Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES

Pedro Enrique Cruz López
Asesorado por el Ing. Edgar René Ornelis Hoíl

Guatemala, septiembre de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DEL LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

PEDRO ENRIQUE CRUZ LÓPEZ
ASESORADO POR EL ING. EDGAR RENÉ ORNELIS HOÍL

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, SEPTIEMBRE DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

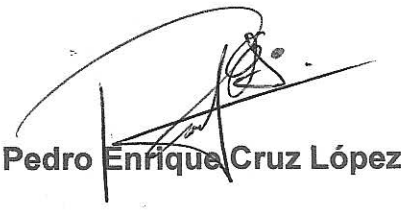
DECANO	Ing. Angel Roberto Sic García a.i.
EXAMINADOR	Ing. Cesar Augusto Fernández Cáceres
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. William Estuardo Escobar
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 31 de agosto de 2016



Pedro Enrique Cruz López

Guatemala 27 de octubre de 2017

Ingeniero
Marlon Antonio Pérez Turk
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Marlon Antonio Pérez Turk:

Me complace saludarle, haciendo referencia al trabajo de graduación titulado **"IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES"**, desarrollado por el estudiante universitario Pedro Enrique Cruz López con número de carné **201114233** y DPI **2143 07514 0101**, que como asesor apruebo el contenido del mismo.

Para su conocimiento y efectos, sin otro particular, me suscribo.

Ing. Edgar René Ornelis Hoíl
Colegiado 4830
Asesor

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 15 de Noviembre de 2017

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **PEDRO ENRIQUE CRUZ LÓPEZ** con carné **201114233 y CUI 2143 07514 0101,** titulado **"IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERÍA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES",*** *realizado por el estudiante, PEDRO ENRIQUE CRUZ LÓPEZ aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 03 de septiembre de 2018

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.322.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES EN EL KERNEL DE LINUX PARA LA MEJORA DEL RENDIMIENTO DE APLICACIONES,** presentado por el estudiante universitario: **Pedro Enrique Cruz López,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

Guatemala, Septiembre de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/cc

ACTO QUE DEDICO A:

Mis padres	Sandra López y Edgar Cruz por su amor y apoyo incondicional durante mi carrera
Mi hermana	Marcela Cruz por estar conmigo en los buenos y malos momentos, su amor y apoyo fueron vitales.
Mis abuelos	Herlinda Ismatul y Pedro López que en vida me brindaron todo su cariño y apoyo, son dos pilares importantes en mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por permitirme tener el honor de estar en sus aulas y ser mi casa de estudios durante varios años.
Facultad de Ingeniería	Por permitir formarme como profesional, por las experiencias y los momentos inolvidables que viví adentro y fuera de sus aulas, los cuales quedaran grabados en mi memoria por siempre.
Mi asesor de trabajo	Ingeniero Edgar Ornelis, por su ayuda incondicional y proporcionar sus conocimientos profesionales para este trabajo.
Mis amigos	Leonel Santizo, Tomas Irving, Juan Diego Lacayo, Iván Cabrera por su amistad y apoyo durante estos años
Mis amigos de la Facultad	Por su amistad y compañerismo, los momentos con ustedes fueron únicos.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
/usr/bin	Directorio que contiene todos los archivos ejecutables de un sistema operativo.
.h	Extensión de archivos de cabecera en lenguaje C.
.c	Extensión de archivos escritos en lenguaje C.
$	Indicador de línea de comandos.
s	Segundo.

GLOSARIO

ACID — ACID es un grupo de 4 propiedades (Atomicidad, consistencia, durabilidad y aislamiento) que garantizan que las transacciones en las bases de datos se realicen de forma confiable.

API — Abreviado por sus siglas en inglés: *Application Programming Interface,* es un conjunto de subrutinas que ofrece determinada biblioteca para poder ser usado por otro software.

Assembly — Lenguaje de bajo nivel para computadoras o cualquier dispositivo programable.

Base de datos — Una base de datos es el conjunto de datos informativos organizados en un mismo contexto para su uso y vinculación.

Benchmark — Conjunto de pruebas para medir el rendimiento de una aplicación.

C — Lenguaje de programación desarrollado por Dennis Ritchie en la década de los años 1970.

DBMS	Por sus siglas en inglés: *Data Base Management System*, es un programa o conjunto de programas que se encargan de administrar las operaciones y los accesos a una base de datos.
DDL	*Data Definition Language* por sus siglas en inglés, es un lenguaje proporcionado por el DBMS para la definición de estructuras de almacenamiento o procedimientos.
DML	*Data Manipulation Language* por sus siglas en inglés, es un lenguaje proporcionado por el DBMS para llevar a cabo tareas de consulta o modificación de datos.
Driver	Programa que administra las acciones de un determinado hardware.
InnoDB	Es un mecanismo de almacenamiento de datos de código abierto para la base de datos MySQL, incluido como formato de tabla estándar en todas las distribuciones de MySQL AB a partir de las versiones 4.0.
Int	Tipo de dato primitivo en programación que representa un entero regularmente de 4 bytes.

Kernel	Es un programa que constituye la parte fundamental de un sistema operativo. En él se administra la memoria, sistemas de archivos y es el medio de comunicación directa entre el software y el hardware.
Módulo de kernel	Archivo que contiene código, el cual es de utilidad para ampliar la funcionalidad del kernel en tiempo de ejecución.
MySQL	Sistema de gestión de base de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation.
PostgreSQL	Sistema de gestión de base de datos relacional y orientado a objetos de código abierto.
SQL	*Structured Query Language* por sus siglas en inglés, es un lenguaje que maneja el acceso y definición a un sistema de base de datos relacional.
UDF	Del inglés, *User Defined Function*, es una función que se ejecuta en un sistema de base de datos, cuyo comportamiento y definición es desarrollada por el usuario.
Unsigned int	Tipo de dato primitivo en programación que representa a un entero sin signo.

Usuario *root*	Es un rol en un sistema operativo el cual no posee permisos restringidos para ejecutar acciones en el mismo.

RESUMEN

El presente proyecto consiste en una investigación y aplicación de un método para crear una opción alternativa a la implementada en la actualidad en cuanto a claves subrogadas.

En el primer capítulo se brinda un marco teórico y un conjunto de conceptos necesarios para poder llevar a cabo su implementación. Se explican las bases de datos y sus componentes así como el núcleo del sistema operativo Linux y su arquitectura, la cual es utilizada como base para implementar la arquitectura de software.

El segundo capítulo proporciona un análisis de las implementaciones actuales de claves subrogadas en *MySQL* y *PostgreSQL*, así como un análisis de la solución y el listado de requerimientos funcionales, no funcionales y casos de uso de la arquitectura a implementar.

En el tercer capítulo se define la arquitectura, se hace un análisis de la misma y se explica cada capa que forma parte de la solución final, se detalla cada componente creado y su funcionalidad como parte de dicha arquitectura, se presentan diagramas y gráficos.

En el capítulo cuatro, está dedicado a pruebas de rendimiento, mostrando resultados y gráficos, manual de usuario para el uso correcto del software y descripciones técnicas.

OBJETIVOS

General

Implementar una alternativa para la generación de claves subrogadas o campos automáticos que funcione en *MySQL Server* y *PostgreSQL*.

Específicos

1. Desarrollar un método para generar campos y claves únicas que pueda ser administrado sin la necesidad de consultar datos de un DBMS.

2. Eliminar la necesidad de crear índices para campos que deben ser generados automáticamente por la base de datos.

3. Reducir operaciones de borrado e inserción en tablas que contengan campos con claves subrogadas debido a los índices.

INTRODUCCIÓN

En la actualidad cualquier solución de software necesita ser optimizada en el mayor grado posible para la satisfacción de sus clientes. La persistencia de la información es uno de los mayores candidatos para calibrar el rendimiento de una aplicación. El presente trabajo se enfoca en la optimización de operaciones de inserción, borrado y actualización de datos para una solución de software especifica.

Existen muchas maneras de optimizar las bases de datos, desde mejora de consultas hasta uso de índices, sin embargo algunos métodos permiten mejorar el rendimiento pero afectan a otros atributos de calidad, el propósito de este trabajo consiste en mejorar y afinar el uso de índices en las bases de datos relacionales.

El presente proyecto muestra una alternativa a las claves subrogadas de los DBMS implementadas de manera nativa. En los siguientes capítulos se hará un análisis detallado y una explicación de cada componente de la arquitectura utilizada, utilizando gráficos, tablas y toda información necesaria para su absoluta comprensión y se cuenta con un manual para su uso. Como se describe en el capítulo cuatro, este trabajo no solo está orientado a bases de datos, también puede utilizarse para otros fines.

1. ASPECTOS GENERALES

En este capítulo se presentan los conceptos generales necesarios para el entendimiento del presente trabajo, que incluye el funcionamiento de un sistema de base de datos y su relación con el sistema operativo, especialmente Linux.

1.1. Historia de las bases de datos

Los inicios de las bases de datos se remontan al siglo XIX, cuando Herman Hollerith, un ingeniero y estadístico estadunidense considerado como el primer informático de la historia, desarrolla una máquina tabuladora para llevar a cabo el censo poblacional de 1890. Hollerith haciendo uso de tarjetas perforadoras para almacenar datos que aplicaban la lógica booleana, se convirtió en el primero en el tratamiento automático de la información.

En 1970 surgió el término *relacional* acuñado a las bases de datos, gracias a Edgar Codd y su artículo “Un modelo relacional de datos para grandes bancos de datos compartidos”, en el cual detalla el funcionamiento de un modelo relacional que trataría mejor los datos y que serviría como base para la construcción de los sistemas de gestión de bases de datos actuales (SGBD).

En la década de los ochenta se desarrolló el lenguaje de consulta estructurado o mejor conocido como SQL (acrónimo en inglés de *Structured Query Language*) diseñado por Donald D. Chamberlin, el cual permite interactuar con las tablas relacionales, ingresar datos, actualizar y modificar la base de datos de manera simple. Durante la década de los noventa el lenguaje SQL sufrió variantes que añadían consultas recursivas, procedimientos y disparadores o *triggers* que ayudan a un mejor tratamiento de la información

contenida en las tablas. En este trabajo únicamente se tratará con bases de datos relacionales.

1.2. Sistemas de bases de datos

C.J. Date (2001) en su libro *Introducción a los sistemas de bases de datos* define a un sistema de base de datos como: “Un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información en base a peticiones”. En efecto ese sistema computarizado que define Date, corresponde a un conjunto de programas y componentes que administran la interacción con la base de datos física de un sistema. El software encargado de definir datos, administrarlos y gestionar la comunicación con el usuario es el sistema de gestión de base de datos o como se conoce por sus siglas en inglés: *Database Managment System*, DBMS.

1.3. Vista general del modelo relacional

El modelo relacional se trata de un modelo teórico matemático basado en algebra relacional que proporciona bases para un modelado de relaciones que incluyen operadores.

1.3.1. Definición de tabla

“Una tabla se define como un conjunto de tuplas que contienen un conjunto de valores atómicos”[1]. Las tablas contienen un número de columnas y a cada una de ellas representa un atributo y este está identificado con un nombre, el cual debe ser único.

[1] Juan Ramón López (2011). *Bases de datos*

1.3.2. Clave primaria

Una clave primaria, es un campo o varios que representa a un registro de manera única, por lo que no deben existir dos registros con el mismo valor de clave primaria. Una llave primaria puede ser una columna existente en la tabla o una columna generada por la base de datos acorde a una secuencia definida.

1.3.3. Clave natural

La clave natural es un campo formado por atributos de la entidad de dominio, como ejemplo se pueden tomar el número de DPI de un ciudadano guatemalteco, número de factura, número de serie etc.

1.3.4. Clave sustituta

Una clave sustituta es un campo que el sistema genera automáticamente, dando un valor numérico o alfanumérico en una columna de una tabla cuando se inserta un nuevo registro. Las claves sustitutas, en ocasiones llamadas subrogadas o secuencias incrementales son a menudo utilizadas en campos de llave primaria ya que garantiza la unicidad del campo y evita violaciones de integridad referencial. Las implementaciones en los diferentes DBMS varían.

1.4. Sistema operativo GNU/Linux

Es uno de los sistemas operativos más escalables y utilizados en la actualidad gracias a su alto rendimiento. A continuación se da una breve historia y se explican algunos conceptos.

1.4.1. Breve reseña

"Linux fue desarrollado en 1991 por Linus Torvalds, un ingeniero de software finlandés, como un sistema operativo para las computadoras personales IBM que se basaban en el procesador Intel 80386"[2]. Linus mantuvo un intenso trabajo sobre su sistema operativo mejorando y actualizando componentes tratando de que el sistema fuera compatible con una variedad de arquitecturas.

El núcleo o *kernel* de Linux está basado en Minix, un sistema operativo pequeño de Unix, desarrollada por Andy Tanenbaum. Hacia finales de agosto de 1991, Linus había escrito la versión 0.01 que tenía un pequeño kernel con funcionalidades básicas de lectura y escritura de archivos en un disquette. El nuevo sistema operativo tiene el nombre de "Linus Unix" y luego cambio a "Linux".

La primera versión era muy básica y no fue anunciada, en octubre de ese mismo año se anunció la primera versión oficial, la 0.02, que incluía lectura de comandos de usuario y un compilador del lenguaje C. Linus brindó el código fuente para que cualquiera pudiera leerlo y modificarlo, así que en la versión 0.02 se implementaron mejoras y más programas útiles.

Tras varias versiones y gracias a la colaboración de varios desarrolladores, el diciembre de 1993 nace Linux 1.0, siendo la primera versión estable y sin errores del sistema operativo.

Hoy en día existen una variedad de distribuciones basadas en el kernel de Linux y se han convertido en uno de los principales sistemas operativos.

[2] Daniel P. Bovet y Marco Cesati (2006). P.1 *Understanding The Linux Kernel*

1.5. Kernel de Linux

El kernel o núcleo es el programa principal del sistema operativo, es cargado en memoria RAM cuando el sistema inicia y contiene procedimientos críticos que son necesarios para que el sistema pueda operar. Según los autores Daniel P. Bovet y Marco Cesati (2006), El kernel debe cumplir dos objetivos fundamentales:

- Interactuar con los elementos del hardware.
- Proveer un ambiente de ejecución a las aplicaciones que se corren en la computadora.

Algunos sistemas operativos como *MS-DOS* permiten la interacción directa del espacio de usuario con el hardware, sin embargo los sistemas operativos basados en *UNIX* no permiten esta interacción y en su lugar, el usuario debe hacer una solicitud para obtener o manipular los recursos de hardware. El kernel de Linux está escrito en su mayoría en lenguaje C y código *assembly*.

1.5.1. Arquitectura del Kernel de Linux

El kernel de Linux presenta una arquitectura modular, la cual, a diferencia de un kernel monolítico, puede extender su funcionalidad sin necesidad de compilar el kernel por completo, tal flexibilidad permite ampliar el código del kernel de Linux según sean las necesidades.

1.5.2. Módulos de kernel

La arquitectura modular, ofrece los módulos de kernel o también conocidos como *Loadable Kernel Module* (LKM), los cuales son objetos cuyo código puede ser añadido o removido del kernel en tiempo de ejecución sin afectar su funcionamiento. El código de los módulos de kernel puede añadir funcionalidades en la capa superior del kernel, sistemas de archivos, añadir un nuevo dispositivo entre otras. Al igual que el kernel, los módulos se escriben en lenguaje C.

1.5.2.1. Device drivers

Es un tipo de modulo que toma un rol especial en el kernel de Linux. Son definidos como distintas “cajas negras” que hacen que una pieza de hardware responda a una interfaz de programación bien definida.

1.5.2.1.1. Char drivers

Es un tipo de *device driver* que particularmente no necesita estar ligado a un hardware en específico, puede actuar con independencia. Principalmente se utilizan para enviar y recibir información del kernel al usuario o viceversa. Vale la pena mencionar que los todos los módulos construidos en este proyecto son de este tipo.

1.5.3. Llamadas al sistema

Una llamada al sistema, *system call* o *syscall* es una solicitud explicita del usuario al kernel hecha por medio de una interrupción de software. Cuando un proceso invoca a una syscall, el procesador cambia de modo usuario a modo

kernel y comienza la ejecución de una función de kernel. A través de llamadas al sistema se logra la comunicación entre el kernel y los procesos de usuario, las llamadas al sistema que se utilizan en la solución son:

- *ioctl*: Es una abreviación de la frase *input/output control*, esta llamada permite a un proceso de usuario, controlar y comunicarse con un *driver* (módulo de kernel).
- *open:* Permite abrir y crear ficheros en el disco duro.

Estas llamadas al kernel, son las principales para establecer comunicación entre el kernel y los distintos procesos de usuario que consumen los datos necesarios.

1.5.3.1. IOCTL

Abreviación de la frase en inglés *Input / Output Control.* Es una llamada al sistema de Unix que permite comunicarse con un *driver* del sistema y escribir o leer información de él. La operación a realizar dependerá del parámetro enviado al método.

Ioctl recibe como parámetros un descriptor de archivos abierto, un número de tipo *unsigned long* el cual determina la acción a realizar y un puntero *void* de datos que se utiliza para almacenar la información de cierto tipo proveniente de la llamada o enviar información al *driver.* Cabe destacar que el prototipo de esta llamada al sistema puede tomar un número variable de parámetros pero para fines de esta arquitectura solo se utiliza la versión disponible para el espacio de usuario, la cual utiliza solo los parámetros descritos anteriormente.

1.6. Librería de objetos compartida

Una librería de objetos compartida, conocida por su abreviación *SO* del inglés *shared object library*, son librerías cargadas por los programas cuando se inician. Cuando estas librerías son instaladas correctamente, puede ser utilizada por otros programas después de iniciarse, estas librerías son indispensables para proporcionar código nuevo al DBMS.

1.7. Linux y sistemas de gestión de bases de datos

Dada la eficiencia y costo nulo de los sistemas operativos Linux, combinar un DBMS con Linux se ha vuelto parte de una tendencia en los últimos años. Las infraestructuras de internet actuales incluyen en su mayoría una distribución Linux combinada con otras tecnologías de código abierto. A mediados de la década de los noventa, surgió el término “pila de software” que básicamente es un grupo de programas que son combinados para un fin específico, primordialmente en aplicaciones web. En 1995 surge PHP, un lenguaje de programación para ejecución del lado del servidor, y el servidor web Apache HTTP. En 1996 nace *MySQL Server*, un sistema de gestión de bases de datos relacional, y da origen a la pila de software LAMP, acrónimo de las iniciales de cada tecnología combinada (Linux, Apache, MySQL y PHP). LAMP es usada para desarrollar aplicaciones web de propósito general y comenzó a ser tendencia a finales de la década de los noventa cuando muchas empresas decidieron desarrollar sus sitios en tecnologías de código libre por razones de presupuesto.

LAMP no es la única pila de software, con el surgimiento de redes sociales, el crecimiento de la información y los servicios en la nube, surgen otras combinaciones de tecnologías como MEAN, SLAMPP, MAMP entre otras. La mayoría de pilas de software que involucran sistemas de bases de datos

relacionales, incluyen a las distribuciones de Linux como sistema operativo base. A continuación se presenta una tabla de las principales pilas de software con DBMS relacional:

Tabla I. **Pilas de software más comunes que usan SO Linux como base**

Acrónimo	**Sistema operativo**	**Servidor web**	**DBMS**	**Programación en el servidor**
LAMP	Linux	Apache	MySQL	PHP / Perl / Python
LAPP	Linux	Apache	PostgreSQL	PHP / Perl / Python
LYME	Linux	Yaws	Mnesia	Erlang
SLAMPP	Linux booteable desde CD-ROM	Apache	MySQL	PHP / Perl / Python / Rubi / Lua

Fuente: elaboración propia.

2. IDENTIFICACIÓN DEL PROBLEMA Y ANÁLISIS DE LA SOLUCIÓN

Algunas rutinas en almacenamiento secundario para recaudar información acerca del estado de una tabla en los DBMS son algunas operaciones que normalmente afectan el rendimiento cuando se ejecutan múltiples operaciones en pequeñas fracciones de tiempo. En este capítulo se abordan el análisis de un posible problema y la evaluación de sus soluciones.

2.1. Implementaciones de claves subrogadas en MySQL Server y PostgreSQL

A continuación se da un resumen de la implementación de claves sustitutas en *MySQL Server* y *PostgreSQL*.

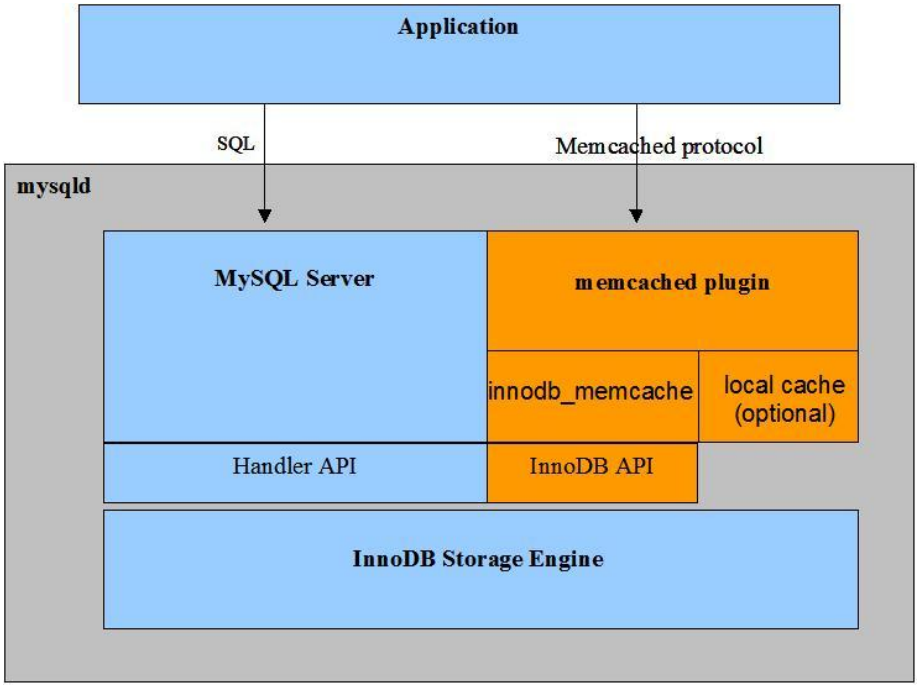
2.1.1. MySQL Server

La implementación de *MySQL* se basa principalmente en el almacenamiento en memoria primaria e índices.

2.1.1.1. Mecanismo de almacenamiento InnoDB

MySQL utiliza el mecanismo *InnoDB* el cual soporta transacciones de tipo ACID (acrónimo de *Atomicity*, *Consistency*, *Isolation* and *Durability*: Atomicidad, Consistencia, Aislamiento y Durabilidad en español) e integridad referencial. InnoDB realiza bloqueos a nivel de fila y proporciona funciones de lectura sin bloqueo lo que permite gestionar múltiples usuarios de manera simultánea.

Figura 1. **Diagrama general de la arquitectura interna de *MySQL***



Fuente: https://dev.mysql.com/doc/. Consulta: 11 de octubre de 2017.

2.1.1.2. Inicialización de contadores incrementales

Cuando se especifica en una tabla el atributo *AUTO_INCREMENT*, la tabla maneja un contador especial en el diccionario de datos de InnoDB denominado contador incremental, el cual es almacenado en memoria principal.

Después de la inicialización del servidor, *InnoDB* ejecuta una sentencia para obtener el mayor valor de la tabla para la primera inserción, luego aumenta el valor obtenido (por defecto en 1) y lo asigna al contador incremental en memoria principal asociado a la tabla. El valor es retenido siempre que el servidor este corriendo, cuando el servidor se detiene, *InnoDB* realiza el mismo proceso en cada tabla para reinicializar los valores. Cabe resaltar que la columna que tenga el atributo *AUTO_INCREMENT* debe ser definida como

parte de un índice para que sea posible realizar un indexado de búsqueda para obtener el máximo valor.

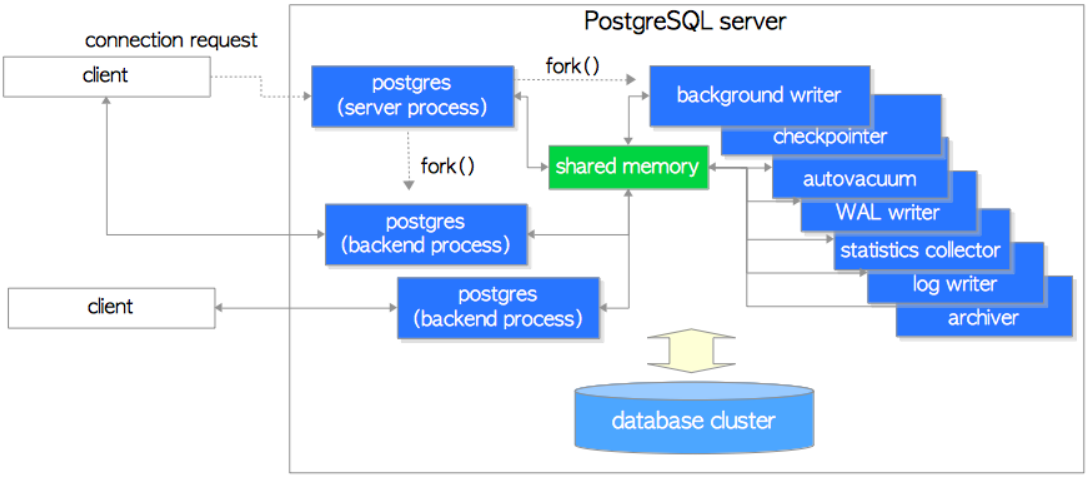
2.1.2. PostgreSQL

En *PostgreSQL* las secuencias se manejan de manera distinta, no existe el atributo *AUTO_INCREMENT* de *MySQL* o *IDENTITY* de *SQL Server*, se utiliza el atributo *SERIAL* para simular una columna con un incremento numeral automático, de manera implícita si se utiliza el atributo *SERIAL* o explícita si crea un objeto generador de secuencias de forma manual, se crea un objeto en la base de datos denominado *SEQUENCE*, el cual es definido desde una sentencia DDL.

El objeto *SEQUENCE* es almacenado, al igual que los datos, en tablas y sus valores pueden ser accedidos por medio de DML. A diferencia de la implementación de *MySQL*, el objeto *SEQUENCE* permite que el usuario especifique si el valor será almacenado en memoria principal para un rápido acceso o si no lo estará. Cabe resaltar que el objeto *SEQUENCE* es una *pseudocolumna*, lo que significa que no se pueden hace inserciones a sus valores.

Para su utilización en una operación de inserción, el usuario debe escribir en la sentencia el nombre del objeto seguido del atributo *NEXTVAL*, que corresponde al siguiente valore de la secuencia.

Figura 2. **Diagrama de la arquitectura interna de *PostgreSQL***



Fuente: http://www.interdb.jp/pg/pgsql02.html#_2.1. Consulta: 11 de octubre de 2017.

2.2. Ventajas del uso de claves sustitutas

- Mientras un campo exista, el valor de la clave no cambiará, esto permite que no se pierda la referencia.
- Evita utilizar más de un campo para utilizarlo como llave primaria, lo que permite la agilidad en las consultas, comparado con tablas con llaves primarias de varios campos.
- Se facilita el proceso de inserciones masivas

2.3. Desventajas del uso de claves sustitutas

A continuación se presentan algunas desventajas en los DBMS del uso de atributos auto-incrementales o claves sustitutas.

2.3.1. Desventajas generales

En el diseño de una solución que utiliza bases de datos relacionales, siempre existe un debate acerca del uso de claves naturales o claves candidatas para una llave primaria.

- Se requiere de un índice adicional y esto representa costo en espacio de disco duro y por consiguiente operaciones de inserción y actualización más lentas.
- La implementación de claves sustitutas o auto-incrementales es diferente en cada DBMS.
- Puede darse el caso de duplicados con claves naturales dado que estas no los previenen.

2.3.2. Desventajas en MySQL Server

MySQL Server administra las claves sustitutas lo más eficiente posible sin embargo posee algunas desventajas:

- El campo *AUTO_INCREMENT* debe ser definido necesariamente como parte de una clave cuando en algunas ocasiones este campo no necesariamente es requerido como tal.
- La operación de borrado es más lenta dado el índice extra
- A pesar de almacenar las variables de contadores en memoria primaria, se requiere un proceso de inicialización de contador por medio de una consulta cada vez que se reinicia el servidor, lo que implica un impacto en los primeras operaciones de inserción.
- Espacio adicional por tabla debido a los índices extras.

2.3.3. Desventajas en PostgreSQL

Como ya se describió en la primera sección de este capítulo, la implementación de claves sustitutas en *PostgreSQL* se hace mediante objetos llamados SEQUENCE. A continuación se presentan algunas desventajas de su uso:

- Costo adicional en disco dado que son objetos de la base de datos
- Si se desea automatizar el proceso de inserción con auto-incrementales utilizando *SEQUENCE*, es necesario definir triggers en cada tabla.
- Proceso de inserción más lento si la secuencia no se creó con la opción CACHE activada dado que los valores son leídos desde disco.
- Los valores que se almacenan en cache por parte de la secuencia desaparecen cada vez que el servidor es apagado o reiniciado, un problema similar al de *MySQL Server*.

2.4. Requerimientos funcionales del sistema

En esta sección se describen los requerimientos funcionales del sistema para resolver las desventajas anteriormente descritas:

Código	RF1	Prioridad	Alta
Nombre	Mecanismo automático de números		
	La solución debe aportar un mecanismo automático de generación de números enteros o caracteres alfanuméricos, estos deben ser producidos en el espacio de kernel ya que deben estar siempre disponibles al usuario ya sea para *MySQL* o *PostgreSQL*		
Nota			

Código	RF2	Prioridad	Alta
Nombre	Colección de secuencias		
	Se debe tener una colección de secuencias de al menos 1024 secuencias de manera que estén disponibles para diferentes procesos referentes a un modelo de bases de datos.		
Nota			

Código	RF3	Prioridad	Alta
Nombre	Comunicación con el kernel		
	Las secuencias deberán poder ser accedidas desde los DBMS *MySQL* y *PostgreSQL*, mediante funciones definidas por el usuario a través de una librería compartida programada en el leguaje C para la comunicación directa con el espacio de kernel.		
Nota			

Código	RF4	Prioridad	Alta
Nombre	Creación de secuencia		
	El sistema debe poder inicializar una secuencia con un valor inicial dado por el usuario. El valor se incrementara en uno.		
Nota	Por defecto el valor inicial debe ser 1		

Código	RF5	Prioridad	Media
Nombre	Actualización de secuencia		
	Se debe tener una función que actualice los valores de cada secuencia.		
Nota			

Código	RF6	Prioridad	Media
Nombre	Eliminación de secuencias		
	El sistema debe tener una función que elimine secuencias en dado caso ya no se necesiten.		
Nota			

Código	RF7	Prioridad	Alta
Nombre	Interacción del usuario		
	El usuario no deberá interactuar con sentencias DDL para la definición de estas funciones, en cambio se debe proporcionar un API mediante un programa en espacio de usuario que permita la creación de claves sustitutas, el proceso de creación de librerías e integración debe ser completamente invisible al usuario.		
Nota			

Código	RF8	Prioridad	Alta
Nombre	Persistencia de datos		
	El almacenamiento de las variables debe escribirse en archivos planos.		
Nota			

Código	RF9	Prioridad	Alta
Nombre	Interfaz de usuario		
	El sistema debe contar con una interfaz capaz de interactuar con el usuario mediante comandos, los cuales permitirán administrar el sistema.		
Nota			

2.5. Casos de uso

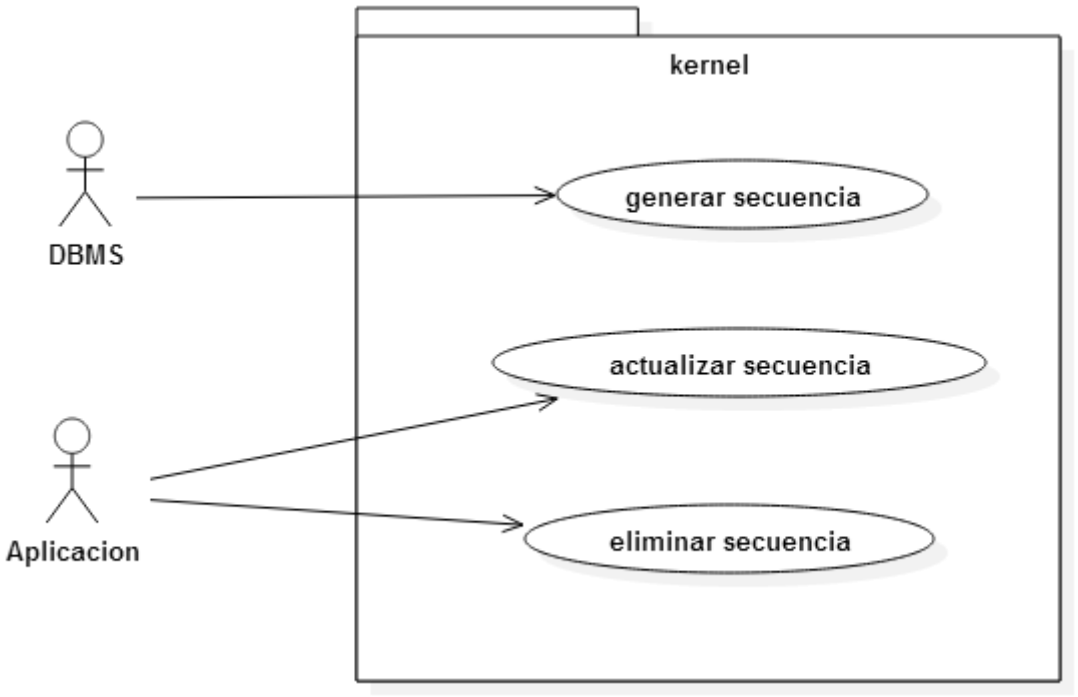
A continuación se detallan los casos de uso de la aplicación.

Caso de uso 01	Generar secuencia
Actores	Kernel, DBMS
Propósito	Generar número entero o alfanumérico que se utilizará como clave sustituta.
Resumen	El DBMS pide al kernel un clave generada
Condiciones	Entorno instalado (módulos, aplicación, DBMS etc.).
Post-condiciones	Se debe tener este valor siempre en memoria principal.
Curso normal	1. El usuario ejecuta una sentencia DML con una llamada la función de obtención de clave instalada. 2. Se ejecuta código de kernel para la solicitud del usuario 3. La aplicación guarda la nueva secuencia
Curso alterno	1. El total de secuencias ha llegado a su límite
Requerimiento funcional	RF1

Caso de uso 03	Eliminar secuencia (válido solo para claves sustitutas numéricas).
Actores	Usuario, kernel
Propósito	Eliminar una secuencia de la aplicación
Resumen	El usuario borra y deja espacio libre para una secuencia nueva
Condiciones	El objeto secuencia debe estar previamente definido.
Post-condiciones	Ninguna
Curso normal	1. Mediante la aplicación, el usuario puede eliminar un objeto secuencia. 2. Se libera el espacio ocupado
Curso alterno	1. La secuencia dada no ha sido definida
Requerimiento funcional	RF6

Caso de uso 04	Actualizar secuencia (válido solo para claves sustitutas numéricas).
Actores	Kernel, Usuario
Propósito	Actualizar una secuencia de la aplicación
Resumen	El usuario actualiza el número activo de la secuencia.
Condiciones	El objeto secuencia debe estar previamente definido.
Post-condiciones	Ninguna
Curso normal	1. Mediante una interfaz, el usuario puede actualizar un objeto secuencia. 2. El valor actual de la secuencia es modificado
Requerimiento funcional	RF5

Figura 3. **Diagrama de casos de uso**



Fuente: elaboración propia, empleando StarUML.

2.6. Interfaz de administración para el usuario

Se trata de uno de los requerimientos funcionales más importantes, ya que una interfaz para el usuario permitirá que se desarrollen los casos de uso por medio de un canal que se comunica con el kernel y la base de datos desde espacio de usuario, esto facilita al usuario la administración del sistema. La interfaz consiste en un programa escrito en C y se accede mediante la terminal de comandos.

2.6.1. Requerimientos funcionales

A continuación se describen los requerimientos funcionales de la interfaz de usuario.

Código	RFA1	Prioridad	Alta
Nombre	Ejecutable en la terminal		
	El usuario debe acceder a la interfaz por medio de un comando ejecutable desde cualquier entorno.		
Nota	El nombre de la aplicación debe ser *seqgen*, por lo cual al ejecutar dicho comando en la terminal accederá a la aplicación		

Código	RFA2	Prioridad	Alta
Nombre	Menú de selección de base de datos		
	Se trata de un menú que se muestra en consola que permite seleccionar al usuario con que DBMS trabajará.		
Nota			

Código	RFA3	Prioridad	Media
Nombre	Persistencia de credenciales		
	El programa debe permitir al usuario guardar las credenciales para usarlas en futuras conexiones si así lo desea.		
Nota			

Código	RFA4	Prioridad	Alta
Nombre	Menú principal		
	Se debe mostrar un menú con una opción para cada caso de uso del sistema. Cada opción necesita como parámetro el número de secuencia, el resultado se desplegara en consola.		
Nota			

Código	RFA5	Prioridad	Alta
Nombre	Instalación de módulos y configuración		
	Cada vez que el sistema operativo inicia, la aplicación instalada debe iniciar e insertar los módulos en el kernel.		
Nota			

Código	RFA6	Prioridad	Alta
Nombre	*Back-up* y restauración		
	El sistema debe ser capaz de guardar el estado de las secuencias para cada base de datos, dichos datos deben ser persistidos en archivos binarios.		
Nota			

Código	RFA7	Prioridad	Baja
Nombre	Línea de comandos		
	Se debe de contar con una línea de comandos que ejecute cada caso de uso con el objetivo de brindar al usuario un acceso directo a la funcionalidad del sistema.		
Nota			

2.6.2. Casos de uso

A continuación se detallan los casos de uso de la aplicación de interfaz de usuario.

Caso de uso 01	Ejecutar aplicación
Actores	Usuario
Propósito	Inicial la aplicación de interfaz de usuario.
Resumen	El usuario inicia el programa por medio de un comando en la terminal.
Condiciones	Ninguna
Post-condiciones	Ninguna
Curso normal	1. El usuario ingresa el comando seqgen en la terminal. 2. La aplicación se inicia
Curso alterno	1. El usuario ingresa el comando con errores
Requerimiento funcional	RFA1

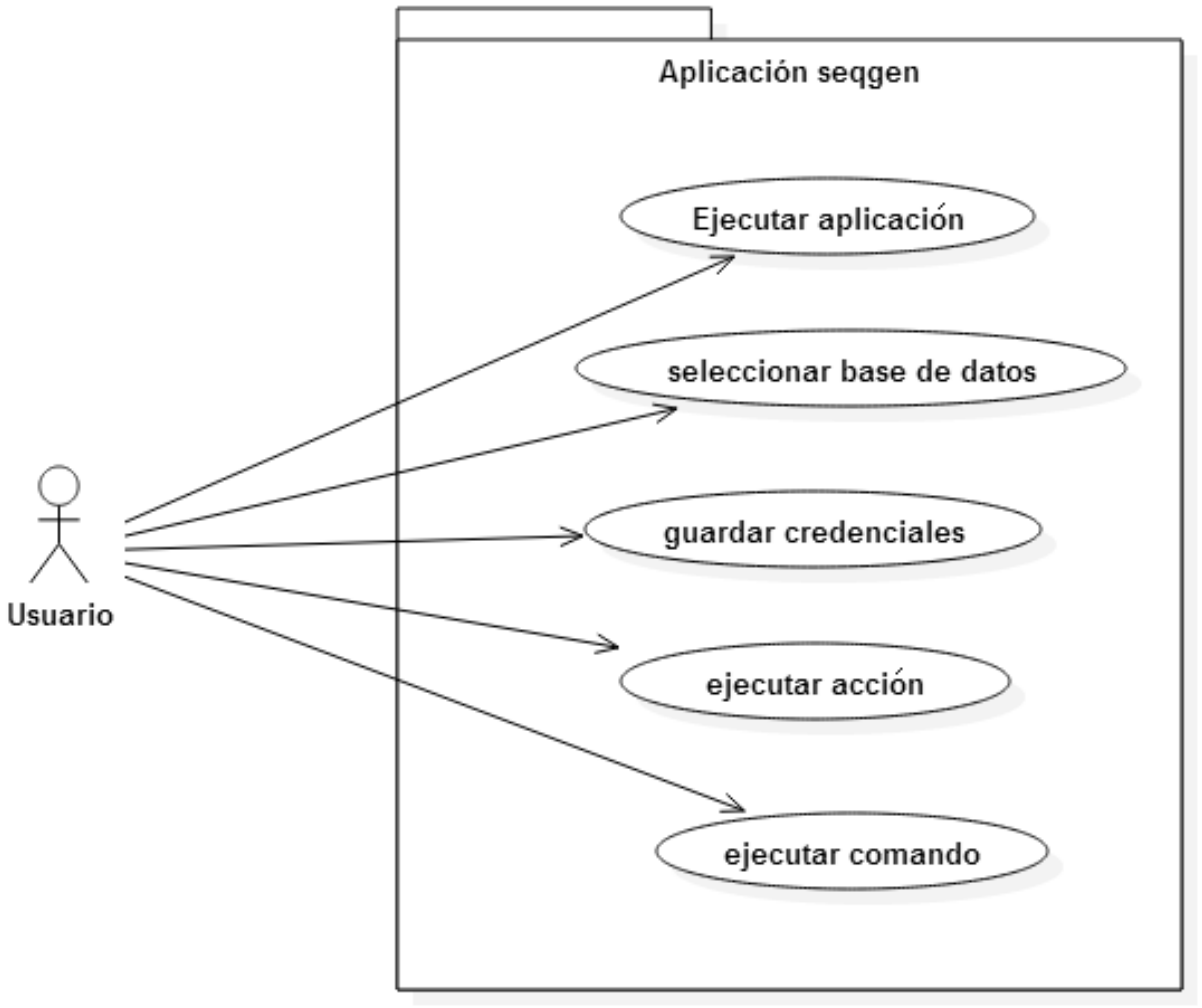
Caso de uso 02	Selección de base de datos
Actores	Usuario
Propósito	Se selecciona MySQL o PostgreSQL.
Resumen	El programa obtiene del usuario la elección del sistema de base de datos.
Condiciones	Ninguna
Post-condiciones	Ninguna
Curso normal	1. El usuario selecciona una de las bases de datos instalada. 2. Se presenta el menú principal
Curso alterno	1. La base de datos seleccionada ha sido desinstalada.
Requerimiento funcional	RFA2

Caso de uso 03	Guardar credenciales
Actores	Usuario
Propósito	Persistir credenciales en el sistema.
Resumen	El programa persiste datos en un archivo binario.
Condiciones	Ninguna
Post-condiciones	Ninguna
Curso normal	1. El usuario selecciona una de las bases de datos instalada. 2. Selección la opción guardar credenciales
Curso alterno	1. La base de datos seleccionada ha sido desinstalada.
Requerimiento funcional	RFA3

Caso de uso 04	Seleccionar acción
Actores	Usuario
Propósito	Seleccionar acción a ejecutar en el kernel
Resumen	El programa obtiene del usuario la elección de la acción en el kernel (crear, actualizar, eliminar o generar *UUID*).
Condiciones	Entorno instalado
Post-condiciones	Ninguna
Curso normal	1. El usuario selecciona la acción a realizar 2. El kernel mediante un módulo ejecuta la acción
Curso alterno	1. Módulos no cargados
Requerimiento funcional	RFA4

Caso de uso 05	Ejecución de comando
Actores	Usuario
Propósito	Ejecutar acción mediante un acceso más rápido
Resumen	El usuario ejecuta un comando
Condiciones	Ninguna
Post-condiciones	Ninguna
Curso normal	1. El usuario ejecuta un comando desde la terminal
Curso alterno	1. El comando no se encuentra
Requerimiento funcional	RFA7

Figura 4. **Diagrama de casos de uso**



Fuente: elaboración propia, empleando StarUML.

2.7. Requerimientos no funcionales del sistema

A continuación se brinda una vista general de los requisitos no funcionales del sistema en general.

2.7.1. Integración con sistemas

El presente sistema generador de claves candidatas, debe ser totalmente adaptable a cualquier arquitectura, siempre y cuando contenga un sistema operativo Linux como base.

2.7.2. Modularidad

El sistema se debe subdividir en módulos o componentes, cada uno con una función específica y debido a su independencia deben ser compilados en distintos tiempos. El objetivo de dividir el sistema en parte es reducir la complejidad de la ejecución de procesos que lo componen para cumplir los objetivos de la aplicación en general.

2.7.3. Portabilidad

El sistema debe ser multiplataforma, cualquier computador que posea un sistema operativo con el kernel de Linux, instalara correctamente el software. Esta portabilidad es gracias a la modularidad del kernel, que permite expandir su funcionalidad con los módulos de kernel.

2.7.4. Compatibilidad

Una de las características más importantes es la compatibilidad del sistema, se puede decir que el sistema es multiplataforma, por lo tanto se adapta a cualquier infraestructura LAMP.

2.7.4.1. Sistema operativo Linux

El sistema es compatible con cualquier distribución Linux con la versión de kernel mayor o igual a la 2.6. Los *char drivers* permiten que nuevo código se agregue al sistema operativo sin necesidad de compilar el kernel. Los módulos administrador y controlador no presentan problemas con la integración de cualquier sistema opeartivo basado en el kernel de Linux.

2.7.4.2. MySQL Server y PostgreSQL

Tanto *MySQL* como *PostgreSQL*, al igual que el kernel de Linux, permiten extender su funcionalidad gracias a la interfaz *UDF*, la cual permite agregar código compilado como librerías de objetos compartidos y añadir o remover la función mediante lenguaje DDL. El cliente de secuencias de ambos, es una librería compilada, que se instala en el sistema de base de datos y puede ser removida en cualquier momento. Esto permite que cualquier versión de estos sistemas gestores de bases de datos, puedan integrar el sistema sin ningún problema.

2.7.5. Lista de requerimientos no funcionales

A continuación se presenta el listado completo de los requerimientos no funcionales.

Código	RNF1	Prioridad	Alta
Nombre	Compatibilidad		
	El sistema debe ser compatible con cualquier distribución Linux con una versión de kernel 2.6 o mayor, *MySQL* con versión mayor a 5.6 o *PostgreSQL* con versión mayor a la 9.1.		
Nota			

Código	RNF2	Prioridad	Alta
Nombre	Portabilidad		
	El sistema debe ser portable, es decir, tanto los módulos como la aplicación deben ejecutarse en cualquier entorno sin importar las condiciones, el comportamiento debe ser el mismo.		
Nota			

Código	RNF3	Prioridad	Media
Nombre	Total de secuencias		
	El sistema debe soportar una como mínimo 1024 secuencias por cada base de datos.		
Nota			

Código	RNF4	Prioridad	Media
Nombre	Transparencia		
	El manejo interno del conjunto de secuencias debe ser transparente al usuario.		
Nota			

Código	RNF6	Prioridad	Alta
Nombre	Documentación		
	Se debe proporcionar manuales para que el usuario pueda manejar de forma adecuada la aplicación.		
Nota			

2.8. Análisis de la solución

Dados los requerimientos de la solución, a continuación se realiza un breve análisis de la solución.

2.8.1. Aspectos generales

El sistema requerido opera sobre espacio de usuario y espacio de kernel, a diferencia de la mayoría de aplicaciones que normalmente operan solo en el espacio de usuario y eventualmente requieren de una llamada al kernel. Se debe establecer una vía de comunicación activa y constante entre ambas capas. A continuación se listan los componentes de cada capa:

Espacio de kernel:

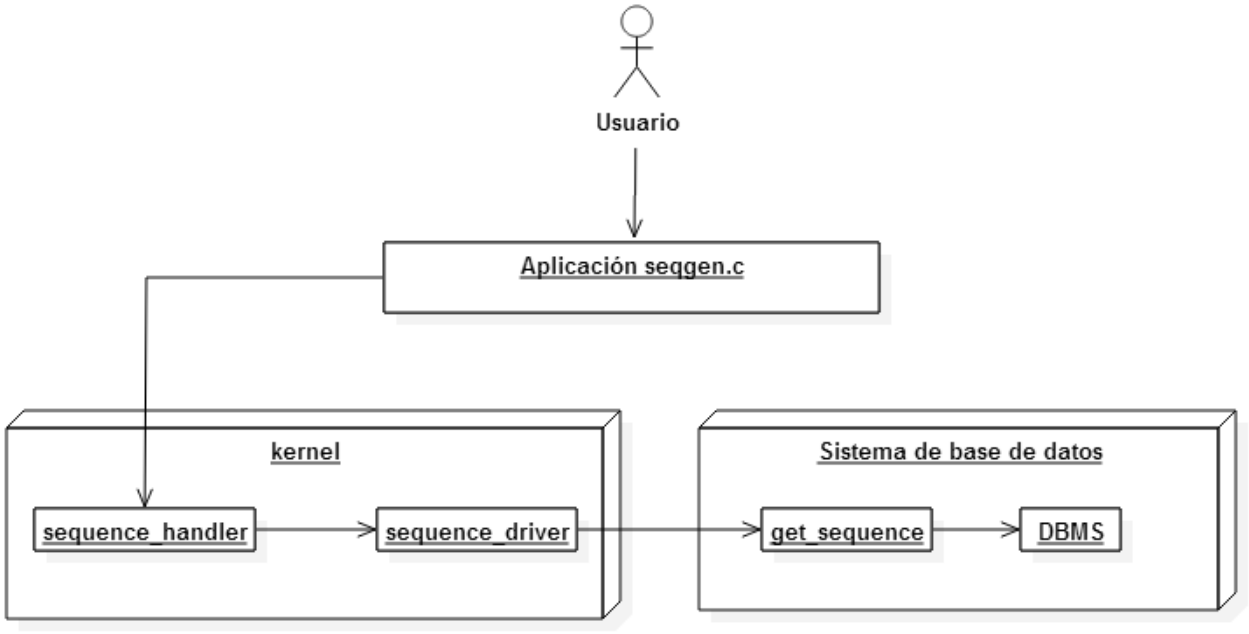
- Módulo de kernel: Es el componente que extiende la funcionalidad del kernel, se encarga de enviar datos numéricos o alfanuméricos al DBMS por medio de un canal de comunicación directa.
- Sistema de archivos: Se utiliza para almacenar la información necesaria del módulo, ya que esta estará disponible únicamente cuando el sistema operativo este activo, si el ordenador se apaga el modulo no guardara su estado y perderá datos.

Espacio de usuario:

- DBMS: Componente que administra la información que es directamente alterada por las operaciones que realice el modulo, recibe información del módulo de kernel.

- Aplicación de usuario: Es un programa que brinda un API al usuario con el fin de que este no interactúe directamente con el kernel, su función principal es actualizar el estado del módulo de kernel.

Figura 5. **Diagrama general de la solución**



Fuente: elaboración propia, empleando StarUML.

3. ARQUITECTURA DE LA SOLUCIÓN

En el presente capítulo describe la arquitectura de software de la aplicación, diagrama componentes y secuencias, dado que la mayor parte del sistema está escrito en lenguaje C y con el paradigma de programación procedural, no se presenta un diagrama de clases como tal.

3.1. Modelo del sistema

El sistema general se divide en 4 componentes organizados en 3 distintas capas. El programa de interfaz de usuario posee funciones para ambas bases de datos si se encuentran presentes en el sistema local.

3.1.1. Definición de componentes

A continuación se describe la visión funcional de la arquitectura del sistema.

3.1.1.1. A nivel de kernel

La capa de kernel se conforma de módulos de kernel, dichos módulos son encargados de gestionar y manipular las variables que se envían a los servidores de bases de datos. Ambos módulos son dispositivos controladores instalables o *drivers* de tipo *char driver* e intercambian información con el usuario gracias a los macros *get_user, __put_user* y *copy_to_user,* definidos en el la cabecera *uaccess.h* del kernel.

3.1.1.1.1. Controlador de secuencias

Consiste en un módulo de kernel encargado de generar los números de una determinada secuencia, el modulo es cargado cada vez que el sistema se inicia. Este componente consta de un arreglo unidimensional de tipo entero con un almacenamiento de clase de tipo registro para acceso más rápido, esta variable es expuesta a todo el kernel para que el modulo administrador, que también se encuentra en espacio de kernel, pueda tener acceso. La actualización del valor de una determinada secuencia se hace por medio de código assembly para mayor rendimiento y dicha acción en este módulo puede ser desencadenada únicamente por el DBMS. Para colocar el valor de una secuencia en espacio de usuario se utiliza un macro llamado *__put_user*.

3.1.1.1.2. Administrador de secuencias

El administrador de secuencias es un módulo en espacio de kernel, cuyo objetivo consiste en actualizar, restaurar o simplemente presentar datos al usuario, al igual que el controlador de secuencias. Cabe destacar que este módulo a diferencia del controlador se secuencias, obtiene instrucciones directas del usuario, las cuales se reciben gracias al macro *get_user*. El administrador de secuencias contiene la referencia al arreglo de secuencias expuesto por el controlador.

3.1.1.2. A nivel de base de datos

En la capa de base de datos, únicamente se encuentra un componente, aquí se recibe la información del kernel y se convierte en datos útiles. Es la capa que resulta afectada directamente ya sea por los módulos de kernel o por el usuario.

3.1.1.2.1. Cliente de secuencias

Este componente consiste en una rutina ejecutable únicamente por el DBMS, escrita como UDF, en lenguaje C y al igual que los módulos de kernel, pretende extender la funcionalidad del DBMS para que sea capaz de usar la información brindada por los módulos de kernel desde sentencias DML. Esta rutina UDF es compilada como una librería de objetos compartidos. Consta de un descriptor de archivos que apunta al dispositivo controlador y el método *get_sequence* el cual obtiene el siguiente valor de una secuencia, bajo este mismo nombre puede utilizarse en un script de lenguaje SQL. En este componente también se encuentra una implementación para generar códigos universales únicos, UUID, por siglas en inglés, *Universal Unique Identifiers*, los cuales pueden usarse también como claves candidatas en el DBMS, estos identificadores se pueden obtener por medio de la función *get_uuid*, que puede usarse en el DBMS en lenguaje SQL.

3.1.1.3. A nivel de usuario

Esta capa, como su nombre lo indica, es la única que interactúa directamente con el usuario, únicamente consta de un proceso que recibe instrucciones de usuario. El programa escrito en C tiene como nombre *seqgen* y es instalado dentro del directorio */usr/bin* del sistema de archivos.

3.1.1.3.1. Programa de interfaz de usuario

Este componente consta de un software con instrucciones para ejecutar tareas en los dispositivos controlador y administrador de secuencias, recibe entradas del usuario.

3.1.2. Comunicación entre componentes

La comunicación entre todos los componentes que conforman la aplicación se hace mediante la rutina del sistema de Unix, *ioctl*.

3.1.2.1. Comunicación entre capa de kernel y capa de base de datos

Se produce cuando el DBMS ejecuta una sentencia DML de tipo *select* o *insert* y contiene a *get_sequence* como parámetro. El cliente de secuencias de *MySQL* o *PostgreSQL* instancia un descriptor de archivos apuntando al módulo controlador de secuencias de la determinada base de datos, el descriptor es abierto solo la primera vez que se ejecuta la función *get_sequence* y es cerrado únicamente cuando el servidor es apagado, esto con fines de reducir el tiempo de ejecución. Utilizando *ioctl*, se crea una operación de tipo generación de valor, entonces *ioctl* recibe como segundo parámetro un número el cual representa el número de secuencia y el modulo controlador se encarga de incrementar el valor en uno.

3.1.2.2. Comunicación entre capa de kernel y capa de usuario

Se produce cuando el usuario mediante el programa de interfaz, actualiza, crea o elimina una secuencia. Al igual que la comunicación con la comunicación entre capa de kernel y de base de datos, se crea un descriptor en el programa de interfaz de usuario, con la diferencia que este apunta al módulo administrador de secuencias. El método *ioctl* recibe como segundo parámetro el macro auxiliar llamado *_IOR* y definido en este sistema como *IOCTL_SET_SEQ,* el cual significa que el proceso que lo utiliza enviara

información a través de *ioctl*. Esta información, se envía al módulo administrador en un *struct* con el nombre *sequence_request,* este tipo de dato abstracto contiene el nuevo valor de la secuencia y el número de secuencia a actualizar, eliminar o restaurar. Cabe mencionar que cuando el usuario requiera consultar el valor actual de una secuencia, el proceso es el mismo que el de la comunicación entre el kernel y la base de datos, con la diferencia de que el modulo administrador no incrementa en uno el valor de la secuencia.

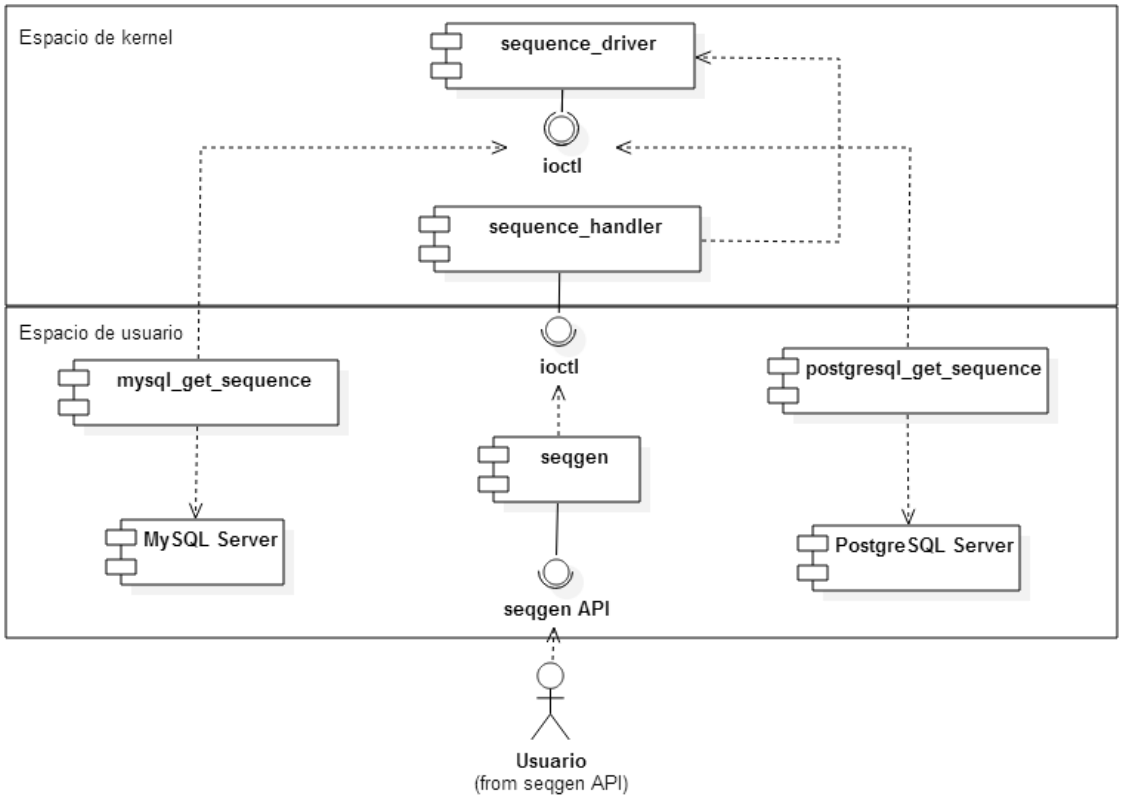
3.1.2.3. Comunicación entre capa de usuario y capa de base de datos

En este proceso no se utiliza *ioctl* ya que se produce únicamente cuando el usuario interactúa directamente con el DBMS, ejecutando sentencias en lenguaje SQL que contienen las funciones *get_sequence* o *get_uuid.*

3.1.3. Diagrama de componentes

A continuación se muestra el diagrama de componentes que representa la arquitectura del sistema.

Figura 6. **Diagrama de componentes del sistema**

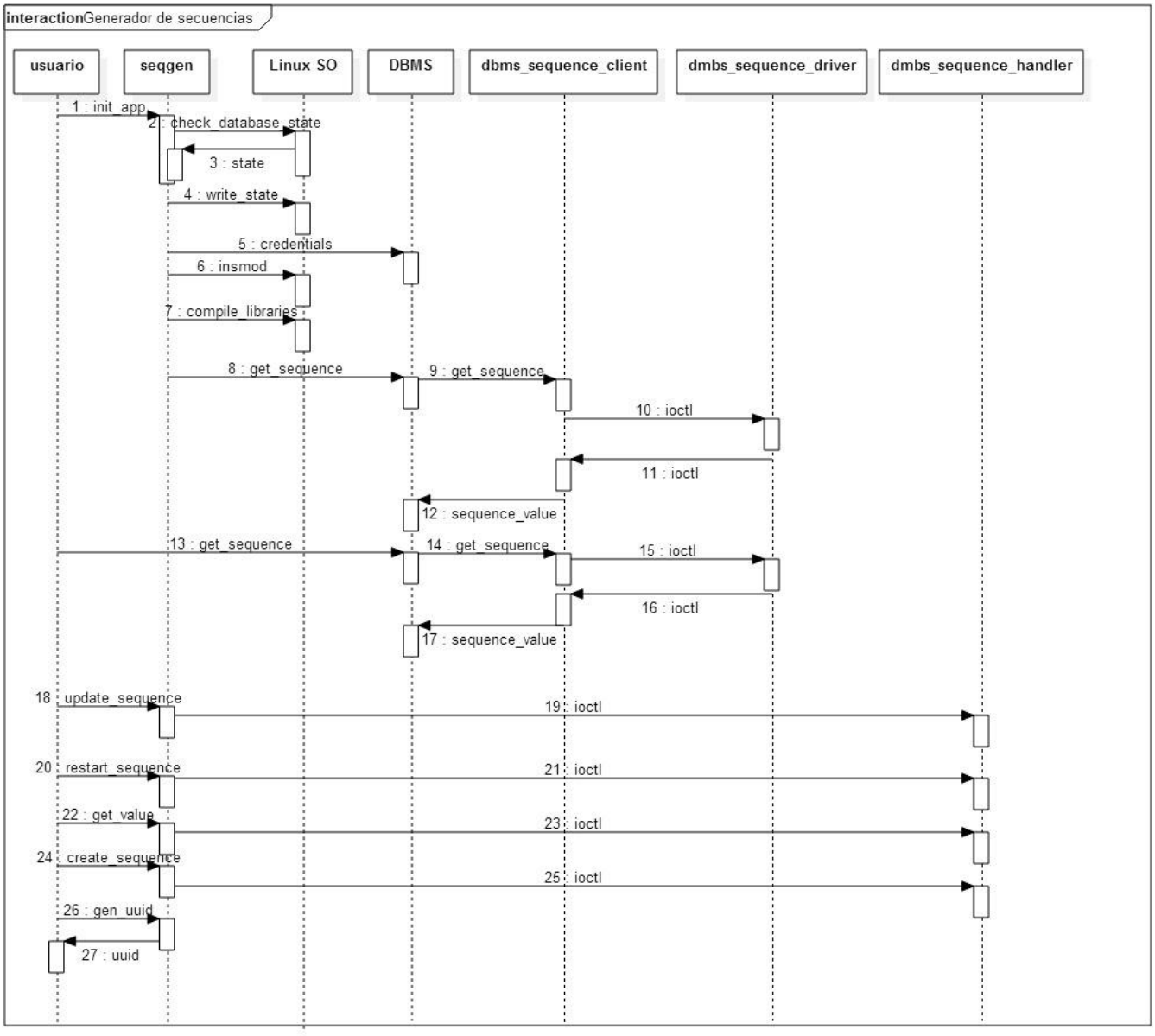


Fuente: elaboración propia, empleando StarUML.

3.1.4. Diagrama de secuencias

A continuación se presenta el diagrama de secuencias del sistema

Figura 7. **Diagrama de secuencias del sistema**



Fuente: elaboración propia, empleando StarUML.

4. IMPLEMENTACIÓN DE SECUENCIAS INCREMENTALES

En este capítulo, se presenta un análisis de rendimiento de la aplicación así como un manual de usuario el cual describe los pasos para la instalación y uso. El objetivo de este capítulo es presentar la solución y resultados finales de la aplicación.

4.1. Benchmark de la aplicación

En las siguientes secciones se presenta una serie de pruebas hechas a *seqgen* instalado en *MySQL* y *PostgreSQL*. Se comparan los tiempos y se brinda una gráfica de los resultados.

4.1.1. Pruebas en MySQL

En las siguientes tablas y graficas se muestran resultados de las pruebas hechas en *MySQL*.

Tabla II. **Resultados de *seqgen* en *MySQL***

No. De registros	Tiempo de inserción (s)	Tiempo de borrado (s)
100	4,53	0,07
400	19,852	0,055
800	48,119	0,066
1 200	70,132	0,085
1 600	100,66	0,077
2 000	134,486	0,99
2 400	151,68	0,79
2 800	170,155	0,099
3 200	196,576	0,121
4 000	258,496	0,99

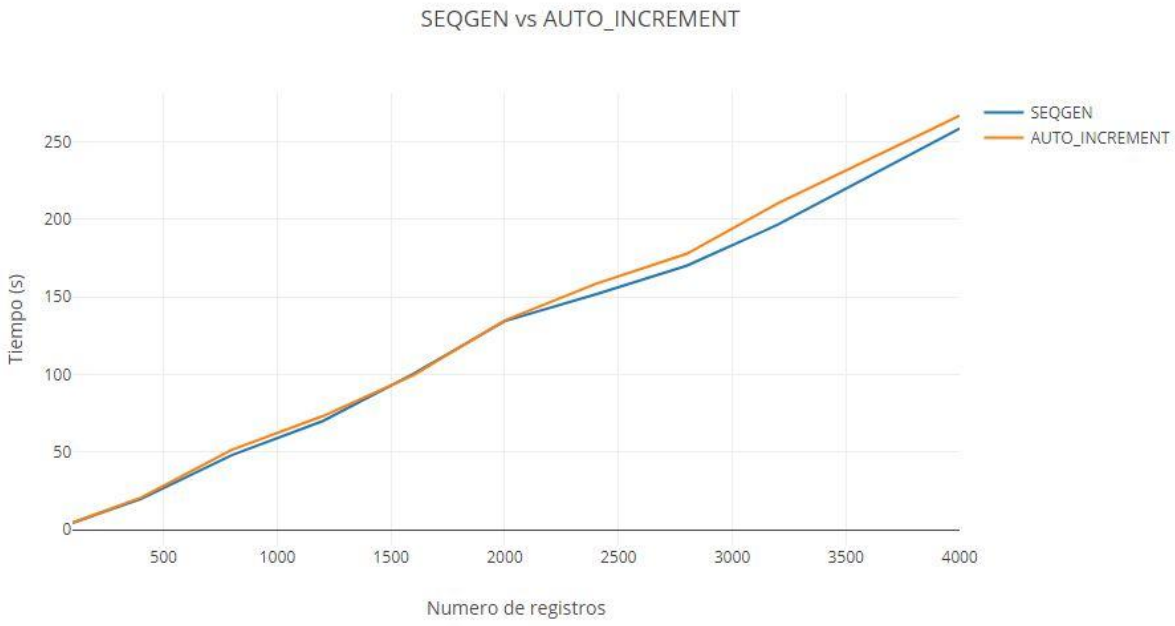
Fuente: elaboración propia.

Tabla III. **Resultados de *AUTO_INCREMENT* de *MySQL***

No. De registros	Tiempo de inserción (s)	Tiempo de borrado (s)
100	4,64	0,08
400	20,722	0,099
800	51,549	0,111
1 200	73,281	0,065
1 600	99,731	0,111
2 000	134,908	0,245
2 400	158,481	2,44
2 800	177,8151	0,1
3 200	210,266	0,111
4 000	266,8	0,132

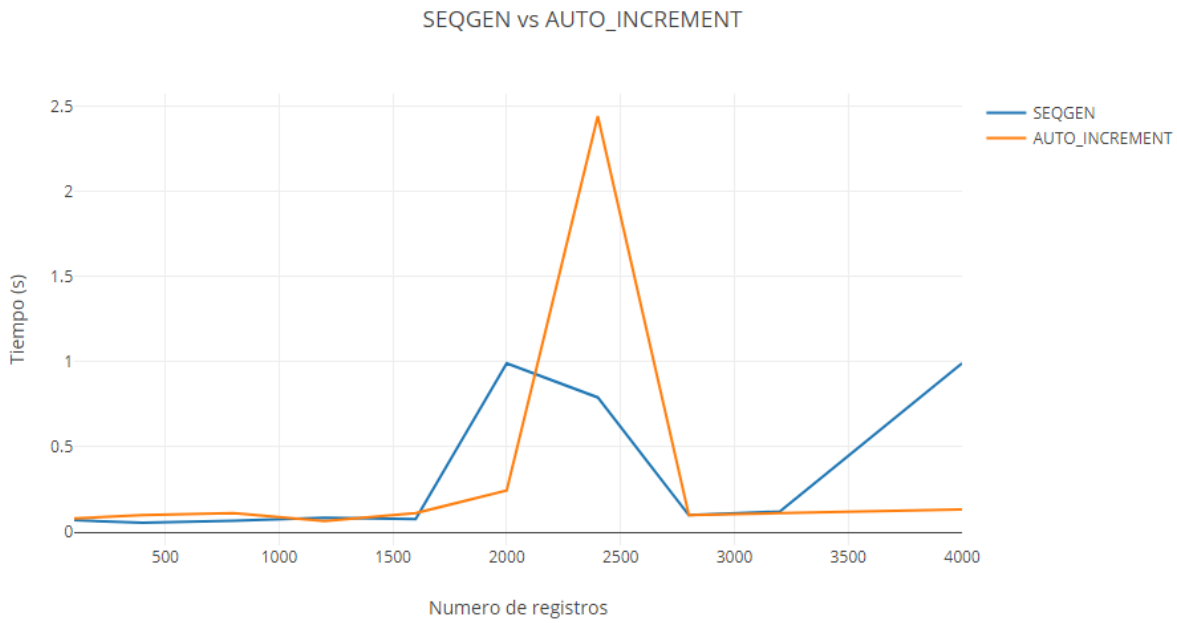
Fuente: elaboración propia.

Figura 8. **Gráfica de inserciones *seqgen* vs *AUTO_INCREMENT***



Fuente: elaboración propia, empleando *Graph Maker Online*.

Figura 9. **Gráfica borrado *seqgen* vs *AUTO_INCREMENT***



Fuente: elaboración propia, empleando Graph Maker Online.

4.1.2. Pruebas en *PostgreSQL*

Se muestran los resultados de las pruebas hechas en *PostgreSQL*

Tabla IV. **Resultados de *seqgen* en *PostgreSQL***

No. De registros	Tiempo de inserción (s)	Tiempo de borrado (s)
100	0,027	0,029
400	0,1	0,067
800	0,083	0,024
1 200	0,13	0,047
1 600	0,137	0,053
2 000	0,142	0,037
2 400	0,191	0,054
2 800	0,19	0,043
3 200	0,327	0,65
4 000	0,277	0,053

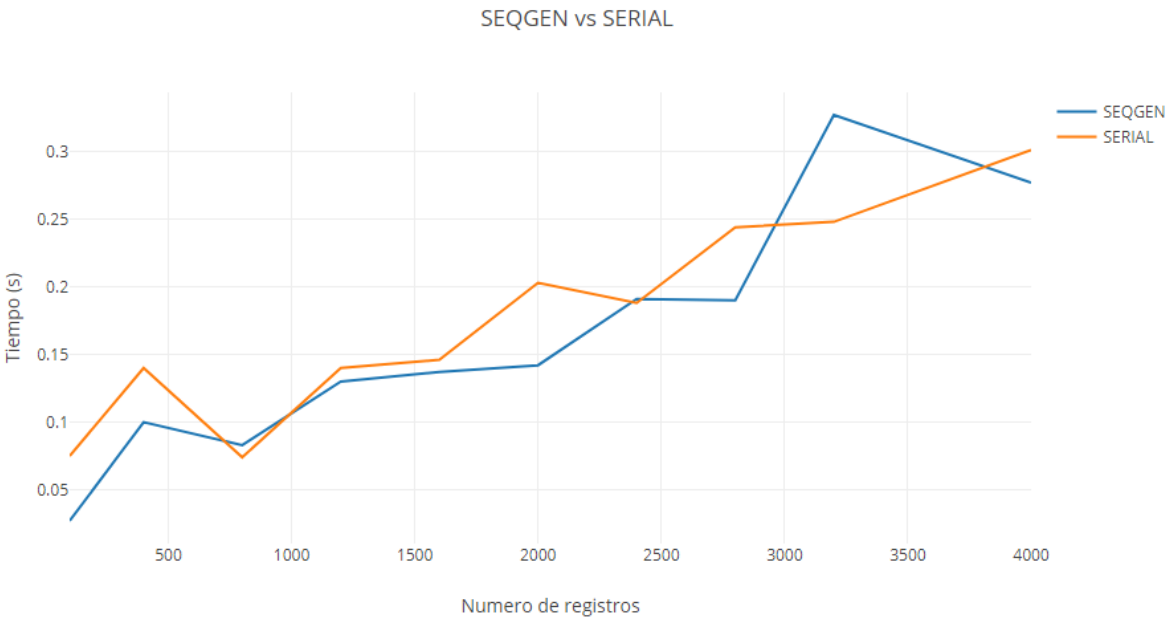
Fuente: elaboración propia.

Tabla V. **Resultados de pruebas de *SERIAL* de *PostgreSQL***

No. De registros	Tiempo de inserción (s)	Tiempo de borrado (s)
100	0,075	0,095
400	0,14	0,086
800	0,074	0,079
1 200	0,14	0,054
1 600	0,146	0,092
2 000	0,203	0,049
2 400	0,188	0,078
2 800	0,244	0,053
3 200	0,248	0,76
4 000	0,301	0,056

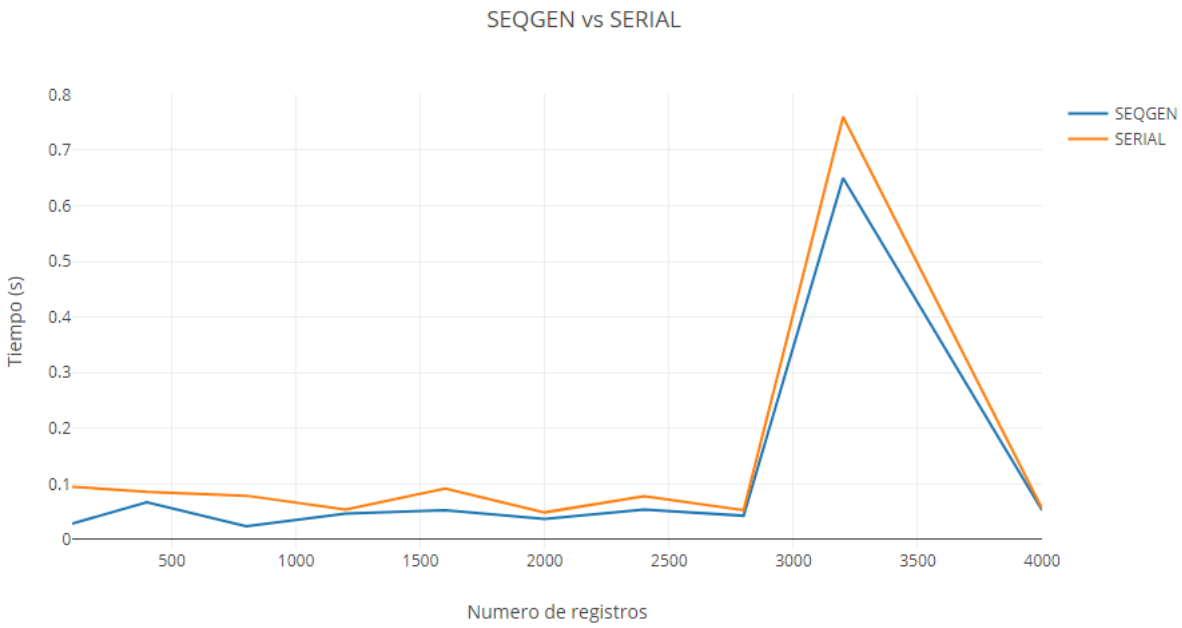
Fuente: elaboración propia.

Figura 10. **Inserciones de *seqgen* vs SERIAL de *PostgreSQL***



Fuente: elaboración propia, empleando *Graph Maker Online*.

Figura 11. **Grafica de borrado de *seqgen* vs *SERIAL***



Fuente: elaboración propia, empleando Graph Maker Online.

4.2. Instalación y uso

Esta sección está dedicada al manual de instalación, requerimientos necesarios para la instalación y uso del usuario.

Seqgen está disponible para su descarga en el enlace: https://github.com/pedrocruzlopez/sequence_generator

4.2.1. Requerimientos de instalación

Se presentan el software necesario para que *seqgen* pueda instalarse.

- Sistema operativo Linux: es necesario tener este sistema operativo como base, no importando la distribución, seqgen es compatible con versiones de kernel mayor o igual a la

2.6, como se describió en el requerimiento no funcional (RNF1) en el capítulo 2.

- Cabeceras de kernel: necesarias para la compilación de módulos. Verificar apéndices para su instalación.
- *GNU Compiler Collection*: colección de compiladores, necesarios para compilar módulos de kernel y la interfaz de usuario. Verificar apéndices para la instalación en las distintas distribuciones.
- *CMake*: herramienta multiplataforma para construir y empaquetar software, necesaria para empaquetar *seqgen*, la versión mínima a usarse debe ser la 2.6. Verificar apéndices para la instalación en las distintas distribuciones.
- *MySQL Server* o *PostgreSQL*: sistemas de base de datos relacionales, pueden instalarse ambos o uno solo, la versión mínima a instalarse debe ser 5.6 y 9.1 respectivamente.
- *Git*: control de versiones, no es necesaria su instalación, únicamente si se desea clonar el proyecto, en otro caso puede descargarse el archivo comprimido.

4.2.2. Instalación

El sistema seqgen se instala en tres pasos básicos, los cuales de muestran a continuación:

1. Descargar *seqgen* manualmente desde el enlace o clonar desde *Git* abriendo la terminal de comandos y escribir:

```
git clone
https://github.com/pedrocruzlopez/sequence_generator.git
```

2. Brindar permisos a la carpeta donde se descargo

```
chmod -R 775 /ruta/sequence_generator
```

3. Ejecutar el archive script de instalación desde la terminal

```
./install.sh
```

4.2.3. Manual de uso

El propósito de esta sección es brindar al usuario un manual en el cual se detalla su uso y manejo.

4.2.3.1. Idioma de instalación

Tanto el código fuente como la interfaz de usuario están en idioma inglés para su mayor escalabilidad y entendimiento.

4.2.3.2. Manejo de claves subrogadas

Seqgen dispone de una interfaz de usuario que permite administrar las secuencias sin necesidad de verificar tablas en el DMBS. Abajo se detallan los pasos para manejar secuencias y comenzar a utilizarlas.

1. Después de instalar seqgen, la primera acción antes de utilizar secuencias en el DMBS, se debe ejecutar seqgen desde una instancia de terminal con el comando con el mismo nombre. Es importante tomar en cuenta que el comando *seqgen* debe ejecutarse como usuario *root.* Cuando se ejecuta *seqgen,* se despliega un menú para seleccionar el sistema de base de datos con el que se va a trabajar:

```
$ seqgen

Please choose your database:
1) MySQL Server
2) PostgreSQL
3) Exit
```

2. Cuando se haya elegido con que DMBS se trabajará, se solicita al usuario credenciales de conexión, el usuario puede elegir que seqgen guarde las credenciales para no volver a solicitarlas. Tomar en cuenta que si es primera vez que se elige el DMBS, entonces *seqgen* procederá a compilar e instalar módulos de kernel y archivos necesarios para el funcionamiento correcto. Cuando esté listo se desplegara el menú con las siguientes opciones:

```
Select option:
1) Create sequence
2) Update sequence
3) Restart sequence
4) Get current number
5) Generate UUID
6) Exit
```

   - Crear secuencia: cada secuencia comienza con 0 por defecto, cuando crea una nueva secuencia se puede establecer un valor inicial y la secuencia incrementará en uno el siguiente número comenzando con el valor inicial, tomar en cuenta que tal y como se describió en requerimiento funcional numero dos (RF2), en el capítulo dos, se tienen hasta 1024 secuencias disponibles para su uso.
   - Actualizar secuencia: se solicita el índice o número de secuencia y se solicita el nuevo valor actual, muy importante tomar en cuenta que esta operación puede generar violaciones de integridad referencial si la secuencia ya está en uso y como parte de una clave primaria.

- Reiniciar secuencia: reinicia determinada secuencia en un valor, al igual que la operación de actualización puede incurrir en riesgos de violación de integridad referencial.
- Obtener número actual: dado un índice, se mostrara en pantalla el valor actual de la determinada secuencia.
- Generar identificador universal único: esta opción es de propósito general y despliega un *UUID* en pantalla para su uso.
- Salir: Terminal la ejecución de *seqgen*

4.2.3.3. Comandos de abreviación

Seqgen también posee su propia línea de comandos la cual proporciona un uso más rápido y abreviado. Los comandos y sus parámetros se muestran en la siguiente tabla:

Tabla VI. **Línea de comandos de *seqgen***

Comando	Abreviación	Parámetros	Descripción
--help	-h	Ninguno	Imprime la ayuda de comandos.
--cre_seq	-c	• DBMS: mysql o psql • Index: número • Valor inicial	Crea una nueva secuencia con un valor inicial.
--set_seq	-s	• DBMS: mysql o psql • Index: número • Valor	Actualiza una secuencia
--res_seq	-r	• DBMS: mysql o psql • Index: número • Valor	Reinicia una secuencia
--get_seq	--g	• DBMS: mysql o psql • Index: número	Obtiene el valor actual de una secuencia.
--uuid	-u	Ninguno	Genera un, UUID para propósitos generales.

Fuente: elaboración propia.

Ejemplo de uso:

```
$seqgen -c mysql 1 10
```

4.2.3.4. Utilización en el DBMS

Cuando se haya instalado y configurado *seqgen*, *MySQL* o *PostgreSQL* tendrán las funciones: *get_sequence* y *get_uuid*.

4.2.3.4.1. Función *get_sequence*

Esta función recibe un parámetro entero sin signo (*unsigned int*) que representa el índice de una secuencia y devuelve un valor *unsigned int* que se incrementará cada vez que se llame a la función para el secuencia específica, *get_sequence* puede llamar desde cualquier instrucción SQL.
Ejemplos:

INSERT:

```
INSERT INTO TABLE (id, name, country) VALUES (get_sequence(1), "Linus Torvalds”, "Finland");
```

SELECT:

```
SELECT get_sequence(2);
```

Considere que la llamada a la función desde una instrucción *SELECT* también incrementará el valor actual de la secuencia.

4.2.3.4.2. Función *get_uuid*

Esta función generará un identificador universal único en el caso de que necesite un campo no numérico generado automáticamente, no hay necesidad de parámetros porque un *UUID* no es parte de una colección de campos como la secuencia numérica.

Example:

```
INSERT INTO TABLE (id, name, country) VAUES (get_uuid(), "Linus
Torvalds" , "Finland");
```

4.3. Utilización para otros propósitos

Seqgen no solo fue diseñado para propósitos de uso en bases de datos, cualquier aplicación que necesita secuencias o identificadores únicos, el usuario es libre de darle a seqgen su propio uso, así como modificar y actualizar código.

4.4. Licencia

Seqgen fue creado bajo la licencia *GNU General Public License* v3.0. Para consultar más detalles, verificar el siguiente enlace:

https://github.com/pedrocruzlopez/sequence_generator/blob/master/LICENSE

CONCLUSIONES

1. Es necesario crear un método alternativo al que actualmente usan los sistemas de gestión de bases de datos *MySQL Server* y *PostgreSQL* para generar claves subrogadas.

2. Contar con un método alternativo haría que no exista la necesidad de crear un índice extra para un campo que contenga un valor generado automáticamente, esta necesidad solo aplica a campos que no sean definidos como claves primarias, la cual utiliza un índice por defecto en su creación

3. Las operaciones de borrado se reducirían en *PostgreSQL*. Y en *MySQL* se mantendrían fluctuantes al igual que los tiempos en operaciones de inserción, estos problemas se tratarían en versiones futuras de la aplicación.

RECOMENDACIONES

1. Utilizar la actual solución de software que permite una alternativa a las implementadas nativamente en *MySQL* y *PostgreSQL*, *seqgen* mejora la gestión y control de llaves candidatas para un sistema.

2. Utilizar *seqgen* en campos de llaves candidatas de una tabla para que se evite la utilización de índices extras.

3. Se recomienda usar el diseño de *seqgen* para tablas con claves subrogadas en un modelo de negocio pequeño, no utilizarlo si por el contrario el modelo de negocio es grande y las tasas de concurrencia son grandes ya que los valores fluctuantes en *MySQL* o *PostgreSQL* no presentarían los resultados esperados, en las versiones futuras de este software se tratara problemas de concurrencia y reducciones de tiempo de operación.

BIBLIOGRAFÍA

1. BOVET, Daniel P y CESATI, Marco, *Understanding the Linux Kernel*. Sebastopol: O'Reilly Media, 2007. 924 p.

2. CORBET, Jonathan, RUBINI, Alessandro y KROAH-HARTMAN, Greg, *Linux Device Drivers*. 3. Sebastopol: O'Reilly Media, 2005. 630 p.

3. DATE, C. J y RUIZ FAUDÓN, Sergio Luis María, *Introducción a los sistemas de bases de datos*. México: Pearson Educación, 2001. 959 p.

4. DRAGOS, Paul, 2011, *Natural versus Surrogate Keys. Performance and Usability* [en linea]. Bucarest, Rumania. [consulta: 26 agosto 2016]. Disponible en: http://www.dbjournal.ro/archive/4/6_Dragos_Pop.pdf

5. FAQ: *Using Sequences in PostgreSQL*, 2016. *Neilconway.org* [en línea]. [consulta: 26 agosto 2016]. Disponible en: http://www.neilconway.org/docs/sequences/

6. Historia de las Bases de Datos – Historia de la Informática, 2011. *Histinf.blogs.upv.es* [en línea]. [consulta: 26 agosto 2016]. Disponible en: http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/.

7. MySQL: MySQL 5.7 Reference Manual: 14.8.1.5 AUTO_INCREMENT Handling in InnoDB, [sin fecha]. *Dev.mysql.com* [en línea]. [consulta: 10 septiembre 2016]. Disponible en: https://dev.mysql.com/doc/refman/5.7/en/innodb-auto-increment-handling.html

8. PostgreSQL Documentation 10: 8.1. Numeric Types, [sin fecha]. *Postgresql.org* [en línea]. [consulta: 19 septiembre 2016]. Disponible en: https://www.postgresql.org/docs/current/static/datatype-numeric.html#DATATYPE-SERIAL

9. PostgreSQL Documentation 10: 9.16. Sequence Manipulation Functions, [sin fecha]. *Postgresql.org* [en línea]. [consulta: 19 septiembre 2016]. Disponible en: https://www.postgresql.org/docs/current/static/functions-sequence.html

10. RICHARDSON, Lee, *Create Data Disaster: Avoid Unique Indexes* – (Mistake 3 of 10). *Leerichardson.com* [en línea]. 2007. [consulta: 20 agosto 2016]. Disponible en: http://www.leerichardson.com/2007/08/create-data-disaster-avoid-unique.html

11. SEDER, Luka, *Say NO to Excessive Use of Surrogate Keys if Performance Really Matters to You*. *Java, SQL and jOOQ.* [en línea]. 2016. [consulta: 27 agosto 2016]. Disponible en: https://blog.jooq.org/2016/07/20/say-no-to-excessive-use-of-surrogate-keys-if-performance-really-matters-to-you/

12. *The Database Terms of Reference (Keys)*, [sin fecha]. [en línea], UHI Millennium Institute. [consulta: 12 agosto 2016]. Disponible en: http://rdbms.opengrass.net/2_Database%20Design/2.1_TermsOfReference/r/2.2_Keys.pdf

13. The Linux Documentation Project, [sin fecha]. *Tldp.org* [en línea]. [consulta: 23 septiembre 2016]. Disponible en: http://www.tldp.org/

APÉNDICES

Apéndice 1. **Instalación de *GNU Compiler Collection***

Para instalar el compilador *GNU*, se debe ejecutar el siguiente comando en la terminal de la computadora:

Distribución basada en *Debian*:

```
$ sudo apt-get install build-essentials
```

Distribución *CentOS:*

```
$ sudo yum install gcc
```

Apéndice 2. **Instalación de *CMake***

CMake es una herramienta para compilar y distribuir software, es requisito para instalar *seqgen*, para su instalación se debe correr el siguiente comando en la terminal:

Distribución basada en *Debian*

```
$ sudo apt-get install cmake
```

Distribución *CentOS:*

```
$ sudo yum install cmake
```

Apéndice 3. **Instalación de *MySQL Server***

Para instalar el servidor de base de datos *MySQL* en un sistema operativo Linux, se debe ejecutar la siguiente secuencia de comandos en la terminal de la computadora:

```
$ sudo apt-get update
```

```
$ sudo apt-get install mysql-server
```

Correr comando de instalación segura:

```
$ mysql_secure_installation
```

Apéndice 4. **Instalación de *PostgreSQL***

Para instalar el servidor de base de datos *PostgreSQL* en un sistema operativo Linux, ejecutar los siguientes comandos:

```
$ sudo apt-get update
```

```
$ sudo apt-get install postgresql postgresql-contrib
```

Cambiar la cuenta de *PostgreSQL* en el servidor, escribiendo:

```
$ sudo apt-get update
```

Iniciar sesión:

```
$ psql
```