función desencripta los datos usando la llave de encriptación dada en una cadena.

3.3.7. Funciones y/o Procedimiento de almacenado

Los procedimientos de almacenados y/o funciones son nuevas funcionalidades de la versión de MySQL 5.0. Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado. La diferencia entre un procedimiento y una función es que la función retorna un único valor de cualquier tipo de dato soportado por MySQL y se llama por "Select My_fucntion (parametro1, par1, par3);" y un procedimiento puede retornar muchos datos como una tabla o no retornar valor y su forma de invocarlo es "Call My_Procedure (par1, par2)". El lenguaje

sql de un procedimiento de almacenado requiere de cierto orden y estructura para que sea correctamente interpretado por el servidor por lo que hay que ser cuidadosos al momento de crearlos. La estructura de un procedimiento se observa a continuación:

```
DELIMITER //
 CREATE PROCEDURE GetAllProducts()
   BEGIN
   declare kw double;  /* declaracion de variables */
   Codigo de usuario

   END //
 DELIMITER;
```

3.3.7.1. Sentencia CREATE PROCEDURE Y CREATE FUNCTION

A través de estas sentencias SQL, se puede crear funciones y procedimientos de almacenado en la base de datos con la siguiente estructura:

```
CREATE PROCEDURE sp_name ([parameter[,...]])
    [characteristic ...] routine_body

CREATE FUNCTION sp_name ([parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

parameter:
    [ IN | OUT | INOUT ] param_name type

type:
    Any valid MySQL data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

routine body:
    procedimientos almacenados o comandos SQL validos
```

Estos comandos crean una rutina almacenada. Desde MySQL 5.0.3, para crear una rutina, es necesario tener el permiso CREATE ROUTINE, y los permisos ALTER ROUTINE y EXECUTE se asignan automáticamente a su creador.

3.3.7.2. Sentencia compuesta BEGIN ... END

```
[etiqueta_inicio:] BEGIN
    [lista_sentencias]
END [etiqueta_fin]
```

La sintaxis BEGIN ... END se utiliza para escribir sentencias compuestas que pueden aparecer en el interior de procedimientos almacenados y triggers. Una sentencia compuesta puede contener múltiples sentencias, encerradas por las palabras BEGIN y END. lista_sentencias es una lista de una o más sentencias. Cada sentencia dentro de lista_sentencias debe terminar con un punto y como (;) delimitador de sentencias. lista_sentencias es opcional, lo que significa que la sentencia compuesta vacía (BEGIN END es correcta.
Ejemplo:

```
DELIMITER //
 CREATE PROCEDURE GetAllProducts()
   BEGIN
   SELECT * FROM products;
   END //
 DELIMITER ;
```

3.3.7.3. Sentencia DELIMITER

Esta sentencia como su nombre lo dice permite cambiar el delimitador tradicional ";" a uno personalizado como en el ejemplo anterior //.

3.3.7.4. Variables en procedimientos almacenados

En los procedimientos de almacenado a veces se requiere de variables para manipular los datos, el usuario puede declarar cualquier tipo de datos soportado por MySQL. En MySQL existen variables locales que solo existen en el procedimiento o función de almacenado y variables de sesión o de usuario que es una especie de variable global que existe mientras el cliente tiene una sesión abierta con el servidor y no requiere declararla. Un ejemplo de ambas es:

```
DECLARE total double; /* Ejemplo de Variable Local */
DECLARE f1 datetime; /* Variable Local tipo Date and time */
DECLARE done INT DEFAULT FALSE;
 Set @error =-8001;    /* Ejemplo de variable de sesion */
```

3.3.7.5. Sentencia DECLARE

Con esta sentencia SQL se pueden declarar variables dentro los procedimientos y funciones de almacenado. También se usa para dar tratamiento específico, Asocia un nombre con una condición de error específica. El nombre puede usarse subsecuentemente en un comando DECLARE HANDLER. Ver ejemplo sección anterior.

3.3.7.6. Sentencia DECLARE Handlers

```
DECLARE handler_type HANDLER FOR condition_value[,...] sp_statement

handler_type:
    CONTINUE
  | EXIT
  | UNDO

condition_value:
    SQLSTATE [VALUE] sqlstate_value
  | condition_name
  | SQLWARNING
  | NOT FOUND
  | SQLEXCEPTION
  | MySQL_error_code
```

Este comando especifica handlers que pueden tratar una o varias condiciones. Si unas de estas condiciones ocurren, el comando especificado se ejecuta. Para un handler CONTINUE, continúa la rutina actual tras la ejecución del comando del handler. Para un handler EXIT, termina la ejecución del comando compuesto BEGIN...END actual. El handler de tipo UNDO todavía no se soporta. SQLWARNING es una abreviación para todos los códigos SQLSTATE que comienzan con 01. NOT FOUND es una abreviación para todos los códigos SQLSTATE que comienzan con 02. SQLEXCEPTION es una abreviación para todos los códigos SQLSTATE no tratados por SQLWARNING o NOT FOUND. Además de los valores SQLSTATE, los códigos de error MySQL se soportan. Ejemplo:

```
DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
```

CURSOR	Comando para crear el cursor
OPEN	Comando,para abrir el cursor
FETCH	Comando,para acceder a datos de una fila del cursor
CLOSE	Comando,para cerrar el cursor

Tabla 3-7.: Comando reservados para usar cursores.

3.3.7.7. Sentencia SET

Con esta sentencia SQL se asigna un valor a una variable en específico, ejemplo:

```
set total = 25000;
set @error =-8015;
```

3.3.7.8. Sentencia SELECT ... INTO

Cuando el cliente desea asignar a la variable un valor de un dato que está almacenado en una tabla en la base de datos, el cliente puede usar la sentencia select ... into con el siguiente formato:

```
SELECT col_name[,...] INTO var_name[,...] table_expr
```

Ejemplo:

```
Declare x,y integer;
SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
```

3.3.7.9. Cursores

Cuando dentro de un procedimiento o función de almacenado se requiere acceso a datos en más de una fila se usa un cursor para acceder a los datos de un SELECT. Un cursor es un objeto que proporciona acceso mediante programación al conjunto de resultados devuelto por su SELECT declaración. Utilizar un cursor para recorrer las filas del conjunto de resultados y tomar medidas para cada fila individual. MySQL soporta cursores simples dentro de procedimientos y funciones almacenadas. La sintaxis es la de SQL empotrado. Los cursores no son sensibles, son de sólo lectura, y no permiten scrolling (acceder a filas ya accedidas). No sensible significa que el servidor puede o no hacer una copia de su tabla de resultados. Los cursores deben declararse antes de declarar los handlers, y las variables y condiciones deben declararse antes de declarar cursores o handlers. Los cursores usan la siguiente lista de comandos reservados:
Ejemplo:

```
CREATE PROCEDURE cursor_demo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE a CHAR(16);
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

  OPEN cur1;
  OPEN cur2;

  REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
          INSERT INTO test.t3 VALUES (a,b);
       ELSE
          INSERT INTO test.t3 VALUES (a,c);
       END IF;
    END IF;
  UNTIL done END REPEAT;

  CLOSE cur1;
  CLOSE cur2;
END
```

3.3.7.10. Bloques

En el desarrollo del proyecto se requirió hacer consultas SQL después de abrir un cursor, Para lograr esto se usan bloques de BEGIN END dentro de otros BEGIN ... END para poder hacerlo ejemplo:

```
DELIMITER //
CREATE PROCEDURE curdemo ()
BLOCK1: BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
```

```
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

  OPEN cur1;
  REPEAT
    FETCH cur1 INTO a, b;
  UNTIL done END REPEAT;
       .. User Code

       BLOCK2: BEGIN
            DECLARE b1, c1 DOUBLE;
            DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
           OPEN cur2;
            .. User Code
             CLOSE cur2;
       END BLOBK2;

  CLOSE cur1;
END BLOBK1;
DELIMITER //
```

3.3.8. Constructores de control de flujo

Los constructores IF, CASE, LOOP, WHILE, ITERATE, y LEAVE están completamente implementados. Estos constructores pueden contener un comando simple, o un bloque de comandos usando dentro del comando compuesto BEGIN ... END. Los constructores pueden estar anidados. Los bucles FOR no están soportados.

3.3.8.1. Sentencia IF

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF;
```

La sentencia IF implementa un constructor condicional básico. Si searchcondition se evalúa a cierto, el comando SQL correspondiente listado se ejecuta. Si no coincide ninguna search-condition se ejecuta el comando listado en la cláusula ELSE. statementlist puede consistir en varios comandos anidados. Ejemplo:

```
if (is_festivo =1) then
        set dia1=1;
        set dia2=1;
    else
        set dia1= (SELECT DAYOFWEEK(f1));
        set dia2=dia1;
 end if;
```

3.3.8.2. Sentence CASE

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE;
```

ó

```
CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list] ...
    [ELSE statement_list]
END CASE;
```

La sentencia CASE implementa muchos IF internamente para evaluar la condición pero lo hace en menos líneas de códigos visuales, haciendo atractivo para evaluar muchas condiciones en una código más compacto.
Ejemplo:

```
            CASE Tipo1
              WHEN 'BASE' THEN
                    if aux > DB then
                       SET DB=aux;
                    end if;
              WHEN 'INTERMEDIO' THEN
                    if aux > DI then
                       SET DI=aux;
                    end if;
              WHEN 'PUNTA' THEN
                    if aux > DP then
                       SET DP=aux;
```

```
                end if;
        END CASE;
```

3.3.8.3. Sentencia LOOP

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

LOOP implementa un constructor de bucle simple que permite ejecución repetida de comandos particulares. El comando dentro del bucle se repite hasta que acaba el bucle, usualmente con un comando LEAVE. Un comando LOOP puede etiquetarse. end_label no puede darse hasta que esté presente begin_label , y si ambos lo están, deben ser el mismo.
Ejemplo:

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN
      ITERATE label1;
    END IF;
    LEAVE label1;
  END LOOP label1;
  SET @x = p1;
END;
```

3.3.8.4. Sentencia ITERATE

```
ITERATE label;
```

ITERATE sólo puede aparecer en comandos LOOP, REPEAT, y WHILE. ITERATE significa “vuelve a hacer el bucle.” Ó ir a la etiqueta. Ver ejemplo sección anterior.

3.3.8.5. Sentencia REPEAT

```
[begin_label:] REPEAT
    statement_list
UNTIL search_condition
END REPEAT [end_label]
```

El comando/s dentro de un comando REPEAT se repite hasta que la condición search_condition es cierta evaluada con UNTIL. Un comando REPEAT puede etiquetarse. end_label no puede darse a no ser que begin_label esté presente, y si lo están, deben ser el mismo. Por ejemplo:

```
delimiter //
CREATE PROCEDURE dorepeat(p1 INT)
BEGIN
SET @x = 0;
REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
END //
```

3.3.8.6. Sentencia WHILE

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

El comado/s dentro de un comando WHILE se repite mientras la condición search_condition es cierta. Un comando WHILE puede etiquetarse. end_label no puede darse a no ser que begin_label también esté presente, y si lo están, deben ser el mismo. Por ejemplo:

```
CREATE PROCEDURE DOwhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;
  WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END
```

3.4. Equipos de medición de parámetros eléctricos

En este capítulo, se muestran los dos medidores de parámetros eléctricos que soporta el software PM850 31 de Schneider Electric y el S203TA-D de SENECA, como sus protocolos de comunicación.

3.4.1. Modbus RTU

Modbus es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), diseñado en 1979 por

Modicon para su gama de controladores lógicos programables (PLCs). Las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son:

- Es público
- Su implementación es fácil y requiere poco desarrollo
- Maneja bloques de datos sin suponer restricciones

En este protocolo el primer byte es la dirección del esclavo de destino el segundo es el byte de comando los siguientes son adicionales a este comando y por últimos los dos últimos dos byte son un código de redundancia cíclica que se calcula a partir de los bytes antes transmitidos y permite verificar que la trama recibida es correcta.

3.4.2. Modbus ASCII

Este protocolo es idéntico al Modbus RTU siendo que en este se envían los datos en formato ASCII lo que hace que utilice más bytes para el envió de información pero permite ser entendido más fácilmente por el ser humano.

3.4.3. Modbus TCP

Este protocolo es idéntico al Modbus RTU siendo que se envían los datos vía TCP/IP y no serial como el RTU, este protocolo puede enviar datos en forma RTU o ASCII. Los medidores utilizados en el proyecto manejan el protocolo Modbus RTU y se cuenta con un módulo TCP/IP para conectarlo directamente a la LAN de la empresa u hogar. De esta forma, el sistema adquiere los datos mediante el protocolo TCP/IP sin necesidad de otro hardware.

3.4.4. Medidor PM850

Es una central de medida, que es un dispositivo multifuncional de instrumentación digital, adquisición de datos y control. Puede sustituir distintos medidores, relés, transductores y Otros componentes. Esta central de medida está equipada con comunicaciones RS485 y Ethernet para su integración en cualquier sistema de control Modbus RTU, TCP/IP y supervisión de potencia y se puede Instalar en varios lugares de un edificio.
Se trata de medidores de rms real, capaces de medir con una precisión excepcional altas cargas no lineales. Su sofisticada técnica de muestreo permite realizar mediciones precisas hasta el armónico de orden 63. Es posible visualizar más de 50 valores de medición, además de datos máximos y mínimos, ya sea directamente en la pantalla o de Forma remota con el software. En la Figura **3-1**. Se resumen las lecturas disponibles de la central de medida PM850 de Schneider Electric [7].
El equipo de medición se puede observare en la Figura **3-2**.

Lecturas en tiempo real	Análisis de la energía
• Intensidad (por fase, residual, trifásica) • Tensión (L-L, L-N, trifásica) • Potencia activa (por fase, trifásica) • Potencia reactiva (por fase, trifásica) • Potencia aparente (por fase, trifásica) • Factor de potencia (por fase, trifásico) • Frecuencia • THD (intensidad y tensión)	• Factor de potencia de desplazamiento (por fase, trifásico) • Tensiones fundamentales (por fase) • Intensidades fundamentales (por fase) • Potencia activa fundamental (por fase) • Potencia reactiva fundamental (por fase) • Desequilibrio (intensidad y tensión) • Rotación de fases • Magnitudes y ángulos de tensión e intensidad de los armónicos (por fase)① • Componentes de secuencia
Lecturas de energía	**Lecturas de la demanda**
• Energía acumulada: activa • Energía acumulada: reactiva • Energía acumulada: aparente • Lecturas bidireccionales • Energía reactiva por cuadrante • Energía incremental • Energía condicional	• Demanda de intensidad (por fase presente, media trifásica) • Media de factor de potencia (total trifásico) • Demanda de potencia activa (por fase presente, punta) • Demanda de potencia reactiva (por fase presente, punta) • Demanda de potencia aparente (por fase presente, punta) • Lecturas coincidentes • Demandas de potencia pronosticadas

Figura 3-1.: Resumen de características principales de la central de medida PM850 [7].

Figura 3-2.: Medidor PM850.

3.4.5. Medidor S203TA-D

El modelo S203TA-D es un analizador trifásico de redes, que soporta 600Vac de entrada y una corriente máxima de 5A. Puede medir los siguientes parámetros: Vrms, Irms, Watt, VAR, VA, frecuencia y $Cos\varphi$. Los valores se pueden medir por medio de comunicación serial tanto en punto flotante como en formato normalizado (excepto la frecuencia y la energía activa). Soporta protocolo RS485 Modus RTU [8]. Para implementar este medidor se desarrolló una interfaz de conversión Modbus RTU RS485 a TCP/IP con una Beaglebone desarrollada por el estudiante pasante Luis Gerardo González Salazar. Ya que LabVIEW ya posee las librerías para comunicación Modbus TCP/IP la cual tiene la facilidad de implementación en cualquier red Ethernet o wifi.

Sobre Modbus RTU RS485 se puede medir:

- V RMS (fase A a neutral)
- V RMS (fase B a neutral)
- V RMS (fase C a neutral)
- V RMS 3-fase
- I RMS fase A
- I RMS fase B
- I RMS fase C
- I RMS 3-fase
- P activa fase A
- P activa fase B
- P activa fase C
- P activa 3-fase
- Q reactiva fase A
- Q reactiva fase B
- Q reactiva fase C
- Q reactiva 3-fase

- S aparente fase A
- S aparente fase B
- S aparente fase C
- Cos fi 3-fase
- activa Energía fase A
- activa Energía fase B
- activa Energía fase C
- activa Energía 3-fase
- solo Energía activa positiva fase A
- solo Energía activa positiva fase B
- solo Energía activa positiva fase C
- solo Energía activa positiva 3-fase
- solo Energía activa negativa fase A
- solo Energía activa negativa fase B
- solo Energía activa negativa fase C
- solo Energía activa negativa 3-fase
- fase A frecuencia
- Reactiva Energía fase A
- Reactiva Energía fase B
- Reactiva Energía fase C
- Reactiva Energía 3-fase
- solo Energía reactiva positiva fase A
- solo Energía reactiva positiva fase B
- solo Energía reactiva positiva fase C
- solo Energía reactiva positiva 3-fase

- solo Energía reactiva negativa fase A
- solo Energía reactiva negativa fase B
- solo Energía reactiva negativa fase C
- solo Energía reactiva negativa 3-fase
- Máximo P activa valor fase A
- Máximo P activa valor 3-fase
- Máximo Q Reactiva valor fase A
- Máximo Q Reactiva valor 3-fase
- Promedio potencia activa 3-fase/A-fase
- Valor de demanda por tiempo (configurable).

El medidor S203TA-D de SENECA se puede observar en la Figura **3-3**.

Figura 3-3.: Medidor S203TA-D.

4. Tarifas Eléctricas y Diseño en MySQL

Las tarifas eléctricas que se utilizan en México las regula Comisión Federal de Electricidad y están organizadas de acuerdo a una diversos parámetros como: región, temperatura, promedio de consumo mensual de energía, demanda de potencia, destino de energía etc. De aquí Comisión Federal de Electricidad genera tres grandes rubros de tarifas, Casa, Negocio e Industria, las tarifas de Negocio e Industria utilizan los mismos algoritmos de cálculo por lo que solo se desarrollan las tarifas de Negocio.

Las tarifas para casa se clasifican principalmente por: región,temperatura media en verano de la región, rangos de consumo y energía destinada a uso domestico. También se caracteriza estas tarifas por no aplicar demandas potencia ni recargos por bajo factor de potencia.

Las tarifas de negocios usan todos los parámetros de clasificación mencionados e incluso en algunas tarifas el costo del KWH depende del día de la semana y hora. Un concepto importante de clasificación para las tarifas de negocio es que aparte del consumo promedio total, es la demanda de potencia eléctrica del negocio concepto que aparece en las facturas de CFE como demanda facturable y este ultimo parámetro influye altamente en costo total de la factura ya que el factor de carga del negocio es afectado por grandes consumos de energía en cortos periodos de tiempo; También en estas tarifas debido a su consumo superior sobre las tarifas de casa, la corrección de factor de potencia es bonificada si se tiene un factor por encima del 90 % y penalizada si la medición del factor de potencia está por debajo de 90 %. Como también un sobre cargo si la medición es en baja tension.

En este capítulo, se describen las tarifas que la comisión federal de electricidad CFE aplica en México, también las rutinas y procedimientos de almacenado diseñadas para cálculo de costos en MySQL. Las tarifas se describen en el orden de la página web de CFE www.cfe.gob.mx; de la misma página se extrajeron segmentos del texto que es la fuente más explícita sobre el uso y aplicación de cada tarifa [9].

4.1. Tarifa Casa 1

Esta tarifa tiene costos dependiendo del rango de consumo, como se observa a continuación:
Cargos por energía consumida para el año 2017
Consumo básico $ 0.793 por cada uno de los primeros 75 (setenta y cinco) kilowatts-hora.
Consumo intermedio $ 0.956 por cada uno de los siguientes 65 (sesenta y cinco) kilowatts-hora.
Consumo excedente $ 2.802 por cada kilowatt-hora adicional a los anteriores.
Mínimo mensual El equivalente a 25 (veinticinco) kilowatts-hora

Se pudo observar que los valores de los rangos de la tarifa no varían mes con mes por lo que se tuvo en cuenta para diseñar la tabla para manejo de la tarifa Casa 1.

4.1.1. Diseño Tabla Tcasa_1

En esta tabla, se diseñó para contener los rangos de KWH (rmin,rmax) y el costo de KHW en el rango estipulado por CFE además de futuras actualizaciones de tarifa. Por lo que se agregó la columna year sumado a una llave única conformada por las columnas (year,rmin,rmax) ya que estas tarifas cambiar anualmente; para la tabla diseñada se puede observar en la Figura **4-1**. Editada con MySQL Query Browser.

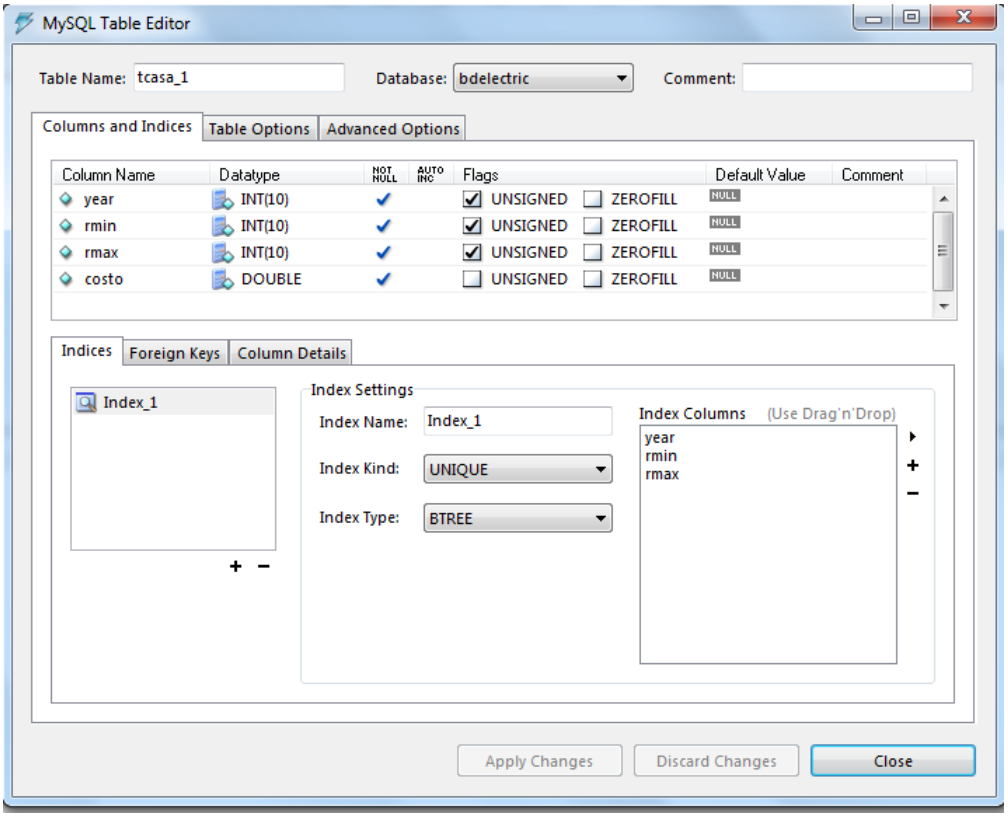


Figura 4-1.: Diseño tabla tarifa casa 1.

El código SQL para creación de la tabla se puede observar en la tabla **4-1**.

```
DROP TABLE IF EXISTS `BDELECTRIC`.`TCASA_1`;
CREATE TABLE `BDELECTRIC`.`TCASA_1` (
  `YEAR` INT(10) UNSIGNED NOT NULL,
  `RMIN` INT(10) UNSIGNED NOT NULL,
  `RMAX` INT(10) UNSIGNED NOT NULL,
  `COSTO` DOUBLE NOT NULL,
  UNIQUE KEY `INDEX_1` (`YEAR`,`RMIN`,`RMAX`)
) ENGINE=INNODB DEFAULT CHARSET=LATIN1;
```

Tabla 4-1.: Código SQL para creación de tabla casa 1

Un ejemplo de dato en el archivo txt separado por comas para importación y actualización de tarifa en la tabla se observa en la tabla **4-2**. Nótese que se colocó como rango máximo 10000kwh, ya que si la persona excede el consumo CFE lo cambiaría de tarifa.

```
Ano,min,max,costo
2016,1,75,0.793
2016,76,140,0.956
2016,141,10000,2.802
```

Tabla 4-2.: Ejemplo de formato de datos según figura **4-1** para actualización de tabla Tcasa_1

Nota cabe notar que el rango mínimo debe ser 1, para que no afecte el cálculo en un KWH Por lo que el algoritmo de cálculo se observa se la Figura **4-2**. Donde fecha1 es la fecha de inicio del cálculo, fecha2 es fecha de final de cálculo.

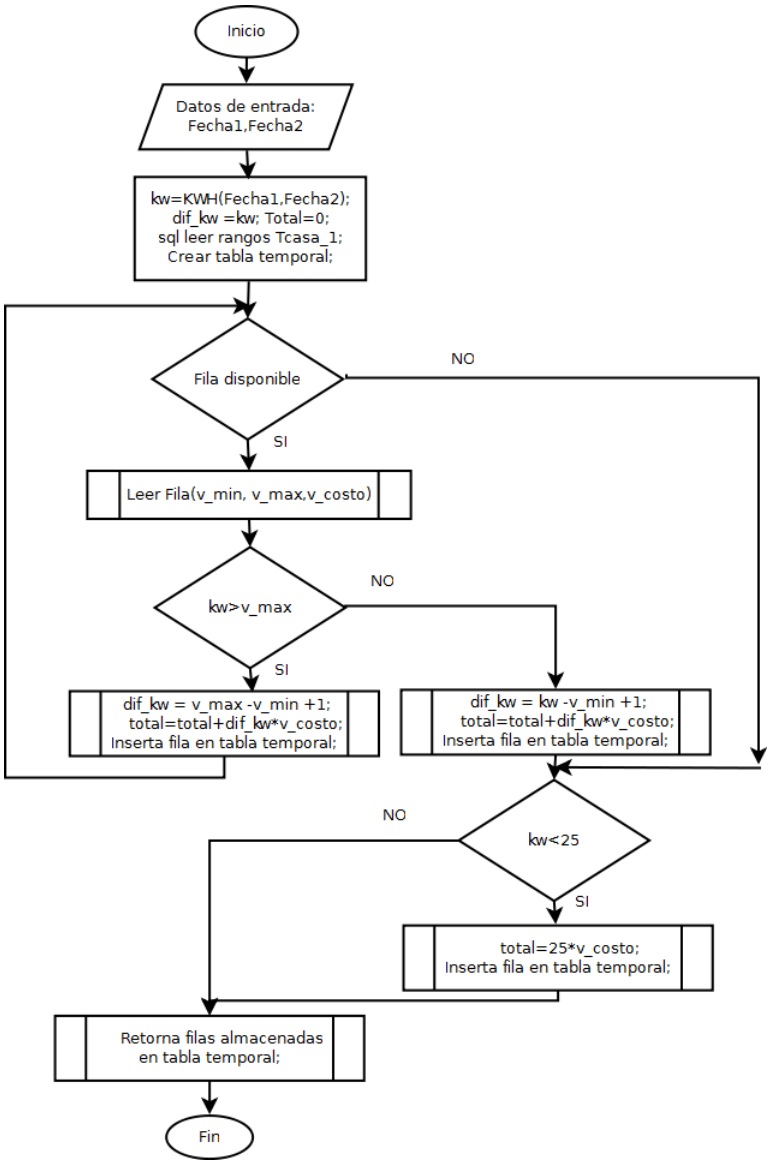


Figura 4-2.: Diagrama de flujo para cálculo de Tarifa Casa 1.

El código SQL del procedimiento se observa a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `Calc_Tcasa_1` $$
CREATE DEFINER=`root`@`localhost`
PROCEDURE `Calc_Tcasa_1`(Fecha1 datetime,Fecha2 datetime)
BEGIN
  DECLARE total double;
  declare  dif_kw double;
  declare  error SMALLINT;

  DECLARE v_min INT; -- variable para rango minimo
  DECLARE v_max int; -- Variable Rango maximo
  DECLARE v_costo,kw double; -- Costo y consumo kw
  DECLARE done INT DEFAULT FALSE;
  DECLARE cursor_i CURSOR FOR SELECT rmin,rmax,costo
  FROM tcasa_1 where tcasa_1.year= year(Fecha2)  ;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  DROP TABLE IF EXISTS Tcasa1;
  CREATE TEMPORARY TABLE Tcasa1(rmin SMALLINT,rmax SMALLINT,
  Costo_kw float,kw float,subtotal float,total float);
  OPEN cursor_i;
  set total =0;
  set error =0;
  set kw = KWH(fecha1,Fecha2,2); --Busca consumo en KWH
-- en la fecha especifica si devuelve valor negativo
-- es porque hubo algun error
  if kw <0 then  set error = kw; end if;

  set dif_kw =kw;

  read_loop: LOOP
    FETCH cursor_i INTO v_min, v_max,v_costo;
    IF done THEN
      LEAVE read_loop;
    END IF;

   if (kw >=v_max) then
     set dif_kw = v_max -v_min +1;
     set total=total+dif_kw*v_costo;
```

```
    else
      set dif_kw = kw -v_min+1;
          if dif_kw <=0 then
          LEAVE read_loop;
          end if;
       set total=total+dif_kw*v_costo;
    end if;

 INSERT INTO Tcasa1(rmin ,rmax ,Costo_kw ,kw ,subtotal ,total)
   VALUES (v_min,v_max,v_costo,dif_kw,dif_kw*v_costo,total);

  END LOOP;
  CLOSE cursor_i;

      if (kw <25) then
       set total=25*v_costo;
      INSERT INTO Tcasa1(rmin ,rmax ,Costo_kw ,kw ,subtotal ,total)
      VALUES (0,0,0,0,0,total);
      end if;

select rmin ,rmax ,Costo_kw ,kw ,subtotal ,total,error from Tcasa1;
END $$

DELIMITER ;
```

la función KHW se explica con mas detalle en la sección 4.20.1. los bloques BLOCK1 y BLOCK2 se usan pon necesidad de programación ver sección .

4.2. Tarifa casa 1A

Esta tarifa es para Servicio doméstico de localidades con temperatura media mínima en verano de 25 grados centígrado, tiene costos dependiendo del rango de consumo y depende del mes en que inicia verano y los siguientes 5 meses siguientes en la región indicada como se observa a continuación para el el año 2017:

Temporada fuera de verano
Consumo básico $ 0.793 por cada uno de los primeros 75 (setenta y cinco)kilowatts-hora.
Consumo intermedio $ 0.956 por cada kilowatt-hora adicional a los anteriores.
Cargos por energía consumida
Consumo básico $ 0.793 por cada uno de los primeros 75 (setenta y cinco) kilowatts-hora.

Consumo intermedio $ 0.956 por cada uno de los siguientes 75 (setenta y cinco) kilowatts-hora.
Consumo excedente $ 2.802 por cada kilowatt-hora adicional a los anteriores.
Mínimo mensual El equivalente a 25 (veinticinco) kilowatts-hora.
Se pudo observar que los valores de los rangos de la tarifa no varían mes con mes por lo que se tuvo en cuenta para diseñar la tabla para manejo de la tarifa Casa 1A.

4.2.1. Diseño Tabla Tcasa_1A

En esta tabla, se diseñó para contener los rangos de KW(rmin, rmax) y el costo de KW en el rango estipulado por CFE por fuera de verano. La tabla diseñada se puede observar en la Figura **4-3**.

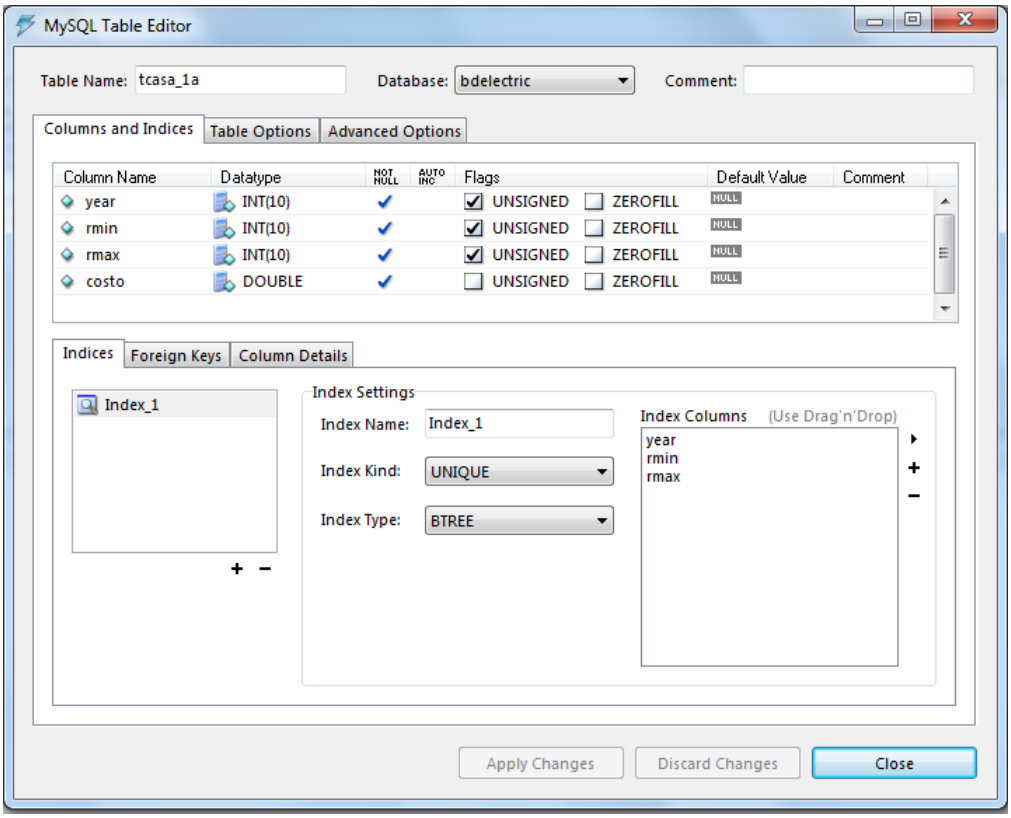


Figura 4-3.: Diseño tabla tarifa casa 1A fuera de verano.

4.2.2. Diseño Tabla Tcasa_1A_verano

En esta tabla, se diseñó para contener los rangos de KWH (rmin,rmax) y el costo de KHW en el rango estipulado por CFE para verano. La tabla diseñada se puede observar en la Figura **4-4**.

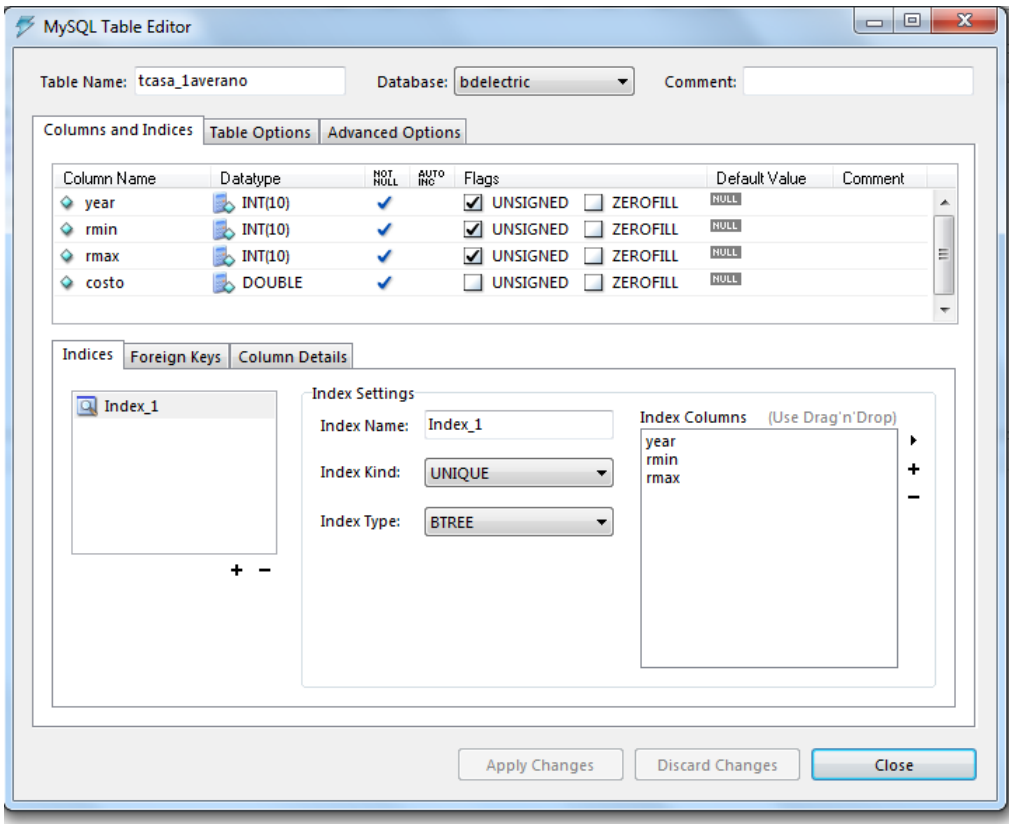


Figura 4-4.: Diseño tabla tarifa casa 1A Verano

El código SQL para creación de la tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tcasa_1averano`;
CREATE TABLE `bdelectric`.`tcasa_1averano` (
  `year` int(10) unsigned NOT NULL,
  `rmin` int(10) unsigned NOT NULL,
  `rmax` int(10) unsigned NOT NULL,
  `costo` double NOT NULL,
  UNIQUE KEY `Index_1` (`year`,`rmin`,`rmax`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Un ejemplo de dato en el archivo txt separado por comas para importación y actualización de datos en la tabla se observa en la Tabla **4-6**. Nótese que se colocó como rango máximo 10000kwh, ya que si la persona excede el consumo CFE lo cambiaria de tarifa como también se debe hacer un archivo tanto para verano (tabla tcasa_1A_Verano) como temporada fuera de verano (tcasa_1A)

Nota cabe notar que el rango mínimo debe ser 1, para que no afecte el cálculo en un KWH Por lo que el algoritmo de cálculo se observa se la Figura **4-5**. Donde fecha1 es la fecha de inicio del cálculo, fecha2 es la fecha de final de cálculo y Mes_verano es el número de mes

```
Ano,min,max,costo
2016,1,75,0.793
2016,76,140,0.956
2016,141,10000,2.802
```

Tabla 4-3.: Ejemplo de formato de datos de archivo .txt para actualización de tabla Tcasa_1A ó tabla tcasa_1A_Verano

donde inicia el verano en la localidad.

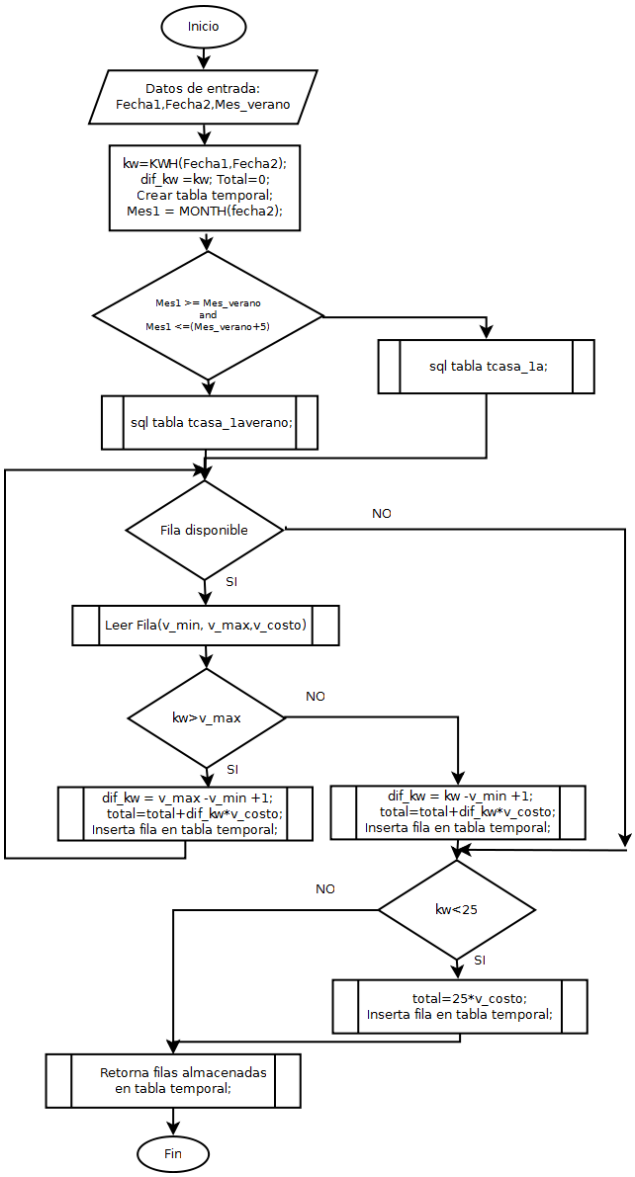


Figura 4-5.: Diagrama de flujo para cálculo de Tarifa Casa 1A.

El código SQL del procedimiento para cálculo de la tarifa Casa 1A se observa a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_tcasa_1A`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_tcasa_1A`(Fecha1 datetime,
Fecha2 datetime, Mes_verano SMALLINT)
BEGIN
  DECLARE total double;
  DECLARE  dif_kw double;
  DECLARE v_min SMALLINT;
  DECLARE v_max,Mes1 SMALLINT;
  DECLARE v_costo,kw double;
  DROP TABLE IF EXISTS Tcasa1a;
  CREATE TEMPORARY TABLE Tcasa1a(rmin SMALLINT,rmax SMALLINT,
  Costo_kw float,kw float,subtotal float,total float);

    set total =0;
    set kw = KWH(fecha1,Fecha2,2);
    if kw <0 then  set @error = kw; else set @error = 0; end if;
    set dif_kw =kw;
    set Mes1 = MONTH(fecha2);

  if (Mes1 >= Mes_verano)and (Mes1 <=(Mes_verano+5))  then

     BLOCK1: BEGIN
      DECLARE done_2 INT DEFAULT FALSE;
      DECLARE cursor_2 CURSOR FOR SELECT rmin,rmax,costo
      FROM tcasa_1averano where tcasa_1averano.year= year(Fecha2);
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done_2 = TRUE;
      OPEN cursor_2;
    read_loop_2: LOOP
        FETCH cursor_2 INTO v_min, v_max,v_costo;
        IF done_2 THEN LEAVE read_loop_2; END IF;

        if (kw >=v_max) then
          set dif_kw = v_max -v_min +1;
          set total=total+dif_kw*v_costo;
        else
          set dif_kw = kw -v_min+1;
```

```
          if dif_kw <=0 then
           LEAVE read_loop_2;
          end if;
         set total=total+dif_kw*v_costo;
        end if;
INSERT INTO Tcasa1a(rmin ,rmax ,Costo_kw ,kw ,subtotal ,total)
VALUES (v_min,v_max,v_costo,dif_kw,dif_kw*v_costo,total);
    END LOOP;
    CLOSE cursor_2;
    END  BLOCK1;
else

   BLOCK2: BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE cursor_i CURSOR FOR SELECT rmin,rmax,costo
   FROM tcasa_1a where tcasa_1a.year= year(Fecha2) ;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cursor_i;

 read_loop: LOOP
   FETCH cursor_i INTO v_min, v_max,v_costo;
     IF done THEN LEAVE read_loop;
     END IF;

   if (kw >=v_max) then
         set dif_kw = v_max -v_min +1;
         set total=total+dif_kw*v_costo;
      else
         set dif_kw = kw -v_min+1;
         if dif_kw <=0 then
          LEAVE read_loop;
          end if;
        set total=total+dif_kw*v_costo;
   end if;
   INSERT INTO Tcasa1a(rmin,rmax,Costo_kw,kw,subtotal,total)
   VALUES (v_min,v_max,v_costo,dif_kw,dif_kw*v_costo,total);
 END LOOP;
 CLOSE cursor_i;
 END  BLOCK2;
end if;
```

```
      if (kw <25) then
        set total=25*v_costo;
        INSERT INTO Tcasa1a(rmin,rmax,Costo_kw,kw,subtotal,total)
        VALUES (0,0,0,0,0,total);
      end if;
    select * ,@error from Tcasa1a;

END $$

DELIMITER ;
```

la función especial KWH se explica a detalle en la sección 4.20.1.

4.3. Tarifa casa 1B

Esta tarifa es para Servicio doméstico para localidades con temperatura media mínima en verano de 28 grados centígrado, tiene costos dependiendo del rango de consumo y depende del mes en que inicia verano en la región indicada como se observa a continuación para el año 2017.

Cargos por energía consumida
Consumo básico $ 0.793 por cada uno de los primeros 75 (setenta y cinco) kilowatts-hora.
Consumo intermedio $ 0.956 por cada uno de los siguientes 100 (cien) kilowatts-hora.

Consumo excedente $ 2.802 por cada kilowatt-hora adicional a los anteriores.

Mínimo mensual El equivalente a 25 (veinticinco) kilowatts-hora.

Se pudo observar que los valores de los rangos de la tarifa no varían mes con mes por lo que se tuvo en cuenta para diseñar la tabla para manejo de la tarifa Casa 1B.

4.3.1. Diseño de tablas Tcasa_1B, Tcasa_1Bverano

Como se puede observar a diferencia de la tarifas de casa 1A, solo varían en los de consumo, por lo que las tablas se diseñaron de igual manera una para consumo por rangos en verano y otra para fuera de este (ver Figuras **4-3**,**4-4**), la rutina de cálculo es igual a la casa 1A, solo que ahora se consultas las tablas Tcasa_1B y Tcasa_1Bverano.

4.4. Tarifa casa 1C, 1D, 1E, 1F

Esta tarifas son para Servicio doméstico, que la igual que la tarifa 1B, solo varían por la temperatura de la localidad (este no afecta en el cálculo solo para clasificación de CFE), rango de consumos y costos por rangos. Por lo que se resumen las tarifas se muestra en la tabla **4-7** y sus rangos, por lo que el diseño de las tablas que igual que la tarifa 1A, como la rutina de cálculo de consumos.

TARIFA	CONSUMO MINIMO KWH	COBRO RANGO NO VERANO CONSUMO KWH	COBRO RANGO EN VERANO CONSUMO KWH
1	25	DE 1 A 75 $0.793 DE 76 A 140 $0.956 MAS DE 141 $2.802	NO APLICA
1A	25	DE 1 A 75 $0.793 DE 76 A 150 $0.956 MAS DE 151 $2.802	DE 1 A 100 $0.697 DE 101 A 150 $0.822 MAS DE 151 $2.802
1B	25	DE 1 A75 $0.793 DE 76 A 150 $0.956 MAS DE 151 $2.802	DE 1 A 125 $0.697 DE 126 A 225 $0.822 MAS DE 226 $2.802
1C	25	DE 1 A 75 $0.793 DE 76 A 175 $0.956 MAS DE 176 $2.802	DE 1 A 150 $0.697 DE 151 A 300 $0.822 DE 301 A 450 $1.050 MAS DE 451 $2.802
1D	25	DE 1 A 75 $0.793 DE 76 A 200 $0.956 MAS DE 201 $2.802	DE 1 A 175 $0.697 DE 176 A 400 $0.822 DE 401 A 600 $1.050 MAS DE 601 $2.802
1E	25	DE 1 A 75 $0.793 DE 76 200 $0.956 MAS DE 201 $2.802	DE 1 A 300 $0.583 DE 301 A 750 $0.726 DE 751 A 900 $0.948 MAS DE 901 $2.802
1F	25	DE 1 A 75 $0.793 DE 76 A 200 $0.956 MAS DE 201 $2.802	DE 1 A 300 $0.583 DE 301 A 1200 $0.726 DE 1201 A 2500 $1.768 MAS DE 2501 $2.802

Tabla 4-4.: Tarifas para casas para el año 2016.

A continuación se muestra las lista de tarifas, nombre tablas asignadas, y nombre de rutina.

Tarifa	Nombre Tabla Rangos Fuera de Verano	Nombre Tabla Rangos Fuera en Verano	Nombre de procedimiento para cálculo de costos:
1A	Tcasa_1A	Tcasa_1Averano	Calc_casa_1a(Fecha1,Fecha2, Mes_verano)
1B	Tcasa_1B	Tcasa_1Bverano	Calc_casa_1b(Fecha1,Fecha2, Mes_verano)
1C	Tcasa_1C	Tcasa_1Cverano	Calc_casa_1c(Fecha1,Fecha2, Mes_verano)
1D	Tcasa_1D	Tcasa_1Dverano	Calc_casa_1d(Fecha1,Fecha2, Mes_verano)
1E	Tcasa_1E	Tcasa_1Everano	Calc_casa_1e(Fecha1,Fecha2, Mes_verano)
1F	Tcasa_1F	Tcasa_1Fverano	Calc_casa_1f(Fecha1,Fecha2, Mes_verano)

Tabla 4-5.: Tarifas de casas, nombre tablas asignadas, y nombre de rutina

4.5. Tarifa 5 Servicio para alumbrado público

Esta tarifa sólo se aplicará al suministro de energía eléctrica para el servicio a semáforos, alumbrado y alumbrado ornamental por temporadas, de calles, plazas, parques y jardines públicos. En las zonas conurbadas del Distrito Federal, Monterrey y Guadalajara, definiéndose éstas como las señaladas en la Segunda Resolución de la Secretaría de Hacienda y Crédito Público, que reforma y adiciona a la que establece reglas generales y otras disposiciones de carácter fiscal para el año de 1989, en su regla 81-A, y en al Cuarta Resolución que reforma, adiciona y deroga algunas disposiciones de la que establece reglas generales y otras disposiciones de carácter fiscal para el año de 1989, publicadas en el Diario Oficial de la Federación, los días 2 de mayo y 26 de junio de 1989, respectivamente.

Horario

Del anochecer al amanecer del día siguiente, excepto el servicio a semáforos; o el que se establezca en los convenios que en cada caso suscriban las partes contratantes.

Cuotas aplicables en el mes de Abril de 2017

Cargo por la energía consumida en los servicios suministrados en media tensión $ 3.206 por cada kilowatt-hora.

Cargo por la energía consumida en los servicios suministrados en baja tensión $ 3.814 por cada kilowatt-hora.

Mínimo mensual

La cantidad que resulte de aplicar las cuotas correspondientes al consumo equivalente a 4 horas diarias del servicio de la demanda contratada.

Consumo de energía Normalmente se medirán los consumos de energía, aunque en los contratos respectivos se establecerán el o los procedimientos para determinar el consumo de energía, de acuerdo con las características en que se efectúe el suministro de servicio y de conformidad con las normas aplicables.

Demanda por contratar La demanda por contratar corresponderá al 100 % de la carga conectada. Cualquier fracción de kilowatt se tomará como kilowatt completo.

Reposición de lámparas

El prestador del servicio deberá reponer las lámparas, los aparatos y materiales accesorios que requiera la operación de las mismas. Tratándose de alumbrado público, cuando el suministrador esté de acuerdo en tomar a su cargo la reposición de las lámparas y dispositivos necesarios, se fijará en los contratos la forma para el cobro de los gastos que origine este servicio adicional al del suministro de energía.

4.5.1. Diseño de Tabla Tnegocio_5

Se procedió a diseñar tabla Tnegocio_5 para la tarifa correspondiente, para que soporte tipo de tensión, valor de la tarifa y actualización como se observa a continuación

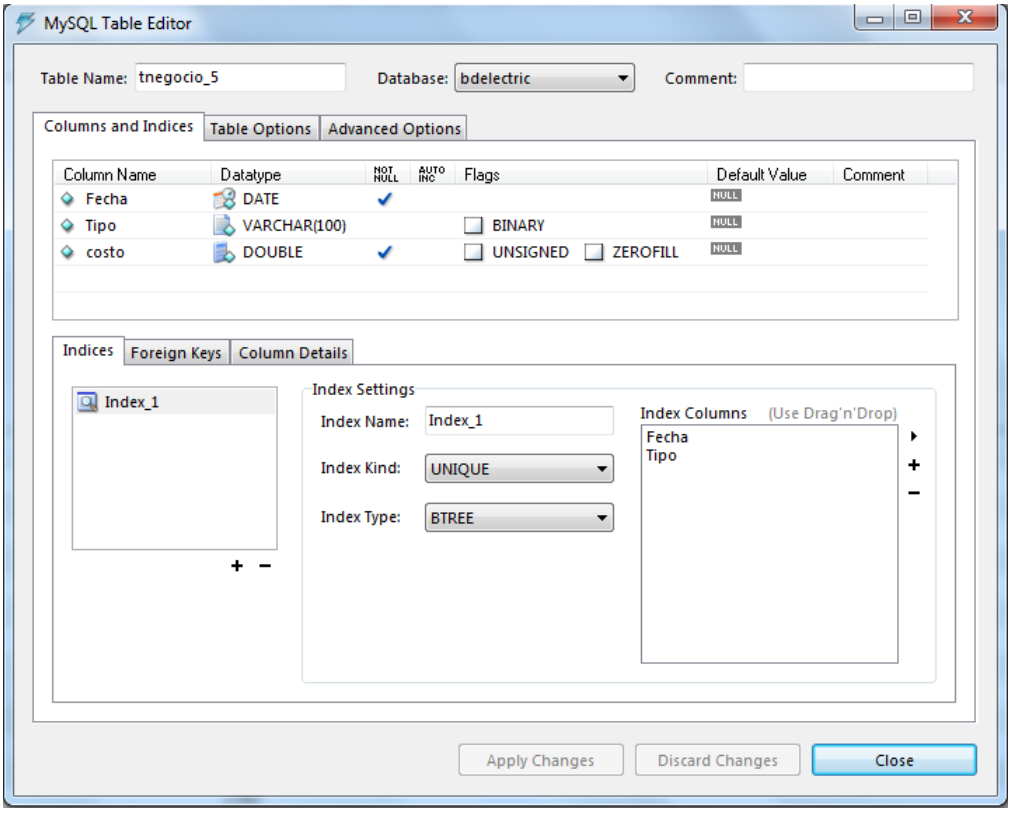


Figura 4-6.: Diseño de tabla Tnegocio_5.

El código SQL para creación de la tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_5`;
CREATE TABLE  `bdelectric`.`tnegocio_5` (
  `Fecha` date NOT NULL,
  `Tipo` varchar(100) DEFAULT NULL,
  `costo` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`,`Tipo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El diagrama de flujo del cálculo de la tarifa se puede observar en la Figura **4-7**.

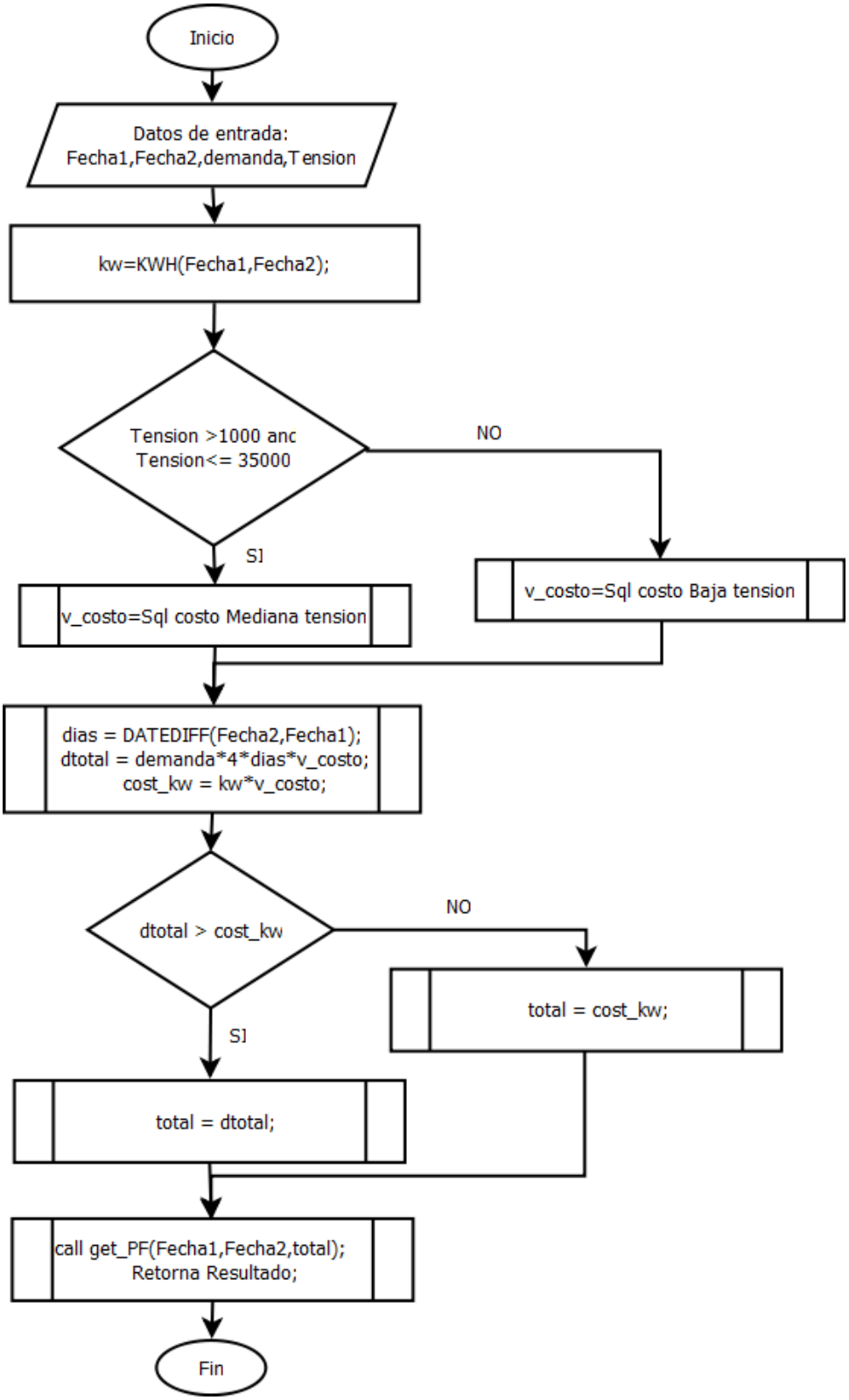


Figura 4-7.: Diagrama de flujo Tarifa negocio 5.

El código SQL para cálculo de tarifa se puede observar a continuación:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_5`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_5`(Fecha1 datetime,Fecha2 datetime,
 demanda double,tension int)
BEGIN
  DECLARE total double;
  DECLARE  dtotal double;
  DECLARE v_costo double;
  DECLARE cost_kw double;
  DECLARE dias,kw double;
  DECLARE ayuda  varchar(200);

  if tension >1000 and  tension <=35000 then
     SELECT costo into v_costo FROM tnegocio_5 where
     month(Fecha)=month(Fecha2) and year(Fecha)=year(Fecha2)
     and Tipo like 'Media_tension' limit 1;
   else
     SELECT costo into v_costo FROM tnegocio_5 where
     month(Fecha)=month(Fecha2) and year(Fecha)=year(Fecha2)
     and Tipo like 'Baja_Tension' limit 1;
  end if;

   IF(v_costo IS NULL) THEN
        set @error =-8005;
    else
         set @error =0;
   END IF;

  set total =0;
  set dtotal =0;
  set cost_kw =0;
  set kw = KWH(fecha1,Fecha2,2);
   if kw <0 then  set @error = kw; end if;
  set dias = DATEDIFF(Fecha2,Fecha1);
  set ayuda = 'Minimo_consumo_demanda*4*dias*v_costo;_';
    set dtotal = demanda*4*dias*v_costo;
    set cost_kw = kw*v_costo;
     if (dtotal > cost_kw) then
```

```
      set total = dtotal;
    else
      set total = cost_kw;
    end if;

  call get_PF(Fecha1,Fecha2,total);
  select kw,v_costo,demanda,dias,total,ayuda,t.KWHA,
  t.Factor_Potencia,t.Bonificacion,t.cargo,
  t.cargo_bajatension,t.total,@error from T_FP t;

END $$
DELIMITER ;
```

el procedimiento de almacenado get_PF se describe en la sección de rutinas especiales para este proyecto ver seccion 4.20, get_PF en especial en 4.20.3.

4.6. Tarifa Negocio 5A

Esta tarifa sólo se aplicará al suministro de energía eléctrica para el servicio a semáforos, alumbrado y alumbrado ornamental por temporadas, de calles, plazas, parques y jardines públicos en todo el país exceptuándose las circunscripciones para las cuales rige la tarifa 5. El diseño de la tabla y rutina de cálculo es igual a la tarifa negocio 5.
A continuación se observa ejemplo de archivo .txt separado por comas para importación y actualización de datos en la base de datos.

```
Fecha,tipo tension, Valor minimo 4 horas diarias
2016-06-01,Media Tension,2.525
2016-06-01,Baja Tension,3.003
2016-07-01,Media Tension,2.537
2016-07-01,Baja Tension,3.018
2016-08-01,Media Tension,2.549
2016-08-01,Baja Tension,3.033
2016-09-01,Media Tension,2.561
2016-09-01,Baja Tension,3.048
2016-10-01,Media Tension,2.573
2016-10-01,Baja Tension,3.063
2016-11-01,Media Tension,2.585
2016-11-01,Baja Tension, 3.078
2016-12-01,Media Tension,2.597
2016-12-01,Baja Tension, 3.093
```

4.7. Tarifa Negocio 6

Esta tarifa es para servicio para bombeo de aguas potables o negras, de servicio público

Aplicación
Esta tarifa se aplicará al suministro de energía eléctrica para servicio público de bombeo de aguas potables o negras.
Cuotas aplicables en el mes de Abril de 2017.
Cargo fijo, independiente de la energía consumida $ 361.91
Cargo adicional por la energía consumida $ 1.984 por cada kilowatt-hora.
Mínimo mensual
Cuando el usuario no haga uso del servicio, cubrirá como mínimo al cargo fijo.
Demanda por contratar La demanda por contratar la fijará inicialmente el usuario; su valor no será menor de 60 % de la carga total conectada ni menor de la capacidad del mayor motor o aparato instalado. Cualquier fracción de kilowatt se tomará como kilowatt completo.
Depósito de garantía
Será de 4 veces el mínimo mensual aplicable.

4.7.1. Diseño de Tabla TNegocio_6

La tabla diseñada se puede observar en la Figura **4-8**.

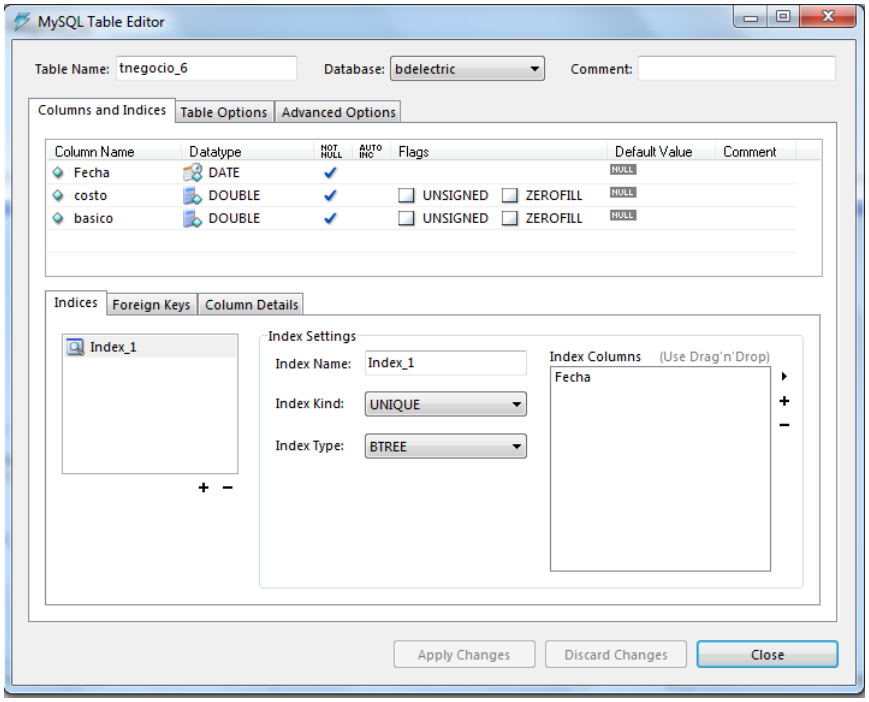


Figura 4-8.: Diseño de tabla Tnegocio_6.

El código SQL para la creación de la tabla es el siguiente.

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_6`;
CREATE TABLE `bdelectric`.`tnegocio_6` (
  `Fecha` date NOT NULL,
  `costo` double NOT NULL,
  `basico` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Un ejemplo de un archivo de texto separado por comas para actualizar los datos se observa a continuación:

```
Fecha,costo,basico  costo= basico+consumoxcosto
2016-07-01,1.902,346.56
2016-08-01,1.911,348.23
2016-09-01,1.920,349.91
2016-10-01,1.929,351.60
2016-11-01,1.938,353.30
2016-12-01,1.947,355.01
```

El algoritmo de cálculo de tarifa se puede observar en la Figura **4-9**.

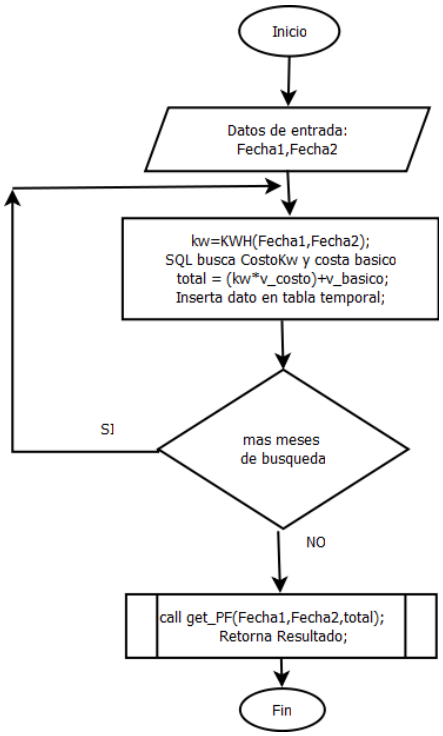


Figura 4-9.: Diagrama de flujo para cálculo de tarifa Negocio 6.

El código SQL para el cálculo de la tarifa Negocio 6 se observa a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_6`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_6`(Fecha1 datetime,Fecha2 datetime)
BEGIN
  DECLARE total double;
  DECLARE v_basico double;
  DECLARE v_costo double;
  DECLARE cost_kw,kw double;
  DECLARE fin ,bandera smallint;
  DECLARE  f1,fechasql  date;
  -- crea tabla temporal para resultados del reporte
  DROP TABLE IF EXISTS TReporte;
  CREATE TEMPORARY TABLE TReporte(fecha date,kw float,
  vcosto double, basico double, pptotal double);
  set total =0;
   -- configuracion bucle
  set fin=0;
  set fechasql = LAST_DAY(Fecha1);
  set f1 = date(Fecha1);
  set total =0;

  select if(fechasql > Fecha2,1,0) into bandera;
  if (bandera =1) then
     select  date(Fecha2) into fechasql ;
  end if;
--  fin configuracion de bucle
  WHILE (fin =0 ) DO
  SELECT costo,basico into v_costo,v_basico FROM tnegocio_6 where
  month(Fecha)=month(fechasql) and year(Fecha)=year(fechasql) ;
  IF(v_costo IS NULL) THEN
        set @error =-8008;
  else
        set @error =0;
  END IF;

  set kw = KWH(fecha1,Fecha2,2);
  if kw <0 then  set @error = kw; end if;
```

```
  set total = (kw*v_costo)+v_basico;
  insert into TReporte(fecha,kw,vcosto,basico,pptotal)
  values(f1,kw,v_costo,v_basico,total);

    -- setup fin bucle y proxima iteracion
  select  fechasql + interval 1 day into f1;
  select if(f1 > Fecha2,1,0) into fin;
  set fechasql = LAST_DAY(f1);

   select if(fechasql > Fecha2,1,0) into bandera;
  if (bandera =1) then
     select  date(Fecha2) into fechasql ;
  end if;
  END WHILE;
  call get_PF(Fecha1,Fecha2,total);
  select r.fecha,r.kw,r.vcosto,r.basico,r.pptotal,fp.KWHA,
  fp.Factor_Potencia ,fp.Bonificacion ,fp.cargo,
  fp.cargo_bajatension ,fp.total,@error from T_FP fp,TReporte r;

END $$
DELIMITER ;
```

4.8. Tarifa Negocio 9

Esta tarifa es para servicio para bombeo de agua para riego agrícola en baja tensión.

Aplicación
Esta tarifa se aplicará exclusivamente a los servicios en baja tensión que destinen la energía para el bombeo de agua utilizada en el riego de tierras dedicadas al cultivo de productos agrícolas y al alumbrado del local donde se encuentre instalado el equipo de bombeo.

Cuotas aplicables en el mes de Abril de 2017.
Cargo por la energía consumida
$ 8.833 por cada uno de los primeros 5,000 (cinco mil) kilowatts-hora.
$ 9.832 por cada uno de los siguientes 10,000 (diez mil) kilowatts-hora.
$ 10.730 por cada uno de los siguientes 20,000 (veinte mil) kilowatts-hora.
$ 11.778 por cada kilowatt-hora adicional a los anteriores.

Tensión y capacidad de suministro

El suministrador sólo está obligado a proporcionar el servicio a la tensión y capacidad disponibles en el punto de entrega.

Demanda por contratar
La demanda por contratar la fijará inicialmente el usuario; su valor no será menor de 60 % de la carga total conectada, ni menor de la capacidad del mayor motor o aparato instalado. Cualquier fracción de kilowatt se tomará como kilowatt completo.

4.8.1. Diseño Tabla TNegocio_9

En base a la normatividad expuesta se diseñó la tabla correspondiente como se observa en la Figura **4-10**. Donde se toma en cuenta el rango de precios de acuerdo el consumo y las actualizaciones futuras de la tarifa.

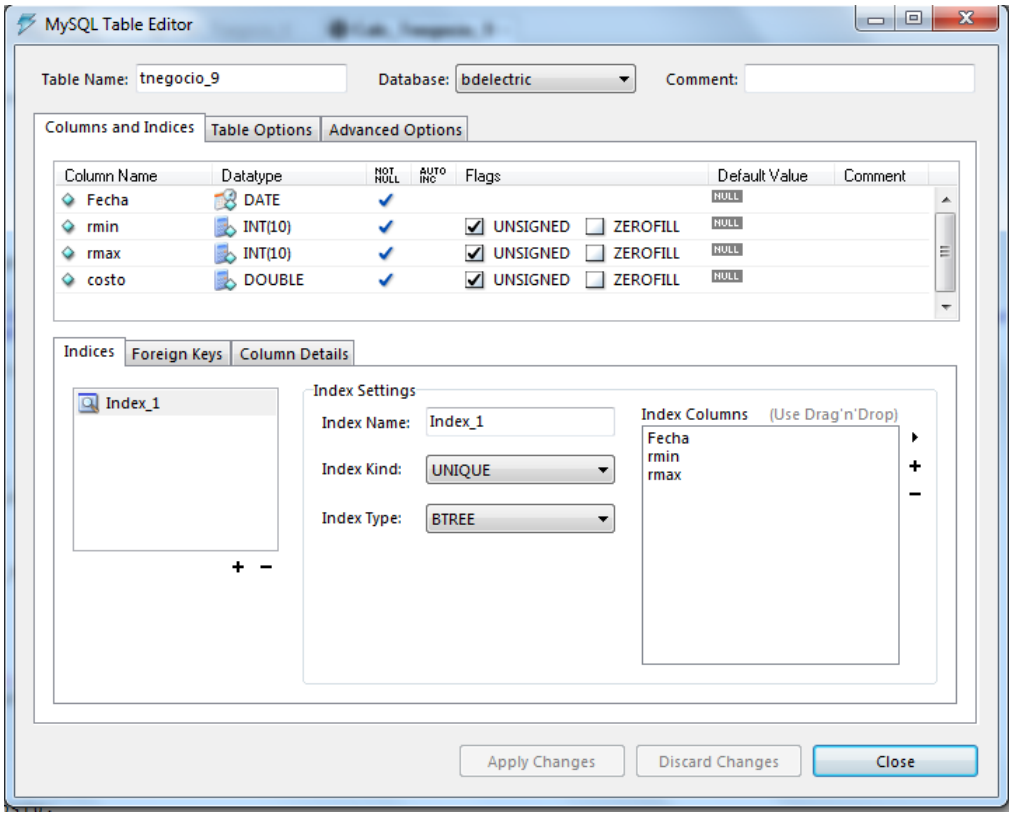


Figura 4-10.: Diseño tabla Tnegocio_9

El código SQL para la creación de tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_9`;
CREATE TABLE  `bdelectric`.`tnegocio_9` (
  `Fecha` date NOT NULL,
```

```
  `rmin` int(10) unsigned NOT NULL,
  `rmax` int(10) unsigned NOT NULL,
  `costo` double unsigned NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`,`rmin`,`rmax`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El algoritmo de cálculo es similar a las tarifas de casa que están determinada por un rango de consumo. Un ejemplo de archivo .txt para importación de datos a la base de datos se observa a continuación: donde el campo fecha indica la fecha a partir empieza a regir la tarifa por esto deben siempre empezar por el día 01.

```
Fecha,min,max,costo
2016-08-01,1,5000,7.539
2016-08-01,5001,15000,8.391
2016-08-01,15001,35000,9.159
2016-08-01,35001,150000,10.053
2016-09-01,1,5000,7.690
2016-09-01,5001,15000,8.559
2016-09-01,15001,35000,9.342
2016-09-01,35001,150000,10.254
2016-10-01,1,5000,7.844
2016-10-01,5001,15000,8.730
2016-10-01,15001,35000,9.529
2016-10-01,35001,150000,10.459
```

4.9. Tarifa Negocio 9M

Esta tarifa es similar a la tarifa 9, pero con rangos distintos y operación en mediana tensión.

4.10. Tarifa Negocio 9-CU

Tarifa de estímulo para bombeo de agua para riego agrícola con cargo único

Aplicación
Esta tarifa de estímulo se aplicará para la energía eléctrica utilizada en la operación de los equipos de bombeo y rebombeo de agua para riego agrícola por los sujetos productivos inscritos en el padrón de beneficiarios de energéticos agropecuarios, hasta por la Cuota Energética determinada por la Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación.

Cuotas aplicables
Durante todo este año, se aplicará el siguiente cargo por la energía consumida, hasta por la Cuota Energética:
Ejemplo de cargo Mes de abril de 2017
Año Cargo por kilowatt-hora de energía consumida 2017 $0.580

Energía excedente
La energía eléctrica consumida que exceda la Cuota Energética, será facturada con los cargos de la Tarifa 9 o 9M, Servicio para Bombeo de Agua para Riego Agrícola en Baja o Media Tensión, según corresponda.

Tensión y capacidad de suministro
El suministrador sólo está obligado a proporcionar el servicio a la tensión y capacidad disponibles en el punto de entrega.

Demanda contratada
La Demanda Contratada la fijará inicialmente el usuario, y su valor no será menor a la carga total conectada. Cualquier fracción de kilowatt se tomará como kilowatt completo.

4.11. Tarifa Negocio 9N

Tarifa de estímulo nocturna para bombeo de agua para riego agrícola.

Aplicación
Esta tarifa de estímulo nocturna se aplicará para la energía eléctrica utilizada en la operación de los equipos de bombeo y rebombeo de agua para riego agrícola por los sujetos productivos inscritos en el padrón de beneficiarios de energéticos agropecuarios, hasta por la Cuota Energética determinada por la Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación. La inscripción a esta tarifa será a solicitud del usuario.

Cuotas aplicables
Durante todo este año, se aplicarán los siguientes cargos por la energía consumida en periodo nocturno y en periodo diurno, hasta por la Cuota Energética:
Ejemplo de tarifa el mes de abril de 2017
Año Cargo por kilowatt-hora de energía consumida en el periodo diurno Cargo por kilowatt-hora de energía consumida en el periodo nocturno 2017 $0.580

Energía excedente
La energía eléctrica consumida que exceda la Cuota Energética, será facturada con los cargos de la Tarifa 9 o 9M, Servicio para Bombeo de Agua para Riego Agrícola en Baja o Media

Tensión, según corresponda. Para los efectos del párrafo anterior, en caso de que durante algunos meses del año calendario el usuario haya recibido el servicio con la tarifa 9-CU, Tarifa de Estímulo para Bombeo de Agua para Riego Agrícola con Cargo Unico, la energía eléctrica facturada con el cargo del numeral 2 de dicha tarifa se agregará a la contabilizada con la tarifa 9-N.

Periodo nocturno y periodo diurno
El periodo nocturno comprenderá de las 0:00 horas a las 08:00 horas y será aplicable todos los días. El periodo diurno comprenderá de las 08:00 horas a las 24:00 horas y será aplicable todos los días. Para los efectos de la aplicación de esta tarifa, se utilizarán los horarios locales oficialmente establecidos.

Tensión y capacidad de suministro
El suministrador sólo está obligado a proporcionar el servicio a la tensión y capacidad disponibles en el punto de entrega.

Demanda contratada
La Demanda Contratada la fijará inicialmente el usuario, y su valor no será menor de la carga total conectada. Cualquier fracción de kilowatt se tomará como kilowatt completo.

Esta tarifa no se implementó debido a que son tarifas especiales de estímulo.

4.12. Tarifa Negocio 7

Esta tarifa es para servicio temporal.

Aplicación
Esta tarifa se aplicará a todos los servicios que destinen la energía temporalmente a cualquier uso, exclusivamente donde y cuando la capacidad de las instalaciones del suministrador lo permitan y éste tenga líneas de distribución adecuadas para dar el servicio.

Horario
Lo convenido en cada caso entre el suministrador y el usuario, el que no deberá hacer uso del servicio fuera del horario estipulado.
Cuotas aplicables en el mes de abril de 2017.

Cargo por demanda
$ 189.37 por cada kilowatt de demanda.
Cargo adicional por la energía consumida
$ 5.423 por cada kilowatt-hora.

Contratación del servicio y determinación de la energía eléctrica
Los contratos se celebrarán por el número de días consecutivos por los que el usuario quiera disponer del servicio. Ningún servicio temporal podrá tener una vigencia mayor de 30 días excepto en los casos de personas o negociaciones que utilicen máquinas de pulir, encerar y lavar pisos, pintar y soldar, etc., cuya vigencia puede ser por un plazo mayor. El computo de la demanda y el consumo se hará de acuerdo con la carga de aparatos instalados y el número de horas que se use el servicio, el que en ningún caso será menor de 4 horas diarias, teniendo el suministrador derecho de verificar en cualquier tiempo la carga individual y el consumo de cada uno de los aparatos instalados.

4.12.1. Diseño de tabla Tnegocio_7

El diseño de la tabla para la tarifa negocio 7 se observa en la figura **4-11**.

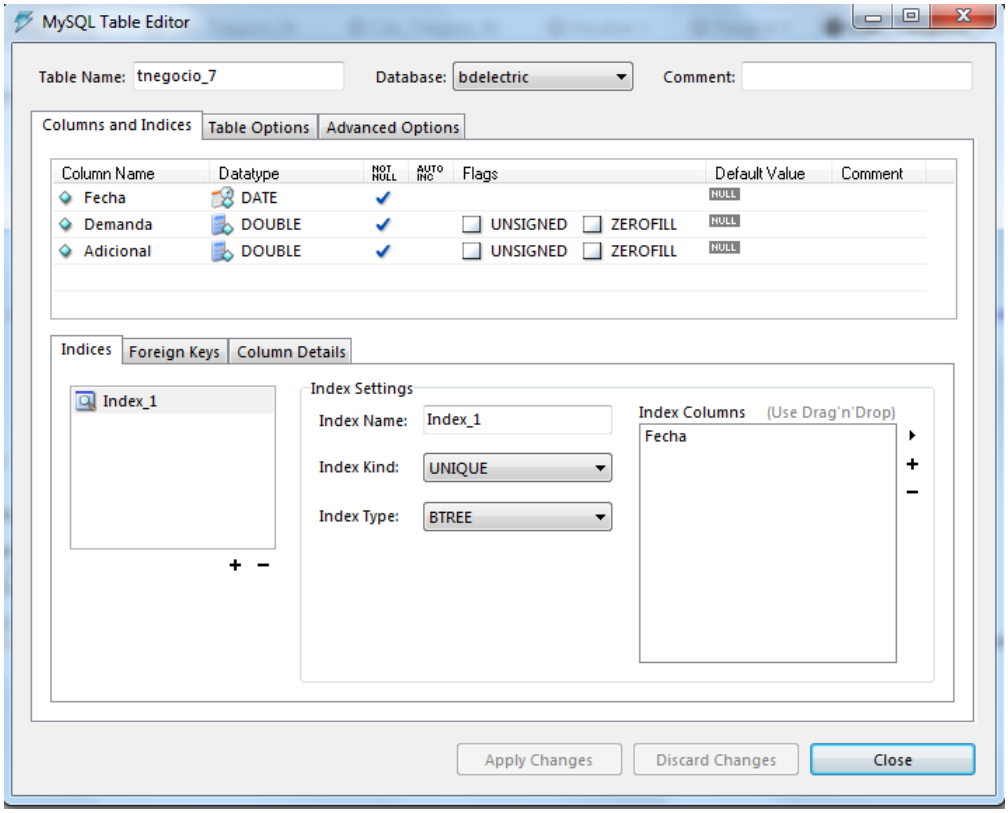


Figura 4-11.: Diseño de tabla tnegocio_7.

El respectivo código SQL para la creación de esta tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_7`;
CREATE TABLE `bdelectric`.`tnegocio_7` (
  `Fecha` date NOT NULL,
  `Demanda` double NOT NULL,
  `Adicional` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El diagrama de flujo para cálculo de la tarifa se observa en la Figura **4-12**.

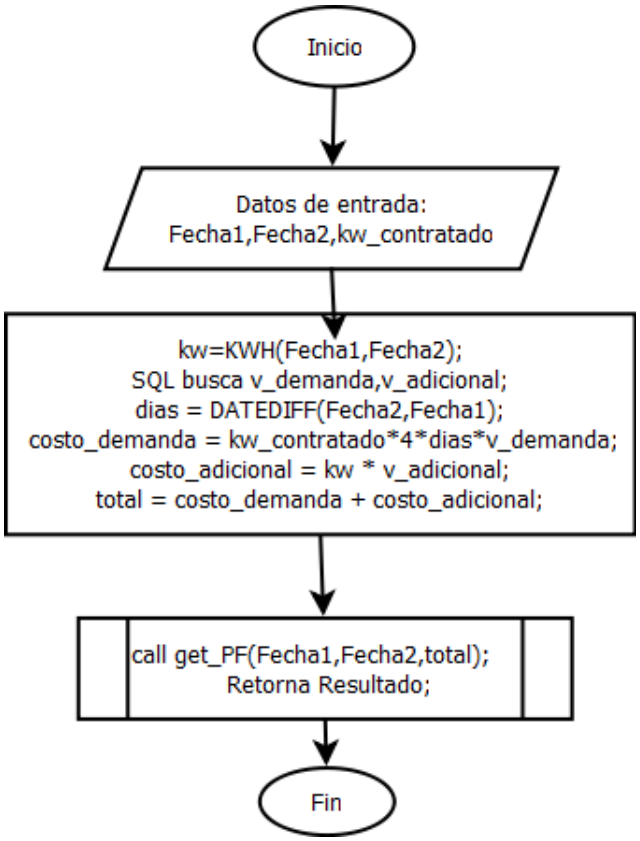


Figura 4-12.: Diagrama de flujo de cálculo de tarifa negocio 7.

El código SQL para cálculo de tarifa se observa a continuación:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_7`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_7`(Fecha1 datetime,Fecha2 datetime,
kw_contratado double)
BEGIN
```

```
  DECLARE total double;
  DECLARE v_demanda double;
  DECLARE v_adicional double;
  DECLARE costo_demanda double;
  DECLARE dias,costo_adicional,kw double;
  set total =0;
  SELECT Demanda,Adicional into v_demanda,v_adicional
  FROM tnegocio_7 where month(Fecha)=month(Fecha2)
  and year(Fecha)=year(Fecha2);
  IF(v_demanda IS NULL) THEN
        set @error =-8009;
  else
        set @error =0;
  END IF;
  set kw = KWH(fecha1,Fecha2,2);
  if kw <0 then  set @error = kw; end if;
  set dias = DATEDIFF(Fecha2,Fecha1);
    set costo_demanda = kw_contratado*4*dias*v_demanda;
    set costo_adicional = kw * v_adicional;
    set total = costo_demanda + costo_adicional;

  call get_PF(Fecha1,Fecha2,total);
  select  kw,v_adicional as 'costo_kw_adicional',kw_contratado,
  dias,costo_demanda,costo_adicional,total as 'ptotal',
  t.KWHA,t.Factor_Potencia ,t.Bonificacion,t.cargo,
  t.cargo_bajatension ,t.total,@error from T_FP t;

END $$
DELIMITER ;
```

4.13. Tarifa Negocio 2

Esta tarifa es para servicio general hasta 25 kW de demanda

Aplicación
Esta tarifa se aplicará a todos los servicios que destinen la energía en baja tensión a cualquier uso, con demanda hasta de 25 kilowatts, excepto a los servicios para los cuales se fija específicamente su tarifa.
Cuotas aplicables en el mes de Abril de 2017.

Cargo fijo $ 66.40

Cargos adicionales por energía consumida
$ 2.743 por cada uno de los primeros 50 kilowatts-hora.
$ 3.307 por cada uno de los siguientes 50 kilowatts-hora.
$ 3.645 por cada kilowatt-hora adicional a los anteriores.

Mínimo mensual
Cuando el usuario no haga uso del servicio cubrirá como mínimo el cargo fijo a que se refiere el punto 2 de esta tarifa.

Demanda por contratar
La demanda por contratar la fijará inicialmente el usuario con base en sus necesidades de potencia. Cualquier fracción de kilowatt se considerará como kilowatt completo. Cuando el usuario exceda la demanda de 25 kilowatts, deberá solicitar al suministrador aplique la tarifa 3. De no hacerlo, a la tercera medición consecutiva en que exceda la demanda de 25 kilowatts, será reclasificado por el suministrador, notificándole al usuario.

Depósito de garantía
Es el importe que resulte de aplicar el cargo adicional por energía consumida. Los consumos mensuales que se indican según los casos:
a) 125 kilowatts-hora para los servicios suministrados con 1 hilo de corriente.
b) 350 kilowatts-hora para los servicios suministrados con 2 hilos de corriente.
c) 400 kilowatts-hora para los servicios suministrados con 3 hilos de corriente.

4.13.1. Diseño Tabla TNegocio_2

Como se puede observar esta tarifa es similar a tarifas de casa con un cargo básico, por lo que la tabla diseñada queda como se observa en la figura **4-13**.

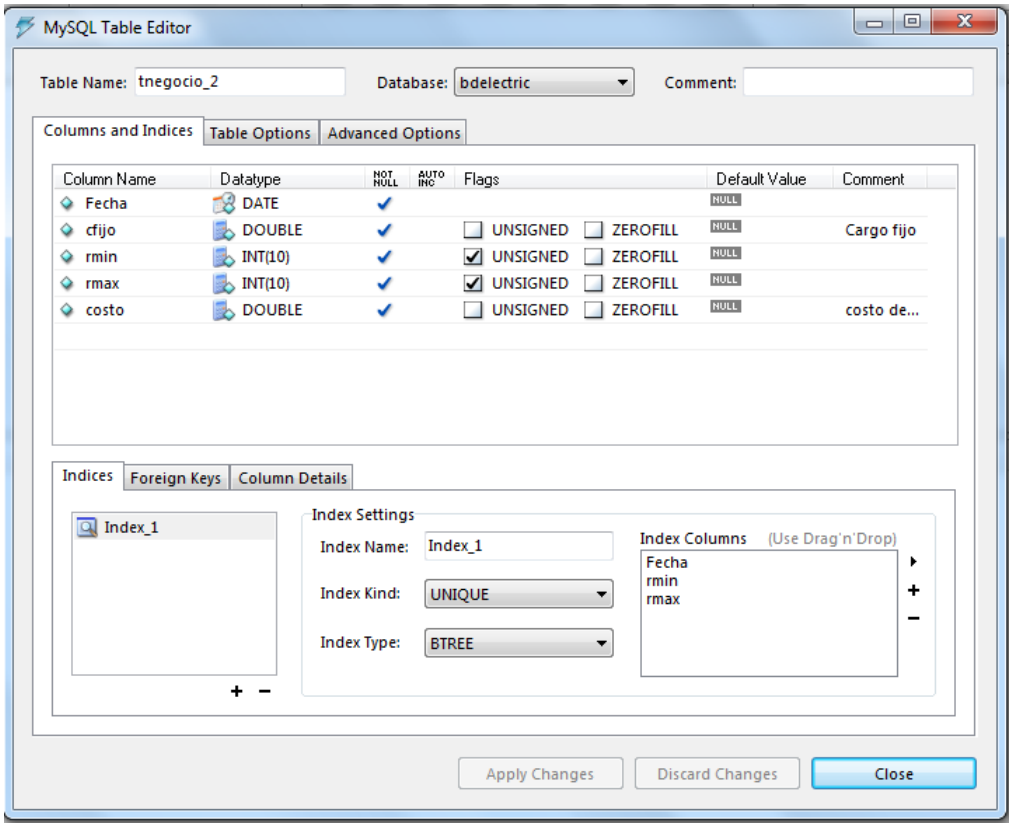


Figura 4-13.: Diseño tabla negocio 2.

El código SQL para crear la tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_2`;
CREATE TABLE  `bdelectric`.`tnegocio_2` (
  `Fecha` date NOT NULL,
  `cfijo` double NOT NULL COMMENT 'Cargo_fijo',
  `rmin` int(10) unsigned NOT NULL,
  `rmax` int(10) unsigned NOT NULL,
  `costo` double NOT NULL COMMENT 'costo_del_rango',
  UNIQUE KEY `Index_1` (`Fecha`,`rmin`,`rmax`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El código del procedimiento para calcular esta tarifa es similar a las tarifas de casas el código SQL se puedo observar a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_2`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_2`(Fecha1 datetime,Fecha2 datetime)
```

```
BEGIN
  DECLARE total double;
  DECLARE  dif_kw ,kw double;
  DECLARE v_min INT;
  DECLARE v_max int;
  DECLARE v_cfijo double;
  DECLARE v_costo double;
  DECLARE FP, Bonificacion_fp , Cargo_fp , t_total ,KWHA, cargobt double;
  DECLARE done INT DEFAULT FALSE;
  DECLARE cursor_i CURSOR FOR SELECT cfijo ,rmin ,rmax, costo
  FROM tnegocio_2 where month(Fecha)=month(Fecha2)
  and year(Fecha)=year(Fecha2);
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  DROP TABLE IF EXISTS Tcasa1;
  CREATE TEMPORARY TABLE Tcasa1(rmin SMALLINT,rmax SMALLINT,
  Costo_kw float ,kw float , subtotal float , total float );
  OPEN cursor_i;
  set total =0;
  set v_cfijo =1;
  set kw = KWH(fecha1 ,Fecha2 ,2);
  if kw <0 then  set @error = kw; else set @error = 0; end if;
  call Get_error('tnegocio_2',Fecha2,-8007);
  set dif_kw =kw;
  read_loop: LOOP
    FETCH cursor_i INTO v_cfijo ,v_min, v_max, v_costo;
    IF done THEN
      LEAVE read_loop;
    END IF;

   if (kw >=v_max) then
     set dif_kw = v_max -v_min +1;
     set total=total+dif_kw*v_costo;
   else
     set dif_kw = kw -v_min+1;
        if dif_kw <=0 then
       LEAVE read_loop;
        end if;
      set total=total+dif_kw*v_costo;

   end if;
```

```
    INSERT INTO Tcasa1(rmin ,rmax ,Costo_kw ,kw ,subtotal ,total)
    VALUES (v_min,v_max,v_costo,dif_kw,dif_kw*v_costo,total);
  END LOOP;
  CLOSE cursor_i;

    if (total <v_cfijo) then
     set total=v_cfijo;
     INSERT INTO Tcasa1(rmin ,rmax ,Costo_kw ,kw ,subtotal ,total)
     VALUES (0,0,0,0,0,total);
    end if;

    call get_PF(Fecha1,Fecha2,total); -- Busca factor de potencia del period
     -- y aplica cargos o bonificaciones
   select t.Factor_Potencia ,t.Bonificacion ,t.cargo ,t.total
   ,t.KWHA,t.cargo_bajatension into FP,Bonificacion_fp,Cargo_fp
   ,t_total,KWHA,cargobt from T_FP t;
   select t.rmin ,t.rmax ,t.Costo_kw ,t.kw ,t.subtotal
   ,t.total as 'ptotal',KWHA,FP,Bonificacion_fp,Cargo_fp,
   cargobt,t_total,@error from Tcasa1 t;

END $$
DELIMITER ;
```

la función get_PF se explica con mas detalle en la sección 4.20.3.

4.14. Tarifa Negocio 3

Esta tarifa es para servicio general para más de 25 kW de demanda.

Aplicación
Esta tarifa se aplicará a todos los servicios que destinen la energía en baja tensión a cualquier uso, con demanda de más de 25 kilowatts, excepto a los servicios para los cuales se fija específicamente su tarifa.

Cuotas aplicables en el mes de Abril de 2017.

Cargo por demanda máxima
$ 301.56 por cada kilowwat de demanda máxima medida
Cargo adicional por la energía consumida
$ 2.014 por cada kilowatt-hora.

Mínimo mensual
El importe que resulte de aplicar 8 veces el cargo por kilowatt de demanda máxima.

Demanda por contratar
La demanda por contratar la fijará inicialmente el usuario; su valor no será menor de 60 % de la carga total conectada ni menor de 25 kilowatts o de la capacidad del mayor motor o aparato instalado. Cualquier fracción de kilowatt se tomará como kilowatt completo.

Demanda máxima medida
La demanda máxima medida se determinará mensualmente por medio de instrumentos de medición que indiquen la demanda media en kilowatts durante cualquier intervalo de 15 minutos, en el cual el consumo de energía eléctrica sea mayor que en cualquier otro intervalo de 15 minutos en el período de facturación.

4.14.1. Diseño de Tabla TNegocio_3

La tabla diseña se observa en la figura **4-14**.

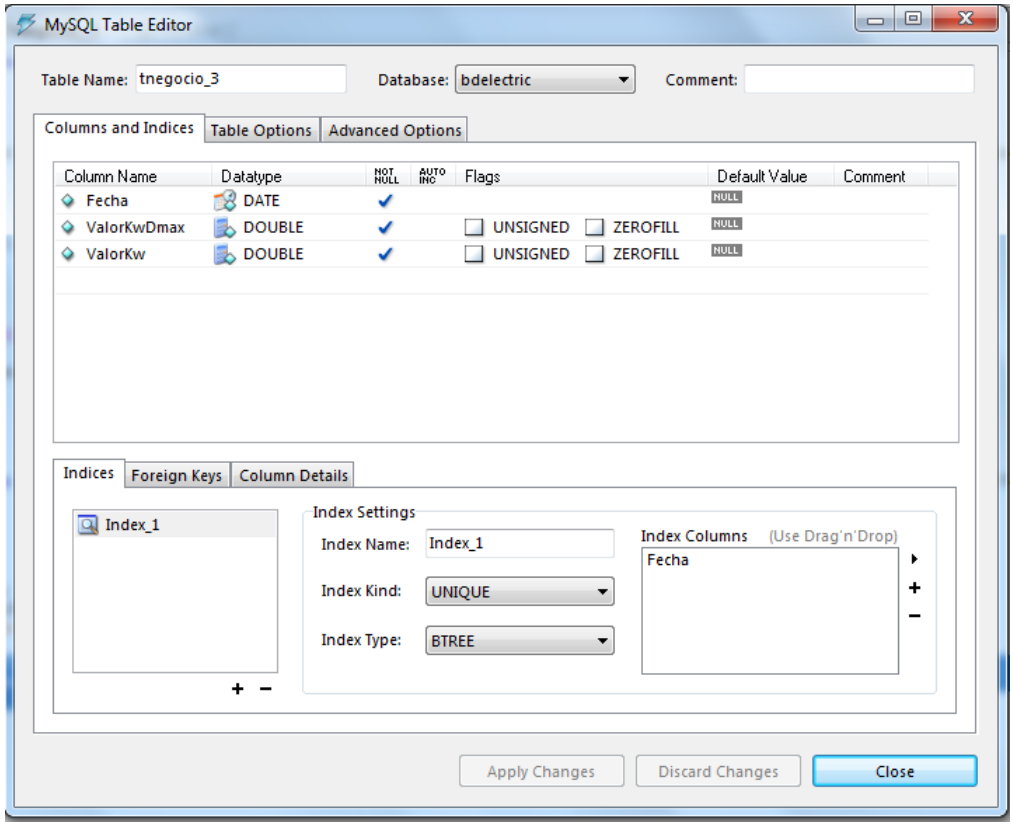


Figura 4-14.: Diseño tabla Negocio 3.

El código SQL para creación de la tabla se observa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_3`;
CREATE TABLE  `bdelectric`.`tnegocio_3` (
  `Fecha` date NOT NULL,
  `ValorKwDmax` double NOT NULL,
  `ValorKw` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El diagrama de flujo para cálculo de la tarifa se puede observar en la Figura **4-15**.

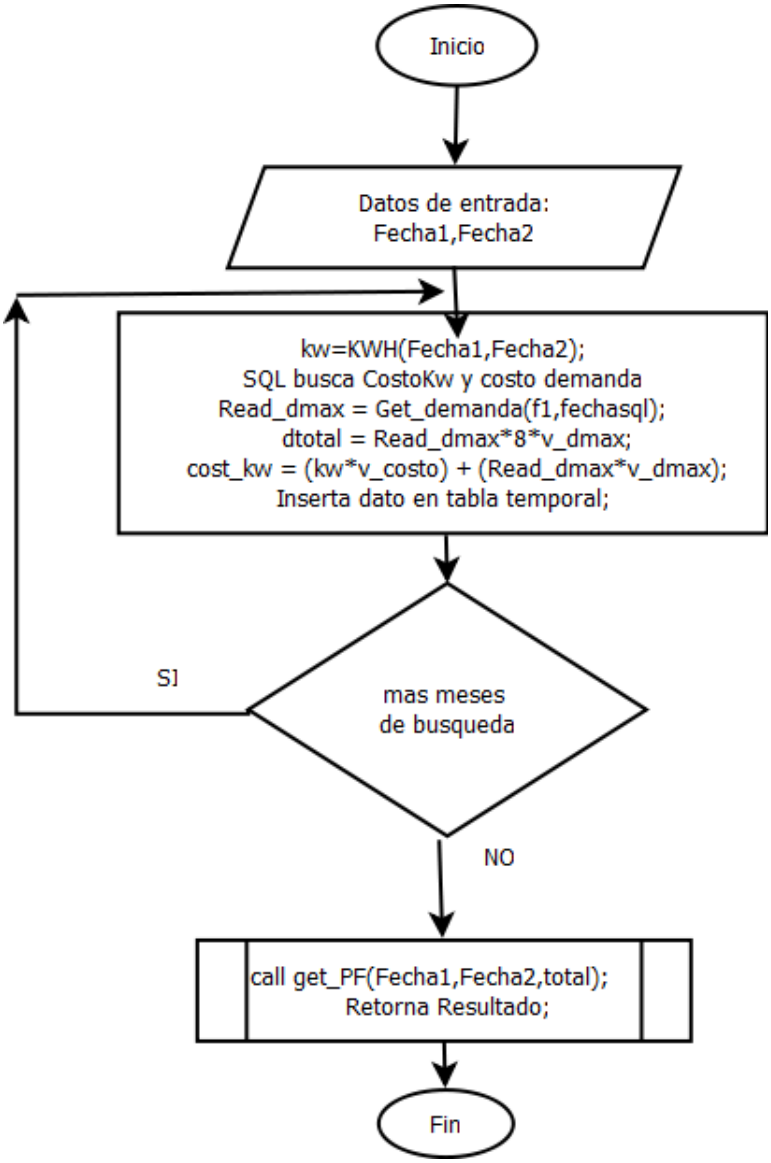


Figura 4-15.: Diagrama de flujo de cálculo de tarifa Negocio 3.

El código SQL del procedimiento de cálculo se observa a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_3`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_3`(Fecha1 datetime,Fecha2 datetime)
BEGIN
  DECLARE total double;
  DECLARE  dtotal double;
  DECLARE v_costo double;
  DECLARE v_dmax double;
  DECLARE cost_kw,kw,Read_dmax double;
  DECLARE ayuda  varchar(200);
  DECLARE fin ,bandera smallint;
  DECLARE  f1,fechasql  date;
  DROP TABLE IF EXISTS TReporte;
  CREATE TEMPORARY TABLE TReporte(fecha date,kw float,
  vcosto double, importekw double, demadmax float,
  v_dmax float,importedmax float, pptotal double);

 -- configuracion bucle
  set fin=0;
  set fechasql = LAST_DAY(Fecha1);
  set f1 = date(Fecha1);
  set total =0;
  select if(fechasql > Fecha2,1,0) into bandera;
  if (bandera =1) then
     select  date(Fecha2) into fechasql ;
  end if;
-- fin configuracion bucle
  WHILE (fin =0 ) DO

  SELECT ValorKwDmax,ValorKw into v_dmax,v_costo
  FROM tnegocio_3 where year(fecha)=year(fechasql)
  and month(fecha)=month(fechasql);
  set total =0;
  set dtotal =0;
  set cost_kw =0;

  set kw = KWH(f1,fechasql,2);
```

```
  if kw <0 then  set @error = kw; else set @error = 0; end if;
  call Get_error('tnegocio_3',fechasql,@error);
  set ayuda = 'Minimo consumo Dmax*8*v_dmax ';

    set Read_dmax = Get_demanda(f1,fechasql);
    set dtotal = Read_dmax*8*v_dmax;
    set cost_kw = (kw*v_costo) + (Read_dmax*v_dmax);

    if (dtotal > cost_kw) then
      set total = dtotal;
    else
      set total = cost_kw;
    end if;

   insert into TReporte(fecha,kw,vcosto,importekw,demadmax,v_dmax,
   importedmax,pptotal)values(f1,kw,v_costo,cost_kw,Read_dmax,
   v_dmax,Read_dmax*v_dmax,total);

  -- setup fin bucle y next iteracion
  select  fechasql  + interval 1 day into f1;
  select if(f1 > Fecha2,1,0) into fin;
  set fechasql = LAST_DAY(f1);

   select if(fechasql > Fecha2,1,0) into bandera;

  if (bandera =1) then
     select  date(Fecha2) into fechasql ;
  end if;

  END WHILE;

  call get_PF(Fecha1,Fecha2,total);
  select r.fecha,r.kw,r.vcosto,r.importekw,r.demadmax,r.v_dmax,
  r.importedmax,r.pptotal,ayuda ,t.KWHA,t.Factor_Potencia
  ,t.Bonificacion ,t.cargo,t.cargo_bajatension ,t.total,
  @error from T_FP t, TReporte r;

END $$
DELIMITER ;
```

la función Get_demanda se explica en la sección 4.20.8 y el procedimiento get_PF en la sección 4.20.3. Un ejemplo del archivo .txt separado por comas para actualización de tarifas se observa a continuación:

```
Fecha, ValorKw demanda maxima,  Costo kw -> Minimo 8*Dmax*Vkwdmax
2017-08-01,275.27,1.472
```

4.15. Tarifa Negocio OM

Tarifa ordinaria para servicio general en media tensión, con demanda menor a 100 kW.

Aplicación
Esta tarifa se aplicará a los servicios que destinen la energía a cualquier uso, suministrados en media tensión, con una demanda menor a 100 kW.

Cuotas aplicables en el mes de abril de 2017.

Se aplicarán los siguientes cargos por la demanda máxima medida y por la energía consumida:

Region	Cargo por kilowatt de demanda máxima medida	Cargo por kilowatt - hora de energía consumida
Baja California	$ 182.54	$ 1.487
Baja California Sur	$ 202.12	$ 2.008
Central	$ 206.49	$ 1.487
Noreste	$ 189.86	$ 1.393
Noroeste	$ 193.83	$ 1.382
Norte	$ 190.67	$ 1.393
Peninsular	$ 213.20	$ 1.419
Sur	$ 206.49	$ 1.437

Figura 4-16.: Costo de tarifa Negocio OM, por Región [9].

Mínimo mensual
El importe que resulta de aplicar 10 veces el cargo por kilowatt de demanda máxima medida.

Demanda contratada
La demanda contratada la fijará inicialmente el usuario; su valor no será menor del 60 % de la carga total conectada, ni menor de 10 kilowatts o la capacidad del mayor motor o aparato instalado. En el caso de que el 60 % de la carga total conectada exceda la capacidad de la subestación del usuario, sólo se tomará como demanda contratada la capacidad de dicha subestación a un factor de 90 %.

Temporadas de verano y fuera de verano
Para la aplicación de las cuotas en las regiones Baja California y Baja California Sur se definen las siguientes temporadas:

Verano:
Región Baja California: del 1 de mayo, al sábado anterior al último domingo de octubre.
Región Baja California Sur: del primer domingo de abril, al sábado anterior al último domingo de octubre.

Fuera de verano:
Región Baja California: del último domingo de octubre al 30 de abril.
Región Baja California Sur: del último domingo de octubre al sábado anterior al primer domingo de abril.

Demanda máxima medida
La demanda máxima medida se determinará mensualmente por medio de instrumentos de medición, que indican la demanda media en kilowatts, durante cualquier intervalo de 15 minutos, en el cual el consumo de energía eléctrica sea mayor que en cualquier otro intervalo de 15 minutos en el periodo de facturación. Cualquier fracción de kilowatt de demanda máxima medida se tomará como kilowatt completo. Cuando la demanda máxima medida exceda de 100 kilowatts, el usuario deberá solicitar al suministrador su incorporación a la tarifa H-M. De no hacerlo, al tercer mes consecutivo en que exceda la demanda de 100 kilowatts, será reclasificado por el suministrador en la tarifa H-M, notificando al usuario.

4.15.1. Diseño tabla TNegocio_OM

La tabla diseñada para la tarifa negocio OM se observa en la Figura **4-17**.

El código SQL para la creación de esta tabla se puede observar a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_om`;
CREATE TABLE  `bdelectric`.`tnegocio_om` (
  `Fecha` date NOT NULL,
  `Region` varchar(45) NOT NULL,
  `ValorKwDmax` double NOT NULL,
  `ValorKw` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`,`Region`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

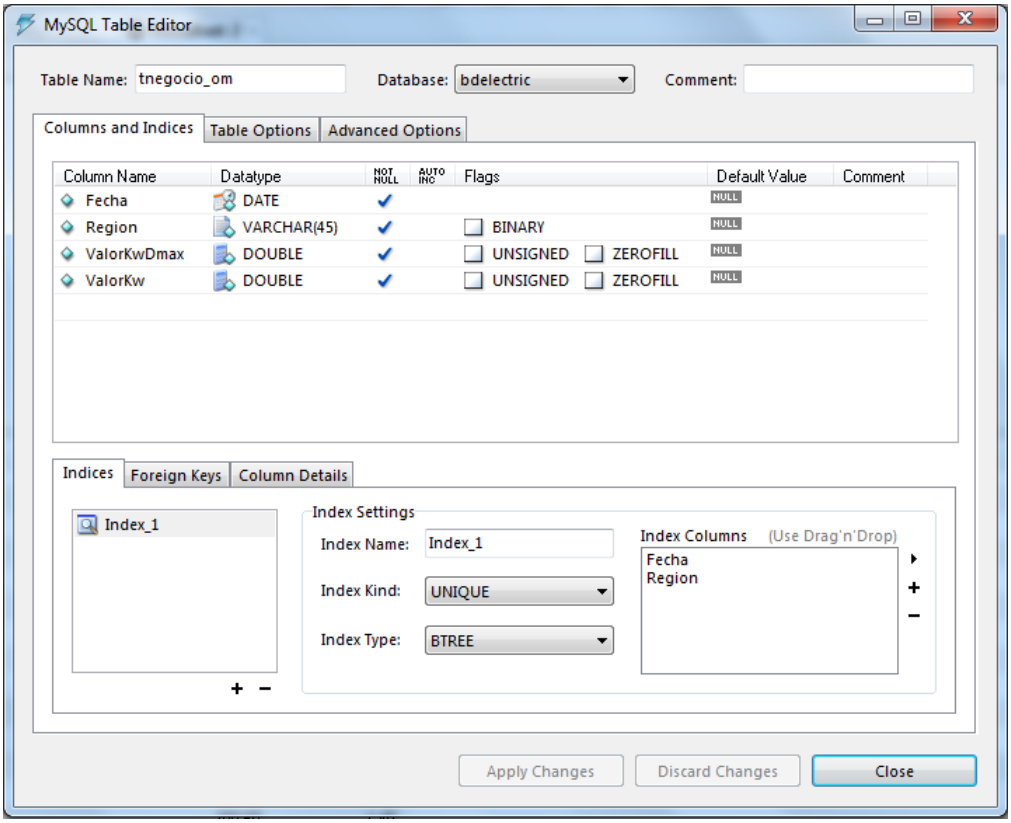


Figura 4-17.: Diseño tabla Negocio OM.

El algoritmo de cálculo de tarifa es similar a la tarifa negocio 3. El código SQL para calculo se puede observar a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_om`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_om`(Fecha1 datetime,
 Fecha2 datetime, Dmax double,Reg varchar(50))
BEGIN
  DECLARE total double;
  DECLARE  dtotal,porc double;
  DECLARE v_costo double;
  DECLARE v_dmax,kwt double;
  DECLARE cost_kw,kw,Read_dmax,cost_dmax double;
  DECLARE fin ,bandera smallint;
  DECLARE  f1,fechasql  date;
  DROP TABLE IF EXISTS TReporte;
  CREATE TEMPORARY TABLE TReporte(fecha date,porcion float,kw float
```

```
  ,vcosto double, importekw double, demadmax float,v_dmax
  float,importedmax float, pptotal double);

-- setup inicio de bucle while
  set fin=0;
  set fechasql = LAST_DAY(Fecha1);
  set f1 = date(Fecha1);
  set kwt = KWH(Fecha1,Fecha2,2);
  set total =0;

   IF(kwt<0) THEN
        set @error =kwt;
   END IF;

  select if(fechasql > Fecha2,1,0) into bandera;

  if (bandera =1) then
     select  date(Fecha2) into fechasql ;
  end if;

WHILE (fin =0 ) DO
 --    bucle while
  set dtotal =0;
  set cost_kw =0;
  set kw=0;
  set v_dmax =0;
   set v_costo =0;

   set kw = KWH(f1,fechasql,2);
   IF(kw<0) THEN
        set @error =kw;
   END IF;

  SELECT ValorKwDmax,ValorKw into v_dmax,v_costo
  FROM tnegocio_om where month(fecha)=month(fechasql)
  and year(fecha)=year(fechasql) and Region like Reg limit 1;

   IF(v_dmax IS NULL) THEN
        set @error =-8004;
   END IF;
```

```
    set Read_dmax = Get_demanda(f1,fechasql);
    set cost_kw = kw*v_costo;
    set cost_dmax =(Read_dmax*v_dmax)*(kw/kwt);

    set total = total + cost_kw + cost_dmax;
    set porc = kw/kwt;

   insert into TReporte(fecha,porcion,kw,vcosto,importekw,
   demadmax, v_dmax,importedmax,pptotal)values (f1,porc,kw,
   v_costo,cost_kw,Read_dmax,v_dmax,cost_dmax,total);

  -- setup fin while + proxima iteracion

  select fechasql + interval 1 day into f1;
  select if(f1 > Fecha2,1,0) into fin;
  set fechasql = LAST_DAY(f1);

   select if(fechasql > Fecha2,1,0) into bandera;

  if (bandera =1) then
     select date(Fecha2) into fechasql ;
  end if;

  END WHILE;

  call get_PF(Fecha1,Fecha2,total);

  select r.fecha,r.porcion,r.kw,r.vcosto,r.importekw,r.demadmax,
  r.v_dmax,r.importedmax,r.pptotal,t.KWHA,t.Factor_Potencia
  ,t.Bonificacion ,t.cargo ,t.cargo_bajatension,
  t.total,@error from TReporte r,T_FP t;

END $$
DELIMITER ;
```

Un ejemplo de archivo .txt separado por comas para actualización de tarifa se observa a continuación:

```
Fecha,region,ValorKwdemandamaxima,Costokw->Minimo10*Dmax*Vkwdmax
2017-02-01,BajaCalifornia,160.67,1.230
```

```
2017-02-01,BajaCaliforniaSur,174.85,1.492
2017-02-01,Central,200.65,1.500
2017-02-01,Noreste,184.49,1.405
2017-02-01,Noroeste,188.35,1.394
2017-02-01,Norte,185.27,1.405
2017-02-01,Peninsular,207.16,1.432
2017-02-01,Sur,200.65,1.450
```

4.16. Tarifa Negocio HM

Haciendo análisis de esta tarifa en especial se identificaron tres tablas para manejo de la tarifa, las variables a considerar para el diseño son:

- Periodos de horarios Base, intermedio y punta
- Regiones
- Demanda Facturable
- Precios Kw/h demanda facturable, Kw/h periodo base , Kw/h periodo Intermedio, Kw/h Periodo punta
- Factores de corrección FRB, FRI
- factor de potencia

De acuerdo a estas variables se diseñó las siguientes tablas para manejo de la tarifa.

4.16.1. Diseño Tabla Tnegocio_hm

Esta tabla contiene los precios por mes, región, Precios Kw/h demanda facturable, Kw/h periodo punta, Kw/h periodo Intermedio, Kw/h Periodo base respectivamente. En la figura **4-18**, se puede observar tabla para negocio HM del mes de Mayo 2017.

Región	Cargo por kilowatt de demanda facturable	Cargo por kilowatt - hora de energía de punta	Cargo por kilowatt - hora de energía intermedia	Cargo por kilowatt - hora de energía de base
Baja California	$ 324.95	$ 25.002	$ 11.015	$ 0.8652
Baja California Sur	$ 312.32	$ 20.060	$ 15.281	$ 10.818
Central	$ 225.15	$ 23.962	$ 12.197	$ 10.196
Noreste	$ 206.99	$ 22.135	$ 11.326	$ 0.9279
Noroeste	$ 211.41	$ 22.265	$ 11.239	$ 0.9409
Norte	$ 207.96	$ 22.293	$ 11.434	$ 0.9295
Peninsular	$ 232.68	$ 23.438	$ 11.461	$ 0.9436
Sur	$ 225.15	$ 23.467	$ 11.652	$ 0.9696

Figura 4-18.: Precios del mes de agosto 2016 por región, costos: Kw/h demanda facturable, Kw/h periodo punta, Kw/h periodo Intermedio, Kw/h Periodo base para tarifa negocio HM, sección 2 CFE [9].

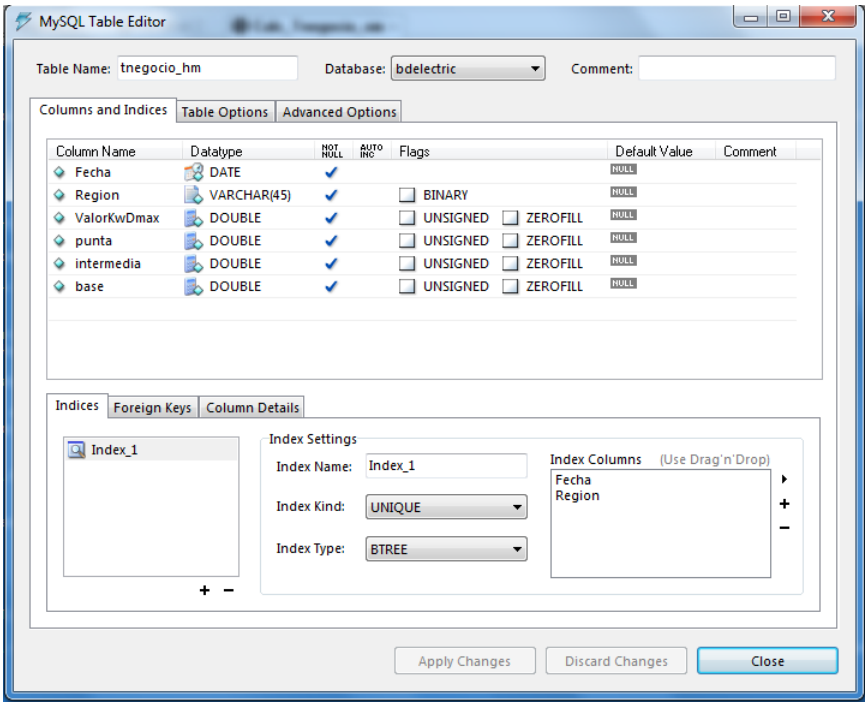


Figura 4-19.: Diseño tabla Tnegocio_hm, MySQL Query Browser

El respectivo código SQL para creación de esta tabla se anexa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_hm`;
CREATE TABLE  `bdelectric`.`tnegocio_hm` (
  `Fecha` date NOT NULL,
  `Region` varchar(45) NOT NULL,
  `ValorKwDmax` double NOT NULL,
```

```
  `punta` double NOT NULL,
  `intermedia` double NOT NULL,
  `base` double NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`,`Region`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Con base a la figura anterior, el ejemplo de archivo .txt para la importación y actualización de datos en la base de datos seria:

```
Fecha, region, Costo, kwDemanda, kwPunta, base
2016-09-01,Baja California,298.75,2.0696,0.8177,0.6423
2016-09-01,Baja California Sur,287.13,1.6604,1.1345,0.8030
2016-09-01,Central,206.99,1.9835,0.9054,0.7569
2016-09-01,Noreste,190.31,1.8321,0.84,0.6888
2016-09-01,Noroeste,194.36,1.8428,0.8343,0.6986
2016-09-01,Norte,191.20,1.8451,0.8487,0.6899
2016-09-01,Peninsular,213.91,1.9400,0.8507,0.7006
2016-09-01,Sur,206.99,1.9426,0.8650,0.7198
```

Nótese que los textos no llevan comillas ya que si se coloca estas serán importadas en la tabla en la base de datos. Seguidamente se proceda a diseñar la tabla para almacenar los periodos base, intermedio punta como el que se muestra en la Figura **4-20**.

Región Baja California

Del 1º de mayo al sábado anterior al último domingo de octubre

Día de la semana	Base	Intermedio	Punta
lunes a viernes		0:00 - 14:00 18:00 - 24:00	14:00 - 18:00
sábado		0:00 - 24:00	
domingo y festivo		0:00 - 24:00	

Del último domingo de octubre al 30 de abril

Día de la semana	Base	Intermedio	Punta
lunes a viernes	0:00 - 17:00 22:00 - 24:00	17:00 - 22:00	
sábado	0:00 - 18:00 21:00 - 24:00	18:00 - 21:00	
domingo y festivo	0:00 - 24:00		

Figura 4-20.: Ejemplo CFE de periodos base, intermedio, punta para Baja california [9]

Nótese que el periodo se divide en dos periodos en el año del 1 mayo último domingo de octubre y del último domingo de octubre al 30 de mayo para esto se diseñó que los periodos

fueran del 1 mayo al 31 de octubre, y del 1 Noviembre al 30 de abril. También se escogió el primer día de la semana como el domingo número que viene por defecto en el motor de base de datos MySQL y es invariante a cambios de configuración para la plataforma de software como Windows.

4.16.2. Diseño Tabla TNegocio_hm_periodos

Esta tabla contiene las siguientes columnas mes con la letras MO (mayo-octubre) ó OA (Octubre mayo), región, tipo de periodo (Base, Intermedio, Punta) día inicio tarifa, día fin de tarifa, hora inicio, hora fin.

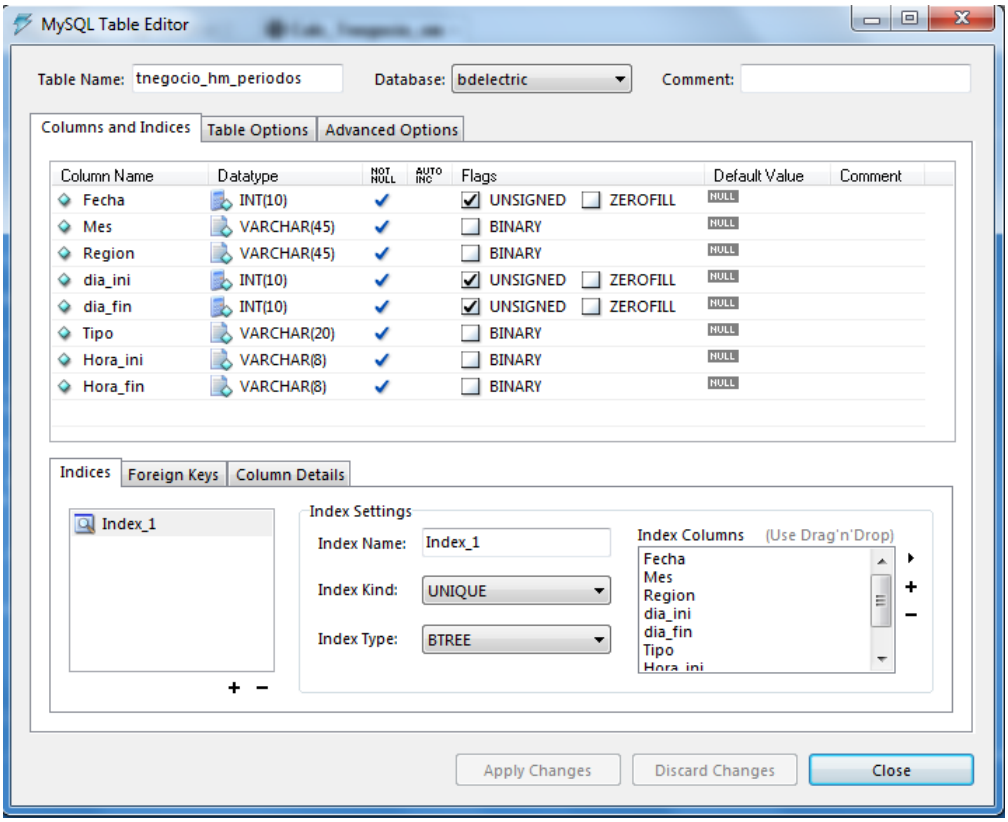


Figura 4-21.: Diseño de tabla Tnegocio_HM_periodos

El código SQL para creación de la tabla se anexa a continuación:

```
DROP TABLE IF EXISTS `bdelectric`.`tnegocio_hm_periodos`;
CREATE TABLE  `bdelectric`.`tnegocio_hm_periodos` (
  `Fecha` int(10) unsigned NOT NULL,
  `Mes` varchar(45) NOT NULL,
  `Region` varchar(45) NOT NULL,
  `dia_ini` int(10) unsigned NOT NULL,
```

```
  `dia_fin` int(10) unsigned NOT NULL,
  `Tipo` varchar(20) NOT NULL,
  `Hora_ini` varchar(8) NOT NULL,
  `Hora_fin` varchar(8) NOT NULL,
  UNIQUE KEY `Index_1` (`Fecha`,`Mes`,`Region`,`dia_ini`,
  `dia_fin`,`Tipo`,`Hora_ini`,`Hora_fin`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Un ejemplo de dato en el archivo .txt para importación y actualización de datos, donde MO significa mayo-octubre y OA Octubre-Abril.

```
Ano,Mes,Region,dia_ini,dia_fin,Tipo,Hora_ini,Hora_fin
2016,MO,Baja California,2,6,INTERMEDIO,00:00,14:00
2016,MO,Baja California,2,6,INTERMEDIO,18:00,24:00
2016,MO,Baja California,2,6,PUNTA,14:00,18:00
2016,MO,Baja California,7,7,INTERMEDIO,00:00,24:00
2016,MO,Baja California,1,1,INTERMEDIO,00:00,24:00
2016,OA,Baja California,2,6,BASE,00:00,17:00
2016,OA,Baja California,2,6,BASE,22:00,24:00
2016,OA,Baja California,2,6,INTERMEDIO,17:00,22:00
2016,OA,Baja California,7,7,BASE,00:00,18:00
2016,OA,Baja California,7,7,BASE,21:00,24:00
2016,OA,Baja California,1,1,BASE,00:00,24:00
2016,MO,Baja California Sur,2,6,INTERMEDIO,00:00,12:00
2016,MO,Baja California Sur,2,6,INTERMEDIO,22:00,24:00
2016,MO,Baja California Sur,2,6,PUNTA,12:00,22:00
2016,MO,Baja California Sur,7,7,INTERMEDIO,00:00,19:00
2016,MO,Baja California Sur,7,7,INTERMEDIO,22:00,24:00
2016,MO,Baja California Sur,7,7,PUNTA,19:00,22:00
2016,MO,Baja California Sur,1,1,INTERMEDIO,00:00,24:00
2016,OA,Baja California Sur,2,6,BASE,00:00,18:00
2016,OA,Baja California Sur,2,6,BASE,22:00,24:00
2016,OA,Baja California Sur,2,6,INTERMEDIO,18:00,22:00
2016,OA,Baja California Sur,7,7,BASE,00:00,18:00
2016,OA,Baja California Sur,7,7,BASE,21:00,24:00
2016,OA,Baja California Sur,7,7,INTERMEDIO,18:00,21:00
2016,OA,Baja California Sur,1,1,BASE,00:00,19:00
2016,OA,Baja California Sur,1,1,BASE,21:00,24:00
2016,OA,Baja California Sur,1,1,INTERMEDIO,19:00,21:00
2016,MO,Central,2,6,BASE,00:00,06:00
2016,MO,Central,2,6,INTERMEDIO,06:00,20:00
```

```
2016,MO, Central ,2 ,6 ,INTERMEDIO,22:00 ,24:00
2016,MO, Central ,2 ,6 ,PUNTA,20:00 ,22:00
2016,MO, Central ,7 ,7 ,BASE,00:00 ,07:00
2016,MO, Central ,7 ,7 ,INTERMEDIO,07:00 ,24:00
2016,MO, Central ,1 ,1 ,BASE,00:00 ,19:00
2016,MO, Central ,1 ,1 ,INTERMEDIO,19:00 ,24:00
2016,OA, Central ,2 ,6 ,BASE,00:00 ,06:00
2016,OA, Central ,2 ,6 ,INTERMEDIO,06:00 ,18:00
2016,OA, Central ,2 ,6 ,INTERMEDIO,22:00 ,24:00
2016,OA, Central ,2 ,6 ,PUNTA,18:00 ,22:00
2016,OA, Central ,7 ,7 ,BASE,00:00 ,08:00
```

Finalmente se diseñó tabla para almacenar los factores de ajustes FRB y FRI que rigen a esta tarifa para cálculo de demanda facturable.

Región	FRI	FRB
Baja California	0.141	0.070
Baja California Sur	0.195	0.097
Central	0.300	0.150
Noreste	0.300	0.150
Noroeste	0.300	0.150
Norte	0.300	0.150
Peninsular	0.300	0.150
Sur	0.300	0.150

Figura 4-22.: Ejemplo de Factores FRI y FRB vs región

La demanda facturable se define como se establece a continuación:

$$DF = DP + FRI * max(DI - DP, 0) + FRB * max(DB - DPI, 0) \quad (4\text{-}1)$$

Dónde:

DP es la demanda máxima medida en el periodo de punta
DI es la demanda máxima medida en el periodo intermedio
DB es la demanda máxima medida en el periodo de base
DPI es la demanda máxima medida en los periodos de punta e intermedio
FRI y FRB son factores de reducción que tendrán los siguientes valores, dependiendo de la región tarifaria.
max devuelve el máximo de los dos argumentos.

4.16.3. Diseño Tabla TNegocio_hm_fac_reduccion

El diseño de la tabla se puede observar en la Figura **4-23**.

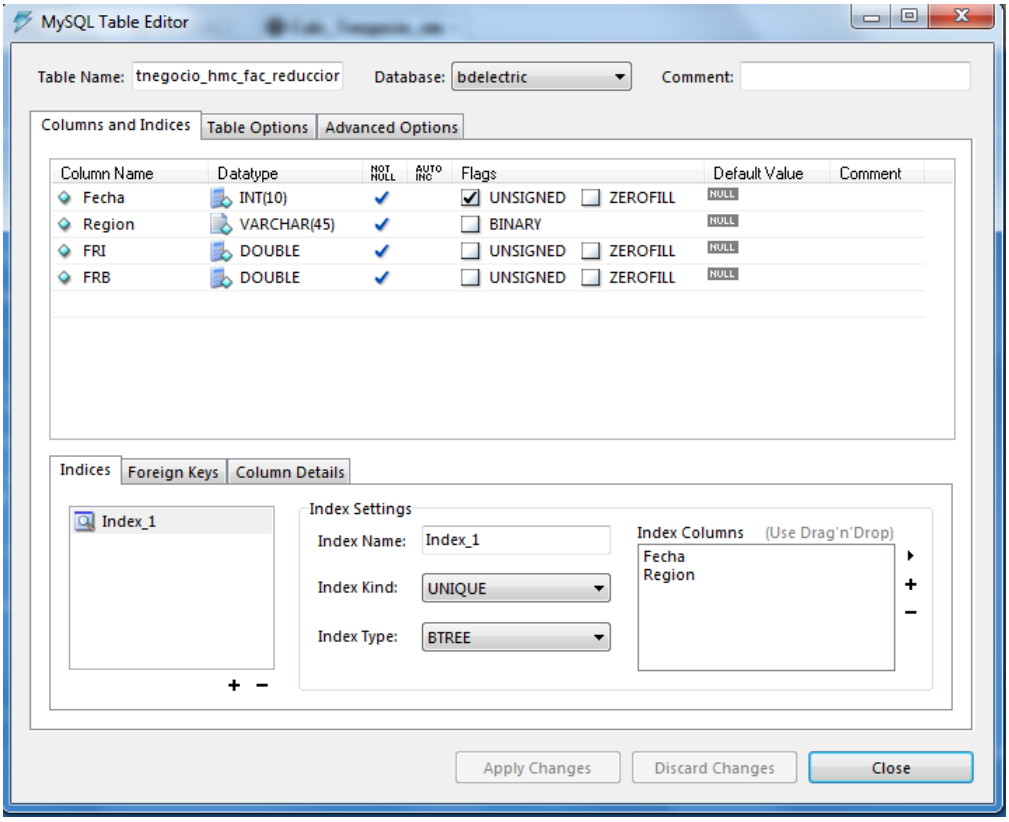


Figura 4-23.: Diseño Tabla TNegocio_hm_fac_reduccion

El ejemplo de archivo .txt para importación y actualización de datos

```
Ano, Region, FRI,FRB
2016,Baja California,0.141,0.070
2016,Baja California Sur,0.195,0.097
2016,Central,0.300,0.150
2016,Noreste,0.300,0.150
2016,Noroeste,0.300,0.150
2016,Norte,0.300,0.150
2016,Peninsular,0.300,0.150
2016,Sur,0.300,0.150
```

4.16.4. Algoritmo de cálculo Tarifa Negocio HM

El algoritmo de cálculo de la tarifa negocio HM se puede observar en el siguiente diagrama de flujo.

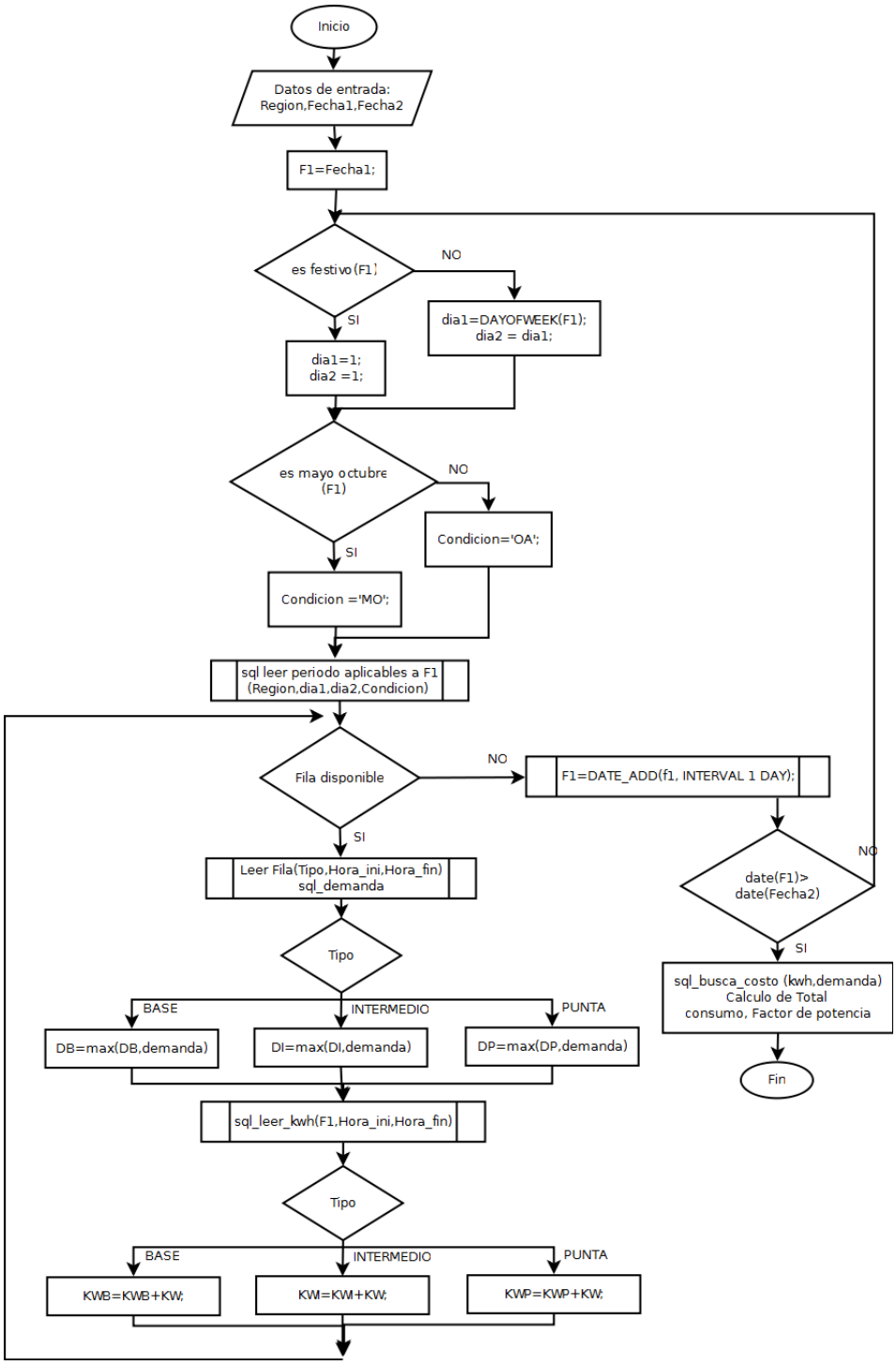


Figura 4-24.: Diagrama de flujo para cálculo tarifa negocio HM

El código SQL para el diagrama de flujo anterior se anexa a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`Calc_Tnegocio_hm`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`Calc_Tnegocio_hm`(Fecha1 datetime,fecha2 datetime,
Reg varchar(40))
BLOCK1: BEGIN

  DECLARE total double;
  DECLARE  f1  datetime;
  DECLARE  flag,Mes1  int;
  DECLARE  fin,is_festivo,is_Mayo_oct,dia1,dia2  int;
  DECLARE  DB,DI,DP,DPI,aux DOUBLE;
  DECLARE  KWB,KWI,KWP DOUBLE;
  DECLARE  condicion,Tipo1 varchar(10);
  DECLARE  H_ini, H_fin time;
  DECLARE v_punta,v_intermedia,v_base,v_dmax double;
  DECLARE e_base,e_intermedia,e_punta,e_dem_fac,demanda_fac double;

    set f1= Fecha1;
    set flag =0;
    set  fin = 0;
    set  DB=0;
    set  DI=0;
    set  DP=0;
    set  KWI=0;
    set  KWB=0;
    set  KWP=0;
   SET  TOTAL =0;

 WHILE (fin =0 ) DO

    set is_festivo = (select count(*) from festivos
    where fecha =date(f1));
    set is_Mayo_oct = (select  if (date(f1)>= ( SELECT
     date(CONCAT_WS('',year(CURDATE()),'-05-01'))) and date(f1)<=
     (SELECT date(CONCAT_WS('',year(CURDATE()),'-10-31'))),1,0  ));

     if is_Mayo_oct =1 then
```

```
      set condicion='MO';
     else
      set condicion='OA';
     end if;

     if (is_festivo =1) then
       set dia1=1;
       set dia2=1;
     else
       set dia1= (SELECT DAYOFWEEK(f1));
       set dia2=dia1;
     end if;
      set @error =-8012;

BLOCK2: BEGIN
       DECLARE done1 INT DEFAULT FALSE;
       DECLARE cursor_1 CURSOR FOR SELECT Tipo,Hora_ini,
       Hora_fin FROM tnegocio_hm_periodos where
       Fecha = year(fecha2) and Mes=condicion  and
       Region like  Reg and  dia1>= dia_ini
       and dia2 <= dia_fin ;
       DECLARE CONTINUE HANDLER FOR NOT FOUND SET done1 = TRUE;
        OPEN cursor_1;

read_loop: LOOP
         FETCH cursor_1 INTO Tipo1, H_ini,H_fin;
         IF done1 THEN LEAVE read_loop; END IF;

         set aux = (select max(RealpowerPresentDemand) as 'max'
         from consumo where date(fecha)>=date(f1)
         and date(fecha)<= date(fecha2) and HOUR(fecha) >=
         hour(H_ini) and HOUR(fecha)<=hour(H_fin));
          SET aux = (select IFNULL(aux,0));

            CASE Tipo1
            WHEN 'BASE' THEN
                  if aux > DB then
                     SET DB=aux;
                  end if;
            WHEN 'INTERMEDIO' THEN
```

```
                if aux > DI then
                   SET DI=aux;
                end if;
            WHEN 'PUNTA' THEN
                if aux > DP then
                   SET DP=aux;
                end if;
            END CASE;

        set aux = (select ((select `Energia Real Total` as 'A'
        from consumo where  date(fecha)=f1 and  hour(fecha)
        <= hour(H_fin) order by `Energia Real Total` desc limit 1)
        -(select `Energia Real Total` as 'B' from consumo
        where  date(fecha)=f1 and  hour(fecha)
        >= hour(H_ini) limit 1))/1000 as 'kw');

            SET aux = (select IFNULL(aux,0));
            if aux <0 then
              set aux=0;
            end if;

             CASE Tipo1
             WHEN 'BASE' THEN set KWB = KWB +AUX;
             WHEN 'INTERMEDIO' THEN set KWI = KWI +AUX;
             WHEN 'PUNTA' THEN set KWP = KWP +AUX;
             END CASE;

          END LOOP;
          CLOSE cursor_1;
END  BLOCK2;

   set f1 = (SELECT DATE_ADD(f1, INTERVAL 1 DAY));
   set  fin = (SELECT  if (DATE(f1)>DATE(fecha2),1,0));
   SET flag = flag +1;
   SET  TOTAL =   TOTAL +1;
   set @error =0;
 END WHILE;

    IF DI > DP THEN
      SET DPI = DI;
```

```
      ELSE
       SET DPI = DP;
     END IF;

    set Mes1 = MONTH(fecha2);
    call Get_error('tnegocio_hm',Fecha1,-8012);

    set @ano =year(Fecha1);

   SELECT ValorKwDmax,punta,intermedia,base into v_dmax,v_punta,
   v_intermedia,v_base FROM tnegocio_hm where
   month(fecha)=month(Fecha2) and year(fecha)=
   year(Fecha2) and Region like Reg;
    set e_base = kwB * v_base;
    set e_intermedia = kwI * v_intermedia;
    set e_punta = kwP * v_punta;
    set demanda_fac = DP + FRI(Reg,1) * max_f(DI - DP,0)
        + FRB(Reg,1) * max_f(DB - DPI,0);
    set e_dem_fac = demanda_fac*v_dmax;
    SET total= e_base + e_intermedia + e_punta + e_dem_fac;

   IF(v_dmax IS NULL) THEN
        set @error =-8012;
   END IF;
    call get_PF(Fecha1,Fecha2,total);

   select KWB,KWI,KWP,DB,DI,DP,DPI,v_dmax,v_punta,v_intermedia,
   v_base,demanda_fac,TOTAL as 'ptotal',t.KWHA,t.Factor_Potencia
   ,t.Bonificacion ,t.cargo,t.cargo_bajatension ,t.total,
   @error from T_FP t;

end BLOCK1 $$
DELIMITER ;
```

4.17. Tarifa Negocio HMC, HS, HSL, HT, HTL

Estas tarifas son similares a la tarifa HM solo que es tarifa HMC es horaria para servicio general en media tensión, con demanda de 100 kw o más, para corta utilización. La tarifa HS es tarifa horaria para servicio general en alta tensión, nivel su transmisión. La tarifa HSL es

tarifa horaria para servicio general en alta tensión, nivel subtransmisión, para larga utilización. La tarifa HT es Tarifa horaria para servicio general en alta tensión, nivel transmisión y la tarifa HTL es Tarifa horaria para servicio general en alta tensión, nivel transmisión para larga utilización.

Por lo que su diseño de tablas y algoritmo de cálculos son iguales a la tarifa HM.
Las demás tarifas presentes en CFE son tarifas especiales muy poco usada y para usuarios específicos por lo que no se tuvieron en cuenta en este proyecto.

4.18. Diseño tabla Errores

Esta tabla contiene dos columnas Iderror y descripción donde se guardan los posibles errores que devuelven las funciones y la explicación de los mismos. El rango de errores se escogió desde -8001 hasta -8017 ya que este rango son errores manejados por el usuario en LabVIEW. La lista de errores se puede observar en la tabla **4-6**.

Iderror	Descripcion
-8017	Las tablas: tnegocio_htl, tnegocio_htl_periodos y tnegocio_htl_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa htl
-8016	Las tablas: tnegocio_ht, tnegocio_ht_periodos y tnegocio_ht_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa ht
-8015	Las tablas: tnegocio_hsl, tnegocio_hsl_periodos y tnegocio_hsl_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa hsl
-8014	Las tablas: tnegocio_hs, tnegocio_hs_periodos y tnegocio_hs_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa hs
-8013	Las tablas: tnegocio_hmc, tnegocio_hmc_periodos y tnegocio_hm_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa hmc
-8012	Las tablas: tnegocio_hm, tnegocio_hm_periodos y tnegocio_hm_fac_reduccion no tiene datos actualizada para la Fecha de consulta en la tarifa hm
-8011	La tabla tnegocio_9n no tiene datos actualizada para la Fecha de consulta en la tarifa 9n
-8010	La tabla tnegocio_9 no tiene datos actualizada para la Fecha de consulta en la tarifa 9
-8009	La tabla tnegocio_7 no tiene datos actualizada para la Fecha de consulta en la tarifa 7
-8008	La tabla tnegocio_6 no tiene datos actualizada para la Fecha de consulta en la tarifa 6
-8007	La tabla tnegocio_2 no tiene datos actualizada para la Fecha de consulta en la tarifa 2
-8006	La tabla tnegocio_5a no tiene datos actualizada para la Fecha de consulta en la tarifa 5A
-8005	La tabla tnegocio_5 no tiene datos actualizada para la Fecha de consulta en la tarifa 5
-8004	La tabla tnegocio_om no tiene datos actualizada para la Fecha de consulta en la tarifa OM
-8003	La base de datos no tiene la información de la tarifa para realizar el cálculo
-8002	la fecha Final no reporta datos en la base de datos, cambie a otra fecha
-8001	la fecha Inicial no reporta datos en la base de datos, cambie a otra fecha

Tabla 4-6.: Lista de errores MySQL.

4.19. Diseño tabla festivos

Como las tarifas HM, HMC, HS, HSL, HT, HTL tienen tarifas dependiendo del día del año y si este día es festivo tiene otra tarifa se creó una tabla festivos con las fechas que son festivas para México y aplican a la tarifa correspondiente como se observa a continuación:

```
DROP TABLE IF EXISTS `BDELECTRIC`.`FESTIVOS`;
CREATE TABLE `BDELECTRIC`.`FESTIVOS` (
  `FECHA` DATE NOT NULL
) ENGINE=INNODB DEFAULT CHARSET=LATIN1;
```

la lista de festivos par el año 2016 y 2017 se puede observar en la tabla **4-7**.

Fecha
2016-01-01
2016-02-01
2016-03-21
2016-03-24
2016-03-25
2016-05-05
2016-09-01
2016-09-16
2016-11-02
2016-11-21
2016-12-12
2017-01-01
2017-02-01
2017-03-21
2017-03-24
2017-03-25
2017-05-05
2017-09-01
2017-09-16
2017-11-02
2017-11-21
2017-12-12

Tabla 4-7.: Lista de Festivos 2016,2017.

4.20. Rutinas especiales en MySQL

Se diseñaron funciones y procedimientos de almacenados que se usaron en las diferentes tarifas implementadas en este proyecto, a continuación se describen las funciones y procedimientos creados para soportar las tarifas.

4.20.1. Función KWH

Esta función tiene como para metros de entrada Fecha1 fecha de inicio, Fecha2 Fecha de fin de búsqueda, opción forma de búsqueda de acuerdo a la necesidad de los algoritmos. la opción 1 busca el registro mas alto en la fecha especifica respecto a la columna 'Energía Real Total' y le resta el primer registro de consumo de 'Energia Real Total' encontrado y luego los divide entre mil para convertirlo a KWH. la opción 2 es una opción mejorada de la 1 ya que valida que el dato guardado sea mayor a 1 ya que aveces se registraban ceros en la base

de datos o nulos.

Esta función es una de las principales del programa que se encarga de buscar el consumo de Energía Real que es la que cobra CFE dentro de un periodo estipulado, la función busca el primer registro del registro de Energía Real acumulada en la base de datos en la tabla consumo si no encuentra datos en la fecha de inicio devuelve el error -8001 ya que se consideró que el sistema debe estar adquiriendo datos constante mente y si deja días sin adquirir la medición va hacer errónea y si tomara datos de otro día cercano también habría error en la medición. En la rutina si no encuentra datos de la Energía Real acumulada en la Fecha2 devuelve el error -8002, este rango de errores fueron tomados desde este rango ya que son errores custom de usuarios en LabVIEW. La rutina después de buscar estos datos los resta y divide entre 1000 ya que los medidores guardan datos en vatios y CFE cobra en Kilovatios. La rutina se puede observar a continuación:

```
DELIMITER $$

DROP FUNCTION IF EXISTS `bdelectric`.`KWH`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`KWH`(fecha1 datetime,Fecha2 datetime,opcion int)
RETURNS double
BEGIN

    declare kw,kw1,kw2 double;

    CASE opcion
       WHEN 1 then
         set kw= (select ((SELECT `Energia Real Total` FROM consumo
         where fecha <= Fecha2  order by `Energia Real Total`
         desc limit 1 )-(SELECT `Energia Real Total`  FROM consumo
         where fecha >= Fecha1 limit 1))/1000 as 'kw') ;
         SET kw = (select IFNULL(kw,0));
         return (kw);

       WHEN 2 THEN

          set kw1 = (SELECT `Energia Real Total` FROM consumo
          where date(fecha)=date(Fecha2) and
          `Energia Real Total` >1 order by idConsumo desc limit 1);
          SET kw1 = (select IFNULL(kw1,-8002));

          if kw1<-1 then
```

```
            return (kw1);
        end if;

        set kw2= (SELECT `Energia Real Total` FROM consumo
        where date(fecha) = date(Fecha1) and
        `Energia Real Total` >1 limit 1);
        SET kw2 = (select IFNULL(kw2,-8001));

        if kw2<-1 then
            return (kw2);
        end if;

        SET kw = (kw1-kw2)/1000;

         if kw <0 then
           set kw= kw*-1;
         end if;

        return (kw);
    END CASE;
END;
 $$
DELIMITER ;
```

4.20.2. Función KWHA

Esta función es similar a KWH pero solo que busca la Energía Aparente acumulada en el periodo específico, usada para medir el factor de potencia en el periodo indicado a través de la relación que se observa en el triángulo de potencia. El código se puede observar a continuación:

```
DELIMITER $$

DROP FUNCTION IF EXISTS `bdelectric`.`KWHA`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`KWHA`(fecha1 datetime,Fecha2 datetime,opcion int)
RETURNS double
BEGIN

    declare kwA double;
```

```
   CASE opcion
      WHEN 1 then

  set kwA= (select ((SELECT `Energia Aparente Total` FROM consumo
     where fecha <= Fecha2  order by `Energia Aparente Total`
     desc limit 1 )-(SELECT `Energia Aparente Total`  FROM consumo
     where fecha >= Fecha1 limit 1))/1000 as 'kw') ;
       SET kwA = (select IFNULL(kwA,0));
       return (kwA);
      WHEN 2 THEN
   set kwA= (select ((SELECT `Energia Aparente Total` FROM consumo
    where date(fecha)= date(Fecha2) and `Energia Aparente Total`>1
    order by idConsumo desc limit 1 )-(SELECT
    `Energia Aparente Total`  FROM consumo where date(fecha)
     = date(Fecha1) and `Energia Aparente Total` >1
     limit 1))/1000 as 'kw') ;
     SET kwA = (select IFNULL(kwA,0));
     return (kwA);
    END CASE;

END;
 $$
DELIMITER ;
```

4.20.3. Procedimiento Get_FP

Esta función busca el factor de potencia entre el periodo de la Fecha1, Fecha2 y aplica cargo al valor total de entrada el cual puede ser una bonificación o una sanción de acuerdo al factor de potencia obtenido en el periodo de búsqueda, a parte busca en la tabla setup si la medición se realiza en baja tensión para aplicar un cargo adicional del 2 % por medición en baja tensión. El código SQL del procedimiento se puede observar a continuación:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `bdelectric`.`get_PF`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`bdelectric`.`get_PF`(Fecha1 datetime,
 Fecha2 datetime,total double)
BEGIN
```

```
    declare FP, Bonificacion_fp , Cargo_fp ,KWHA1, cargobt , aux double;
    DROP TABLE IF EXISTS T_FP;
  CREATE TEMPORARY TABLE T_FP( Factor_Potencia float ,
   Bonificacion float , cargo float  , KWHA float , total float ,
   cargo_bajatension float );

    set  FP = FP_all_date(Fecha1 , Fecha2 );
    set  KWHA1 = KWHA( Fecha1 , Fecha2 , 2 );
    set  cargobt =0;
    delete from T_FP;
      if FP >=90  then
         set Bonificacion_fp = 1/4 * ( 1 - ( 90 / FP ) ) * 100 ;
         if Bonificacion_fp >2.5 then
           set Bonificacion_fp =2.5;
         end if;
         SET total = total - (total*Bonificacion_fp/100);
         set Cargo_fp =0;
      else
         set Cargo_fp = 3/5 * ( ( 90 / FP ) - 1 ) * 100;
         if Cargo_fp >20 then set Cargo_fp =20; end if;
         SET total = total + (total*Cargo_fp/100);
         set Bonificacion_fp=0;
      end if;

    select valor into aux from setup where NombreParametro like
    '%Medicion_en_baja_tension:';

    if aux =1 then
      set cargobt = total*0.02;
    else
      set cargobt = 0;
    end if;

    INSERT INTO T_FP( Factor_Potencia , Bonificacion , cargo
    ,KWHA, cargo_bajatension , total )VALUES
    (FP, Bonificacion_fp , Cargo_fp ,KWHA1, cargobt , total );

END $$

DELIMITER ;
```

4.20.4. Función FP_ALL_DATE

Esta función busca el factor de potencia en el periodo específico, sirve de soporte a la función Get_FP, el código SQL de esta función se puede observar a continuación:

```
DELIMITER $$
DROP FUNCTION IF EXISTS `bdelectric`.`FP_all_date`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
 `bdelectric`.`FP_all_date`(Fecha1 datetime,Fecha2 datetime)
 RETURNS double
BEGIN

  DECLARE fp double;
  declare kw_aparente,kw_real double;

  set kw_aparente = (select ((SELECT `Energia Aparente Total`
  as 'B' FROM consumo where date(fecha)= date(Fecha2)
  order by idConsumo desc limit 1 )-(SELECT
  `Energia Aparente Total` as 'A' FROM
  consumo where fecha > date(Fecha1) limit 1))/1000 as 'kw');
  set kw_aparente =ROUND(kw_aparente,2);
  set kw_real = (select ((SELECT `Energia Real Total` as 'B'
  FROM consumo where date(fecha)= date(Fecha2)  order by
  idConsumo desc limit 1 )-(SELECT `Energia Real Total`
  as 'A' FROM consumo where fecha > date(Fecha1) limit 1))/1000
  as 'kw');
  set kw_real = ROUND(kw_real,2);

  set fp =  (kw_real/kw_aparente) *100;
  set fp = ROUND(FP,2);
  return fp;

END;
 $$
DELIMITER ;
```

4.20.5. Función MAX_F

Esta función devuelve el valor mayor de los valores de entrada (V1 y V2) ya que este valor mayor se usa es las tarifas HM, HMC, HS, HSL, HT, HTL. El código SQL se observa a continuación:

```
DELIMITER $$
DROP FUNCTION IF EXISTS `bdelectric`.`max_f`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`max_f`(v1 double,v2 double) RETURNS double
BEGIN

  DECLARE total double;

  if v1>v2 then
    set total =V1;
  else
    set total =V2;
  end if;
   return total ;
END;
 $$
DELIMITER ;
```

4.20.6. Función FRI

Esta función busca el factor de reducción en horario intermedio aplicable a la región específica y dependiendo de la tarifa de trabajo HM (opción 1), HMC(opción 2), HS (opción 3), HSL (opción 4), HT (opción 5), HTL (opción 6). El código SQL se pude observar a continuación:

```
DELIMITER $$

DROP FUNCTION IF EXISTS `bdelectric`.`FRI`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`FRI`(Reg varchar(29),opcion int) RETURNS double
BEGIN

  DECLARE total double;

  CASE opcion
     WHEN 1 then return (SELECT FRI FROM tnegocio_hm_fac_reduccion
       where  fecha=@ano and Region like Reg) ;
     WHEN 2 THEN return (SELECT FRI FROM tnegocio_hmc_fac_reduccion
       where fecha=@ano and Region like Reg) ;
     WHEN 3 THEN return (SELECT FRI FROM tnegocio_hs_fac_reduccion
```

```
        where  fecha=@ano and Region like Reg) ;
      WHEN 4 THEN return (SELECT FRI FROM tnegocio_hsl_fac_reduccion
        where fecha=@ano and Region like Reg) ;
      WHEN 5 THEN return (SELECT FRI FROM tnegocio_ht_fac_reduccion
        where  fecha=@ano and Region like Reg) ;
      WHEN 6 THEN return (SELECT FRI FROM tnegocio_htl_fac_reduccion
        where fecha=@ano and Region like Reg) ;
       ELSE
         BEGIN
         END;
     END CASE;
END;
 $$
DELIMITER ;
```

4.20.7. Función FRB

Esta función busca el factor de reducción en horario base aplicable a la región específica y dependiendo de la tarifa de trabajo HM (opción 1), HMC(opción 2), HS (opción 3), HSL (opción 4), HT (opción 5), HTL (opción 6). El código SQL se pude observar a continuación:

```
DELIMITER $$
DROP FUNCTION IF EXISTS `bdelectric`.`FRB`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`FRB`(Reg varchar(29),opcion int) RETURNS double
BEGIN

  DECLARE total double;

   CASE opcion
    WHEN 1 then return (SELECT FRB FROM tnegocio_hm_fac_reduccion
      where  fecha=@ano and Region like Reg) ;
    WHEN 2 THEN return (SELECT FRB FROM tnegocio_hmc_fac_reduccion
     where fecha=@ano and Region like Reg) ;
    WHEN 3 THEN return (SELECT FRB FROM tnegocio_hs_fac_reduccion
     where  fecha=@ano and Region like Reg) ;
    WHEN 4 THEN return (SELECT FRB FROM tnegocio_hsl_fac_reduccion
     where fecha=@ano and Region like Reg) ;
    WHEN 5 THEN return (SELECT FRB FROM tnegocio_ht_fac_reduccion
     where  fecha=@ano and Region like Reg) ;
```

```
    WHEN 6 THEN return (SELECT FRB FROM tnegocio_htl_fac_reduccion
      where fecha=@ano and Region like Reg) ;
   END CASE;

END;
 $$
DELIMITER ;
```

4.20.8. Función Get_demanda

Esta función busca la demanda máxima medida en el periodo de búsqueda (Fecha1, Fceha2). El código SQL de esta función se observa a continuación:

```
DELIMITER $$

DROP FUNCTION IF EXISTS `bdelectric`.`Get_demanda`$$
CREATE DEFINER=`root`@`localhost` FUNCTION
`bdelectric`.`Get_demanda`(Fecha1 datetime,Fecha2 datetime)
RETURNS double
BEGIN

  DECLARE Demanda double;

   set Demanda = (SELECT max(RealpowerPresentDemand) FROM
    consumo  where date(Fecha)>=date(Fecha1)
    and date(Fecha)<=date(Fecha2) );
   return Demanda;

END;
 $$
DELIMITER ;
```

5. LabVIEW y MySQL

Para que LabVIEW pueda interactuar con una base de datos en MySQL, lo hace a través de unas librerías ya desarrolladas por MySQL las cuales se conocen como conector ODBC. Un conector ODBC (Open Database Connectivity) es una API (Application Programming Interface) que es un estándar de acceso a bases de datos. Sirve para poder acceder a cualquier dato dentro de una base de datos desde cualquier tipo de aplicación, ya sea en lenguaje C, Python, C#, G, etcétera. Funciona traduciendo los comandos de la aplicación en cuestión a algún comando que el gestionador de sistema de bases de datos pueda entender. LabVIEW ya posee las funciones para interactuar con un conector ODBC, las cuales están en connectivity− > Database como se observa en la Figura **5-1**.

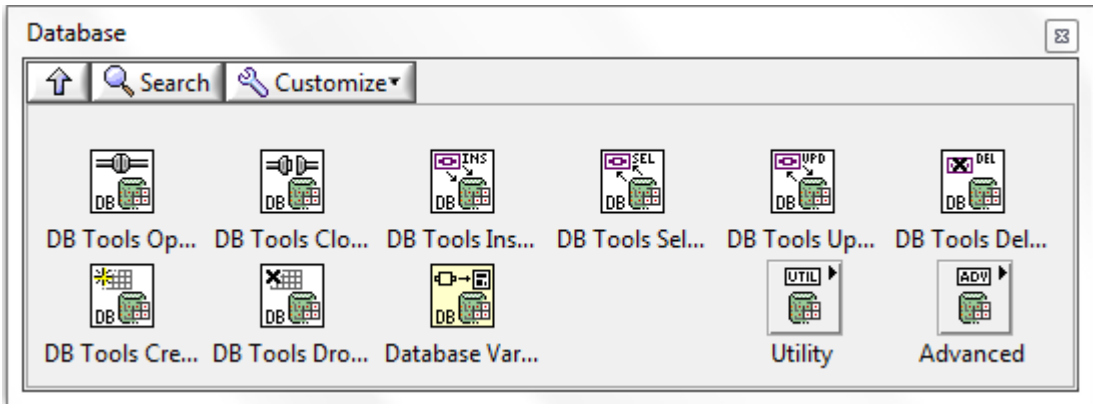


Figura 5-1.: Funciones de LabVIEW para Manejo de base de datos.

Un ejemplo de ejecución de una rutina de cálculo de tarifa OM, se observa en la Figura **5-2**, donde la primera función abre la conexión a través del origen de datos ODBC, la segunda función ejecuta el comando SQL "Call Calc_Tnegocio_om('2017-03-15','2017-04-15',29,'Central');" , la tercera función obtiene los datos en forma de una tabla de datos tipo variante, luego se libera la memoria y por último se cierra la conexión. Los datos obtenidos por la función FECH ALL son convertidos a un array de string por medio de la función Variant to Data

A continuación se muestras en las Figuras **5-3** y **5-4**, si la rutina de cálculo de tarifa se fuese hecho directamente en LabVIEW:

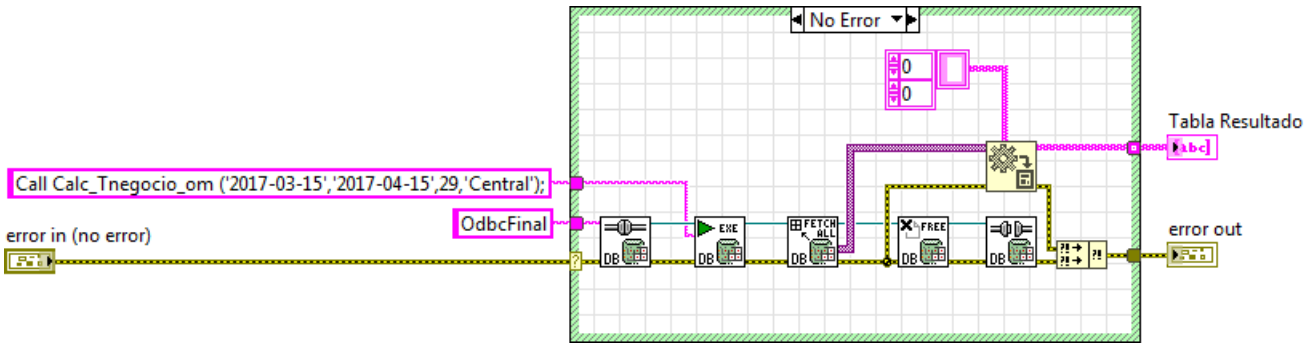


Figura 5-2.: Ejemplo de ejecución de sentencia SQL en LabVIEW con MySQL.

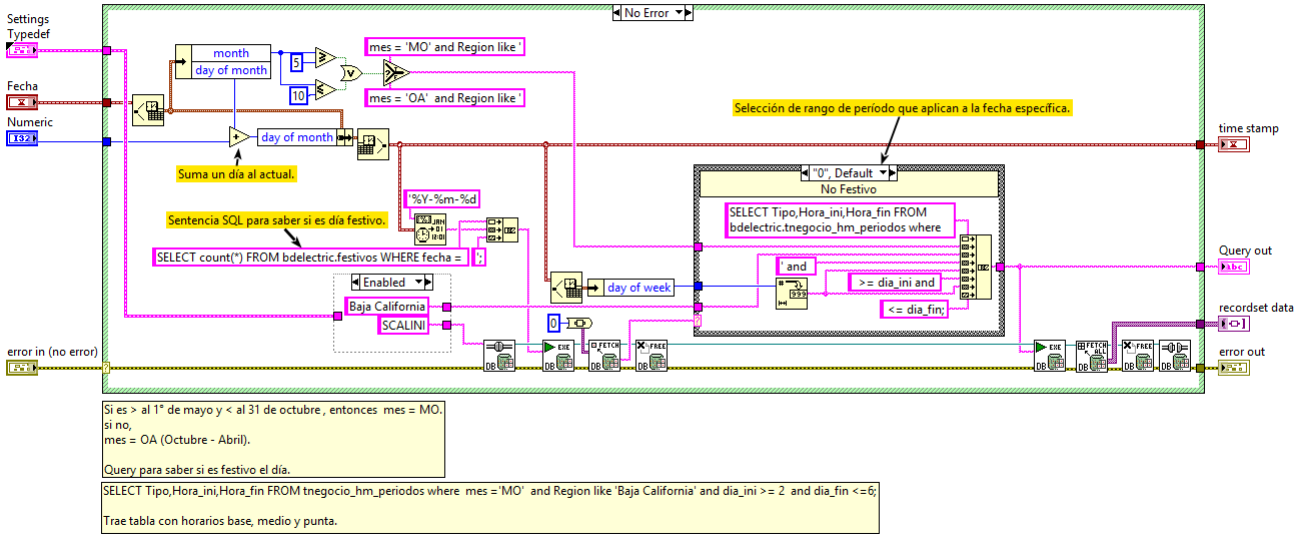


Figura 5-3.: Ejemplo de cogido en LabVIEW para cálculo por código manual (solo código LabVIEW) de tarifa negocio HM.

Este proyecto tuvo la participación de muchas personas, donde el aporte principal del autor de este trabajo fue el diseño de la base de datos, desarrollo de funciones y procedimientos de almacenas en la base de datos, desarrollo de sentencias SQL especializadas y ayuda a integración de código LabVIEW con MySQL. Y otras personas hicieron la programación de algoritmos en LabVIEW. Para observar con más profundidad los algoritmos realizados en LabVIEW puede consultar la tesis de grado [6], realiza por el Ingeniero Alfredo Rentería Villanueva y el M.I. Enrique Noe Arias. El software se denomino PIME (Plataforma Integral de Monitoreo Electrónico) usa una arquitectura basada en un manejador de mensajes en filas llamada “Continuous Measurement and Logging” (Medición y logeo continuo).

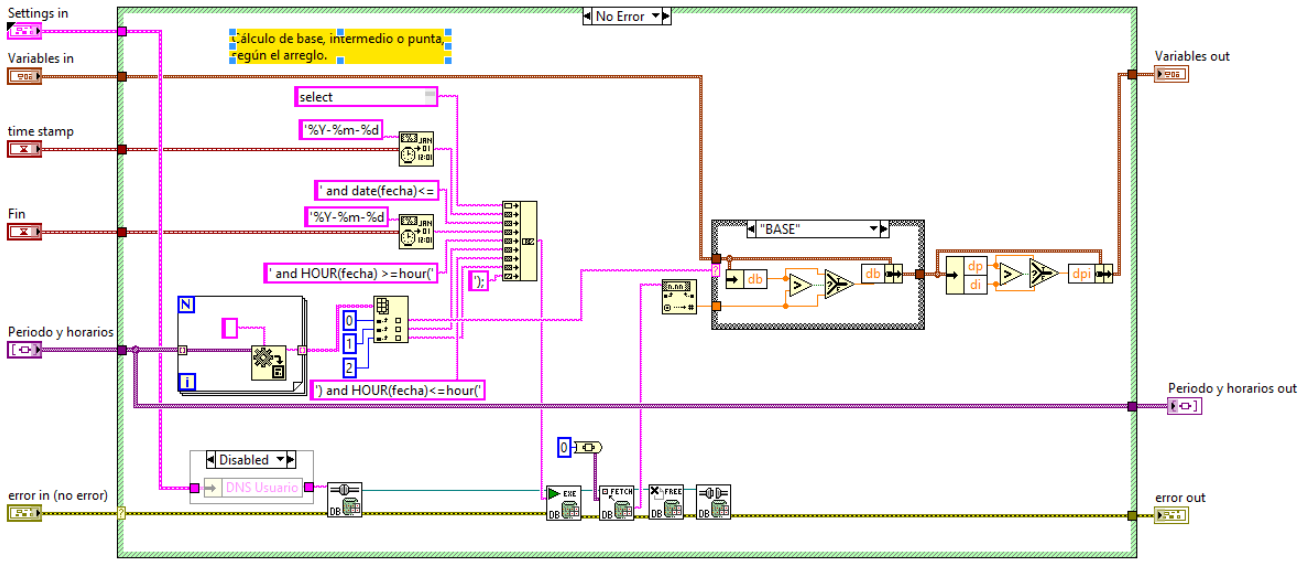


Figura 5-4.: Ejemplo de cogido en LabVIEW para cálculo de manda en periodo base, intermedio y punta por código manual (solo código LabVIEW) de tarifa negocio HM.

5.1. Rutina de Actualización de tarifas en tablas de MySQL

Se diseñaron las sentencias SQL y secuencia para actualización de tablas de tarifas, las cuales siguen la siguiente secuencia lógica.

a) Selección de archivo .zip donde están todos los archivos .txt con el nombre de la tablas a la cual se va actualizar

b) Descromprimir archivo .zip en carpeta temporal

c) Tomar archivo verificar que el nombre del archivo corresponde a un nombre de la tabla en la base de datos.

d) Crear tabla temporal con el igual número de columnas u nombre y tipo que la tabla de origen usando la SQL.

```
-- ejemplo de creacion tabla temporal apartir de
-- tabla TNEGOCIO_HM
DROP  TABLE IF EXISTS TMP;
CREATE  TABLE TMP LIKE TNEGOCIO_HM ;
```

e) Cargar datos de archivo .txt separado por comas a tabla temporal usando la sentencia SQL ejemplo:

```
LOAD DATA INFILE 'D:\\MopticaCIO\\Tesis\\Tarifas\\Negocio
\\Tnegocio_HM.txt' INTO TABLE tmp
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\r\n'
 IGNORE 1 LINES;
```

f) Actualizar tabla de interés con la SQL atravez de creación de SQL dinámicas usado SQL "Describe tablename" de la cual obtenemos nombre de columnas y se crea la SQL como la siguiente:

```
INSERT INTO TNEGOCIO_HM SELECT * FROM TMP T ON
DUPLICATE KEY UPDATE VALORKWDMAX = T.VALORKWDMAX,
PUNTA = T.PUNTA, INTERMEDIA = T.INTERMEDIA, BASE = T.BASE;
```

Con esta SQL y combinado con las llaves únicas, hace que MySQL actualicé la fila si ya está en la base de datos o que la inserte si no está. Haciendo mucho más escalable y mantenible el sistema de actualización de tarifas

g) Proceder a siguiente archivo desde paso c hasta el último archivo.

El código de LabVIEW para posterior integración al proyecto principal se puede observar a continuación:

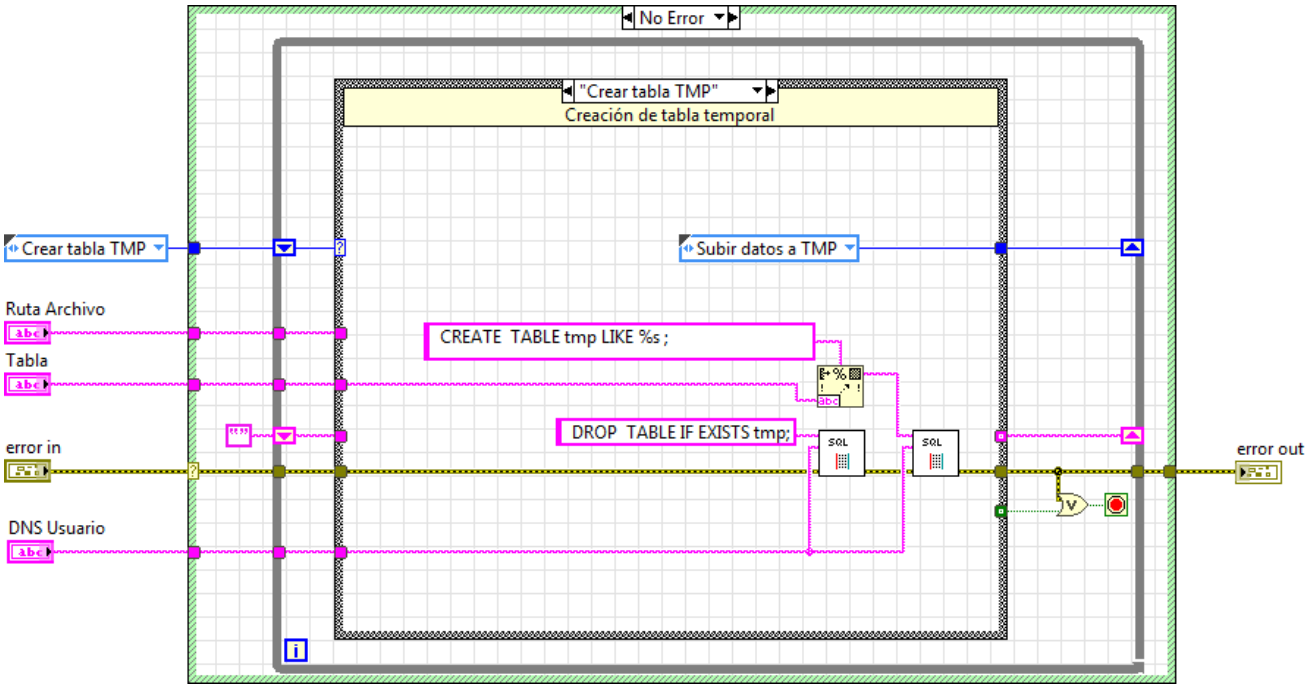


Figura 5-5.: Máquina de estados para actualización de tarifas paso 1. Crear tabla temporal

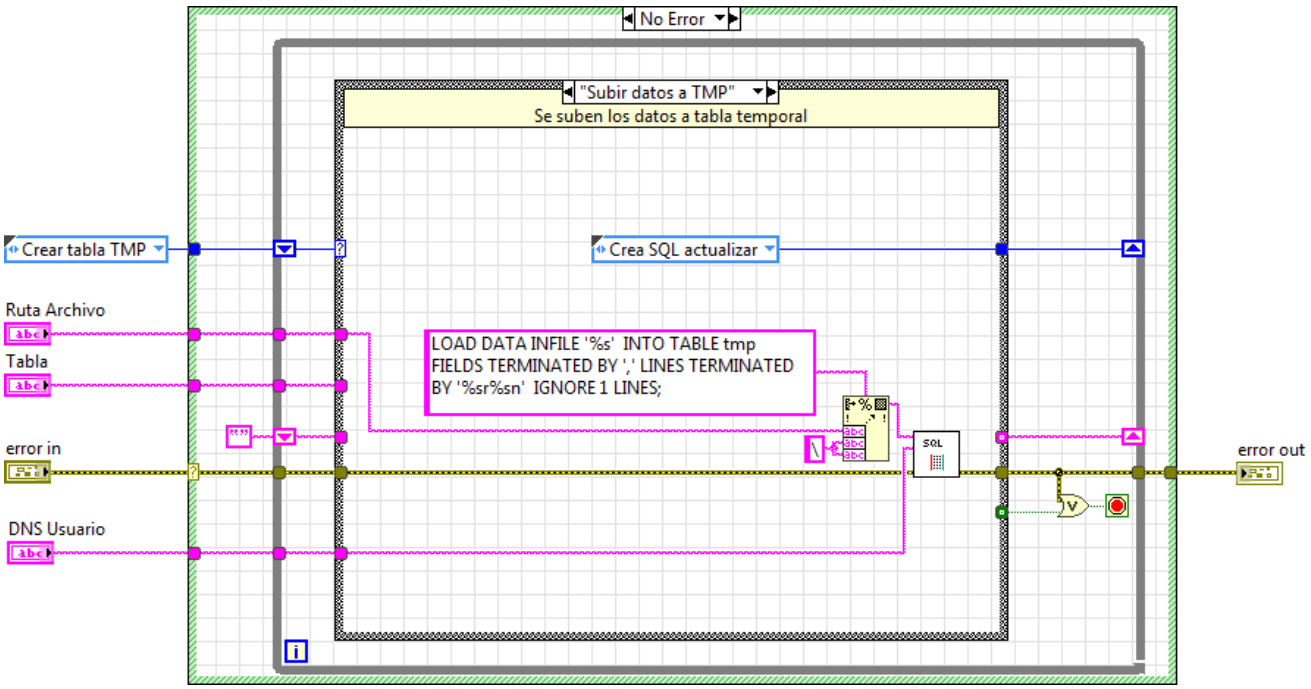


Figura 5-6.: Máquina de estados para actualización de tarifas paso 2. Subir datos a tabla temporal

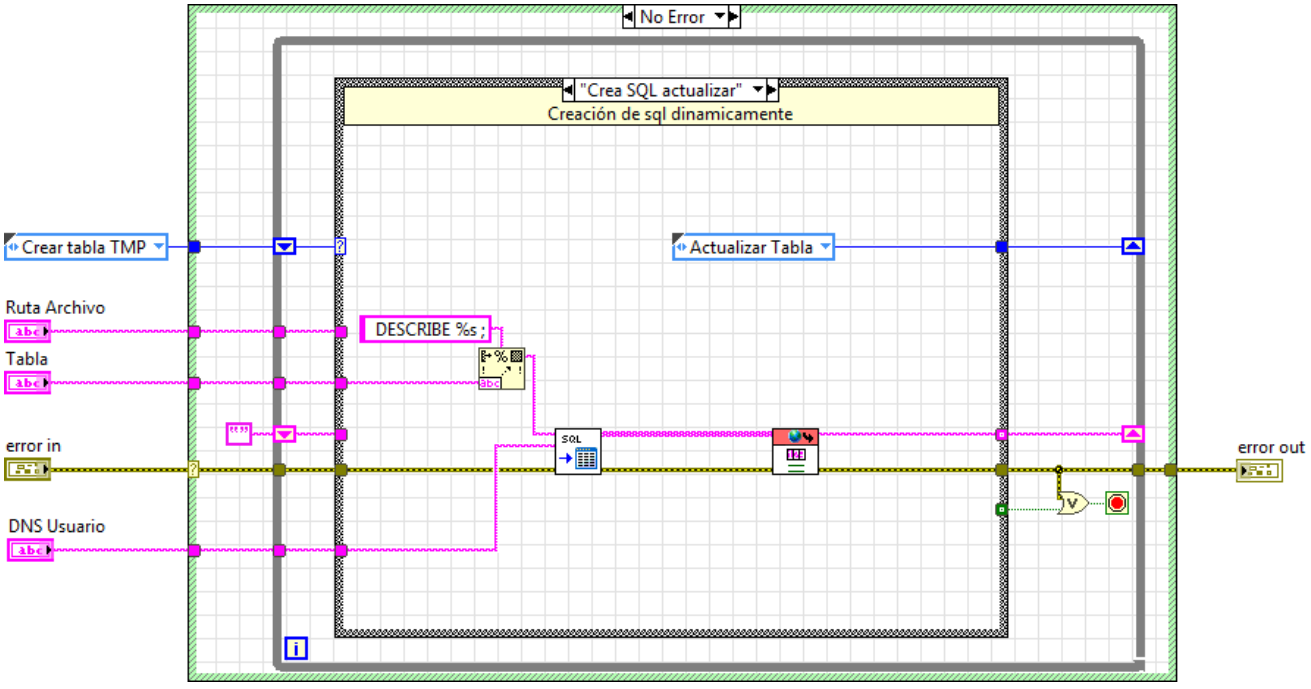


Figura 5-7.: Máquina de estados para actualización de tarifas paso 3. Crear SQL dinámica para actualizar tabla de tarifa.

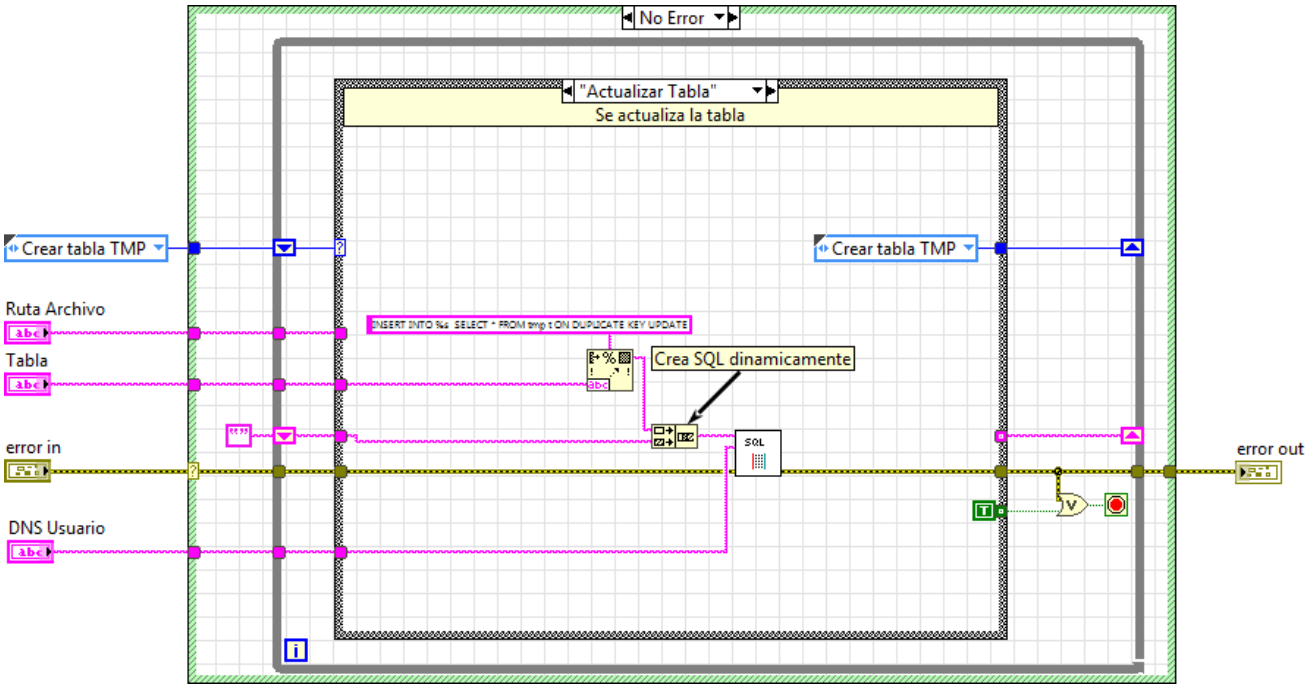


Figura 5-8.: Máquina de estados para actualización de tarifas paso 4. Ejecuta SQL dinámica para actualizar tabla de tarifas en la base de datos.

6. Implementación y resultados

En este capítulo se muestra la implementación y resultados del proyecto

6.1. Implementación

A continuación se muestra la implementación física del sistema de medición de parámetros eléctricos en industrias ESCALINI con el medidor PM850 como se observa en la Figura **6-1**. Donde se hicieron pruebas preliminares en el Centro de Investigaciones en Óptica.

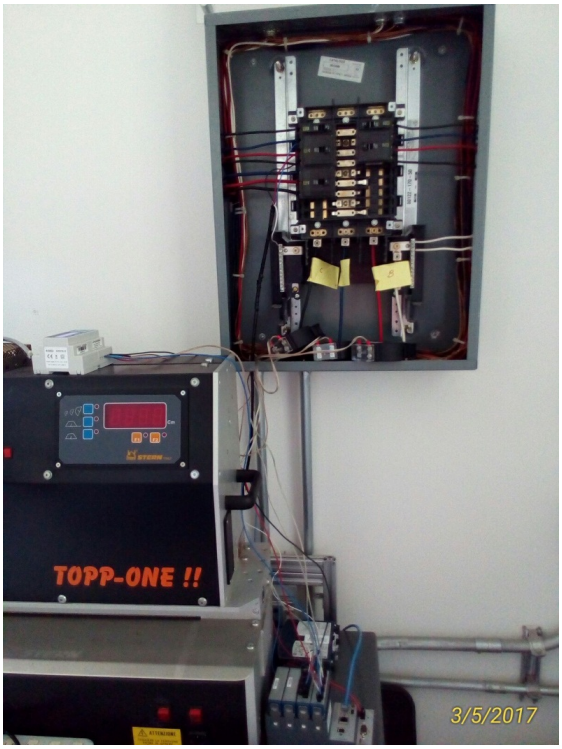


Figura 6-1.: Implementación de medidor S203TA-D, en red trifásica del laboratorio de electrónica Centro de investigaciones en Óptica.

En la Figura **6-2**. Se observa la implementación del medidor PM850 en industrias SCALINI.

Figura 6-2.: Distribución de la instalación del hardware en Industrias Scalini [6].

6.2. Resultados

- Se diseñó la base datos de todo el software PIME, donde se cubrieron aspectos como administración de usuarios donde se cifraron las claves usando la función AES_ENCRYPT para protegerlo de personas que puedan tener acceso a la base de datos, tablas de tarifas, tabla de errores, tabla de días festivos de México, tabla para almacenamiento de medición de parámetros eléctricos, tabla de configuración del sistema, tabla de alertas de usurario, procedimientos de almacenado, funciones etc.

- Se diseñaron las tablas para las principales tarifas de CFE, donde cada una de estas soporta actualización automática de la misma a través de un algoritmo desarrollado en LabVIEW el cual usa sentencias SQL especializadas para actualizar las tarifas y junto con la combinación de llaves únicas en las tablas hace que si el costo de la tarifa no existía lo crea y si ya existía lo actualiza de manera automática desde una archivo de texto separado por comas donde se ignora la primera fila, la cual se usa como información de que tipo de datos soporta cada campo. Con esto el usuario puede hacer consultas de costos de en un amplio rango de fechas siempre siempre y cuando la medición de parámetros eléctricos y tarifas estén actualizadas en la base de datos.

- Se implementó el cálculo de las principales tarifas usadas por CFE en procedimientos de almacenado donde con solo la poco parámetros de entrada (fecha de inicio, Fecha de fin del reporte, Region etc). El procedimiento consulta las diferentes tablas usadas por la tarifa y calcula los costos de la misma, factor de potencia, cargo por factor de potencia, cargos por medición en baja tensión etc. Devolviendo un desglosado del

cálculo de la tarifa.

- Se implementó un sistema de errores tipo custom que le permite a PIME informar al usuario que existen tarifas desactualizadas, ya que CFE para las tarifas de negocio para mediana y alta tensión los valores de las mismas son actualizados mes por mes en la página web www.cfe.gob.mx. Por lo tanto el software requiere una actualización de tarifas para poder hacer el cálculo.

- Se diseñaron las sentencias SQL especializadas de integración entre LabVIEW y la base de datos en MySQL, como por ejemplo encriptación y desencriptación de clave de acceso de usuarios, consultas de tarifas, actualización de tarifas, procedimientos y funciones de almacenado etc.

- Se registró el software desarrollado PIME ante registro público de derecho de autor ver apéndice 1.

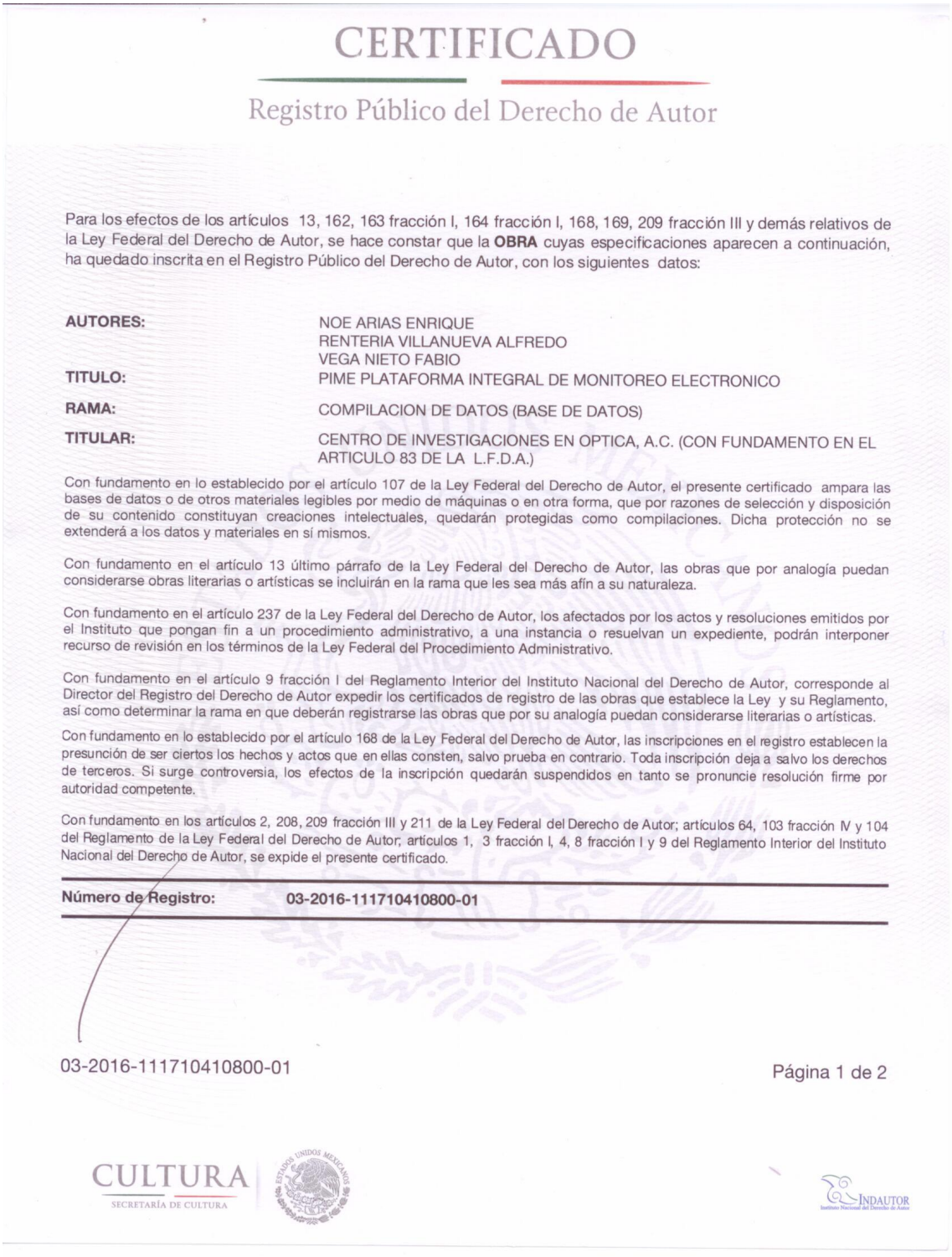
- Se obtuvo certificación de desarrollador asociado de LabVIEW Ver apéndice 2.

7. Conclusiones

Se logró diseñar e implementar una plataforma capaz de enlazarse a diferentes equipos medidores de variables eléctricas con comunicación Modbus RTU/TCP y concentración de datos correspondientes a los parámetros eléctricos en base de datos MySQL, para arrojar datos comparativos con la facturación de consumos eléctricos proporcionados por CFE, históricos de consumo, con la finalidad de fomentar la cultura de ahorro energético, planeación de mantenimiento y estrategias de consumo.
El sistema implementa un acceso jerárquico, que permite de acuerdo al tipo de usuario registrado realizar diferentes acciones en el software, como son el súper administrador, administrador, consultor.

Con este sistema hardware software Industrias SCALINI puede apuntar a incursionar en un mercado de negocios nuevo relacionados con sus actuales clientes de la industria del calzado y ofrecerles un nuevo producto/servicio que les ayudara a administrar su consumo energético y tomar medidas para disminución del mismo.

A. Registro software ante derecho Nacional de autor



CERTIFICADO

Registro Público del Derecho de Autor

Para los efectos de los artículos 13, 162, 163 fracción I, 164 fracción I, 168, 169, 209 fracción III y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la **OBRA** cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes datos:

AUTORES:	NOE ARIAS ENRIQUE RENTERIA VILLANUEVA ALFREDO VEGA NIETO FABIO
TITULO:	PIME PLATAFORMA INTEGRAL DE MONITOREO ELECTRONICO
RAMA:	COMPILACION DE DATOS (BASE DE DATOS)
TITULAR:	CENTRO DE INVESTIGACIONES EN OPTICA, A.C. (CON FUNDAMENTO EN EL ARTICULO 83 DE LA L.F.D.A.)

Con fundamento en lo establecido por el artículo 107 de la Ley Federal del Derecho de Autor, el presente certificado ampara las bases de datos o de otros materiales legibles por medio de máquinas o en otra forma, que por razones de selección y disposición de su contenido constituyan creaciones intelectuales, quedarán protegidas como compilaciones. Dicha protección no se extenderá a los datos y materiales en sí mismos.

Con fundamento en el artículo 13 último párrafo de la Ley Federal del Derecho de Autor, las obras que por analogía puedan considerarse obras literarias o artísticas se incluirán en la rama que les sea más afín a su naturaleza.

Con fundamento en el artículo 237 de la Ley Federal del Derecho de Autor, los afectados por los actos y resoluciones emitidos por el Instituto que pongan fin a un procedimiento administrativo, a una instancia o resuelvan un expediente, podrán interponer recurso de revisión en los términos de la Ley Federal del Procedimiento Administrativo.

Con fundamento en el artículo 9 fracción I del Reglamento Interior del Instituto Nacional del Derecho de Autor, corresponde al Director del Registro del Derecho de Autor expedir los certificados de registro de las obras que establece la Ley y su Reglamento, así como determinar la rama en que deberán registrarse las obras que por su analogía puedan considerarse literarias o artísticas.

Con fundamento en lo establecido por el artículo 168 de la Ley Federal del Derecho de Autor, las inscripciones en el registro establecen la presunción de ser ciertos los hechos y actos que en ellas consten, salvo prueba en contrario. Toda inscripción deja a salvo los derechos de terceros. Si surge controversia, los efectos de la inscripción quedarán suspendidos en tanto se pronuncie resolución firme por autoridad competente.

Con fundamento en los artículos 2, 208, 209 fracción III y 211 de la Ley Federal del Derecho de Autor; artículos 64, 103 fracción IV y 104 del Reglamento de la Ley Federal del Derecho de Autor; artículos 1, 3 fracción I, 4, 8 fracción I y 9 del Reglamento Interior del Instituto Nacional del Derecho de Autor, se expide el presente certificado.

Número de Registro: **03-2016-111710410800-01**

03-2016-111710410800-01

Página 1 de 2

CULTURA
SECRETARÍA DE CULTURA

INDAUTOR
Instituto Nacional del Derecho de Autor

CERTIFICADO

Registro Público del Derecho de Autor

La presente firma ampara el registro número: 03-2016-111710410800-01

México D.F., a 17 de noviembre de 2016

EL DIRECTOR DEL REGISTRO PÚBLICO DEL DERECHO DE AUTOR

JESUS PARETS GOMEZ

SECRETARÍA DE CULTURA
INSTITUTO NACIONAL DEL DERECHO DE AUTOR
DIRECCIÓN DEL REGISTRO PÚBLICO DEL DERECHO DE AUTOR

CULTURA
SECRETARÍA DE CULTURA

INDAUTOR
Instituto Nacional del Derecho de Autor

B. Certificación CLAD de Labview

Serial Number: 100-316-16326
Issue Date: 11/29/2016
Expiration Date: 11/28/2018

NI CUSTOMER EDUCATION

Certification

Fabio Vega Nieto

Has successfully completed all requirements and is now granted the title of:

NATIONAL INSTRUMENTS LabVIEW

CERTIFIED ASSOCIATE DEVELOPER

James J. Truchard
President and CEO
National Instruments

NATIONAL INSTRUMENTS

Bibliografía

[1] Morales Vizuete Diana Carolina (2011). Diseño e implementación de un sistema de monitoreo mediante la telemedicion del consumo de energía eléctrica de clientes espéciales, de la empresa regional ambatos regional centro norte S.A. (Tesis de pregrado). ESCUELA POLITÉCNICA NACIONAL, Ecuador.

[2] Armijos Abril Jairo Andrés, Pesántez Alvarado Álvaro Daniel (2016). Diseño De Un Medidor Inteligente Con Funciones De Respuesta A La Demanda En Infraestructuras De Medición Avanzada (Tesis de pregrado). UNIVERSIDAD DE CUENCA, Cuenca, Ecuador.

[3] Valero Alarcón Fran Alberto (2006). Dispositivo remoto para medición y monitoreo de consumo de energía eléctrico (Tesis de pregrado). UNIVERSIDAD SIMÓN BOLÍVAR, Colombia.

[4] Abonza Covarrubias Javier (2008). Sistema de supervisión, control y adquisición para el ahorro de energía (Tesis de Maestría). INSTITUTO POLITECNICO NACIONAL, México.

[5] Manual Mysql 5.0, Consultado 29 de marzo de 2017, disponible en https://downloads.mysql.com/docs/refman-5.0-es.pdf.

[6] Alfredo Rentería Villanueva (2016), Implementación de capa de conectividad LabVIEW – MySQL para el proyecto de venta del Centro de Investigaciones en Óptica e Industrias Scalini llamado "PIME".(Tesis de Pregado), Universidad de la Salle bajio, León, Guanajuato.

[7] hoja de dato PM850, consultados 1 de agosto de 2016, disponible en http://static.schneider-electric.us/docs/Power %20Management/PowerLogic %20Products /Metering/PM800 %20series %20power %20meters/63230-500-210A2.pdf.

[8] hoja de dato S203TA-D, consultado 1 de agosto de 2016, disponible en: https://www.seneca.it/media/2188/mi004075-e.pdf .

[9] Tarifas de Negocio comisión federal de electricidad CFE, consultada 1 agosto de 2016, disponible en: http://app.cfe.gob.mx/Aplicaciones/CCFE/Tarifas/Tarifas/tarifas_negocio.asp