Universidad de las Ciencias Informáticas

Propuesta de diseño de la base de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Ireisy Bermúdez Hoyo
Lázaro Rubén García Martínez

Tutor:

Ing. Renier Sotés Mesa

Ciudad de La Habana, Cuba
Junio de 2011

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado: *"Propuesta de diseño de la base de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones"*, y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de junio de 2011.

Ireisy Bermúdez Hoyo

_____

Firma del Autor

Lázaro Rubén García Martínez

_____

Firma del Autor

Ing. Renier Sotes Mesa

_____

Firma del Tutor

AGRADECIMIENTOS

Quisiera aprovechar este espacio para agradecer a las personas que me han ayudado de una forma u otra con la realización de este trabajo. En primer lugar a mi familia por apoyarme durante todos estos años, en especial a mi mamá, que ha sido mi principal guía, a Julio por brindarme todo su cariño y apoyo incondicional, a Irely por ser mi hermanita querida. Otro agradecimiento especial va dedicado a mi novio, con quien tuve la oportunidad de realizar este trabajo, por su entrega, dedicación, comprensión, cariño y amor; a su familia por brindarme su cariño y apoyo. Agradecer también a los amigos que siempre estuvieron cuando los necesité, a los compañeros del proyecto, en especial a Sotes por su apoyo y paciencia. A todos muchas gracias.

Ireisy

Agradezco especialmente a mi mamá y mi hermana, por apoyarme y guiarme durante todos mis años de estudio y a mi novia, y más que novia, al amor de mi vida, Ireisy por saber comprenderme, guiarme, aconsejarme y acompañarme durante todos los momentos buenos y malos que hemos convivido juntos. También agradezco a mi tutor por apoyarnos enormemente durante el desarrollo de la investigación, a mis compañeros del proyecto y a todas las personas que de una forma u otra ayudaron con la creación de este trabajo.

Lázaro

DEDICATORIA

Ireisy

Dedico esta investigación a mi familia, en especial a mi mamá.

Lázaro

A mi hermana y mi mamá, por apoyarme, aconsejarme y confiar en mí durante toda mi vida como persona y estudiante. A mi amor Ire, por acompañarme, cuidarme, y amarme como me ama durante toda mi vida universitaria.

RESUMEN

El presente documento muestra los resultados de una investigación para el diseño de la base de datos para los módulos solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones, teniendo en cuenta las características funcionales y no funcionales, las técnicas y patrones utilizados, así como la validación del mismo a través de las pruebas de integridad al modelo. Se obtiene como resultado un modelo de datos normalizado y la caracterización de un conjunto de herramientas de replicación. Se propone un procedimiento que permita mejorar la disponibilidad de los datos y se tratan aspectos relacionados con la seguridad en el acceso a los mismos.

Palabras claves: Base de datos, herramientas de replicación, modelo de datos.

ABSTRACT

This document shows the results of a database design for the modules of solicitude and enrollment data for diplomatic passport system, taking into account the functional and nonfunctional features, techniques and patterns used and its validation through model testing integrity. It results in a standardized data model and characterization of a replication tools. A procedure is proposed to improve the availability of data and discusses issues related to secure access to them.

Keywords: Database, replication tools, database model.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1: SURGIMIENTO Y DESARROLLO DE LAS BASES DE DATOS 16

1.1. SURGIMIENTO DE LAS BASES DE DATOS 17
1.2. MODELOS DE DISEÑO DE BASES DE DATOS 18
1.3. NORMALIZACIÓN DE BASES DE DATOS 23
1.4. SISTEMAS GESTORES DE BASES DE DATOS 24
1.4.1. ARQUITECTURA DE TRES NIVELES 25
1.5. SISTEMA GESTOR DE BASES DE DATOS POSTGRESQL 26
1.6. ARQUITECTURA DE LAS BASES DE DATOS 29
1.7. DISPONIBILIDAD Y RÉPLICA DE DATOS 31
1.7.1. SOLUCIONES DE REPLICACIÓN EN POSTGRESQL 33
1.8. BASES DE DATOS Y SISTEMAS DE IDENTIFICACIÓN 34
1.9. AMBIENTE DE DESARROLLO 36
1.9.1. HERRAMIENTA CLIENTE PARA LA ADMINISTRACIÓN DEL SERVIDOR DE BASE DE DATOS POSTGRESQL 36
1.9.2. HERRAMIENTA CASE 36
1.9.3. LENGUAJE PROCEDURAL PL/PGSQL 37
1.10. PROPUESTA DE SOLUCIÓN TEÓRICA 37
1.11. CONCLUSIONES 37

CAPÍTULO 2: DISEÑO DE LA BASE DE DATOS Y PROPUESTA DE SOLUCIÓN 39

2.1. CARACTERÍSTICAS FUNCIONALES Y TÉCNICAS DEL SISTEMA 40
2.1.1. DESCRIPCIÓN DE LOS PROCESOS DEL SISTEMA 40
2.1.2. NECESIDADES TÉCNICAS DEL SISTEMA 42
2.2. HERRAMIENTAS DE REPLICACIÓN 43
2.2.1. SLONY-I 43
2.2.2. BUCARDO 44
2.2.3. CYBERCLUSTER 45
2.2.4. PGCLUSTER 46
2.2.5. PGPOOL-II 47
2.3. ELECCIÓN DE LA HERRAMIENTA 48
2.4. DISEÑO DE LA BASE DE DATOS 51
2.4.1. CONCEPTOS DE NEGOCIO SIGNIFICATIVOS 51
2.4.2. ESTRUCTURA GENERAL DE LA BASE DE DATOS 52
2.4.2.1. Vista por esquemas 53

2.4.2.2. Vista por sub-modelos 53
2.4.3. NOMENCLATURA 56
2.4.5. RESTRICCIONES DEL MODELO DE DATOS 59
2.4.6. NORMALIZACIÓN 60
2.4.7. PATRONES DE DISEÑO UTILIZADOS 61
2.4.8. ARQUITECTURA DE SOFTWARE 64
2.5. CONCLUSIONES 65

CAPÍTULO 3: CONFIGURACIÓN Y PRUEBA DE LA SOLUCIÓN PROPUESTA 66

3.1. PROPUESTA DE DESPLIEGUE PARA LA SOLUCIÓN 67
3.2. RECUPERACIÓN ONLINE CON PGPOOL-II 68
3.2.1. PROCEDIMIENTO DE RECUPERACIÓN 68
3.3. SEGURIDAD EN LAS BASES DE DATOS 72
3.3.1. SEGURIDAD EN POSTGRESQL 72
3.4. PRUEBAS 75
3.4.1. PRUEBAS DE INTEGRIDAD AL MODELO 75
3.4.2. VALIDACIÓN FUNCIONAL DE LA SOLUCIÓN PROPUESTA 79
3.5. ANÁLISIS DE LOS RESULTADOS ALCANZADOS EN BASE A LA SOLUCIÓN EXISTENTE 81
3.6. CONCLUSIONES 82

CONCLUSIONES GENERALES 83

RECOMENDACIONES 84

BIBLIOGRAFÍA 85

GLOSARIO DE TÉRMINOS 89

ANEXOS 90

ÍNDICE DE FIGURAS

Figura 1. Representación del modelo jerárquico 19
Figura 2. Representación del modelo de red 19
Figura 3. Representación del modelo relacional 21
Figura 4. Representación del modelo entidad relación 22
Figura 5. Arquitectura en tres niveles ANSI-SPARC 26
Figura 6. Arquitectura centralizada 30
Figura 7. Arquitectura distribuida 31
Figura 8. Modelo de dominio 52
Figura 9. Esquemas de la base de datos 53
Figura 10. Vista por sub-modelos 54
Figura 11 Sub-modelo dirección 60
Figura 12. Dependencia transitiva entre atributos 61
Figura 13. Patrón de asociación (muchos a muchos) 62
Figura 14. Patrón RBAC 63
Figura 15. Diagrama de software 64
Figura 16. Diagrama de despliegue 67
Figura 17. Error mostrado al tratar de insertar una clave primaria nula 76
Figura 18. Error mostrado al tratar de insertar una clave primaria duplicada 77
Figura 19. Error mostrado al tratar violar la restricción impuesta por el dominio dom_cod_pais 78
Figura 20. Error emitido por el gestor al tratar de violar la integridad referencial 79

ÍNDICE DE TABLAS

Tabla 1. Límites de una base de datos en PostgreSQL 29
Tabla 2 Comparación de herramientas 49
Tabla 3. Prefijos del modelo de datos 57
Tabla 4. Pruebas funcionales 79
Tabla 5. Parámetros de pgbench 81
Tabla 6 Operacionalización de las variables 90
Tabla 7. Dominios de la base de datos 98
Tabla 8. Parámetros utilizados para la instalación de PostgreSQL 101

INTRODUCCIÓN

La información es uno de los recursos más valiosos de una organización. En un inicio era procesada de forma manual y los documentos que se generaban, eran guardados en archivos físicos, lo que provocaba retrasos en las búsquedas y el trabajo era engorroso. Debido al desarrollo tecnológico de hoy en día, la mayoría de los datos están en formato digital, lo que ofrece un amplio rango de soluciones a los problemas asociados con su almacenamiento, los procedimientos tradicionales de manipulación y control han sido sustituidos por métodos automatizados de almacenamiento que proveen un ambiente de trabajo para la toma de decisiones y el manejo de resultados estadísticos, además que permite ejecutar las tareas en menor tiempo y aportando mayor productividad.

Dada su importancia como base para la toma de decisiones estratégicas y su relevancia a la hora de establecer ventajas competitivas, es necesario contar con una eficiente gestión de la información, ya que en la actualidad los volúmenes de información con que se trabaja son cada vez más elevados, por lo que es necesario contar con los medios y técnicas que permitan almacenarla de la manera más adecuada y que permitan gestionar los datos a lo largo de su evolución, permitiendo conocer qué tratamiento darles, y cómo recuperarlos en caso de que sea necesario volver a trabajar con ellos.

La información tiene particular connotación en los sistemas para la emisión de documentos de identificación que en la actualidad han tenido un gran auge. Estos sistemas manejan información sensible, estratégica para el negocio, pues contienen datos personales de las personas y de la organización para la que han sido creados, lo que conlleva a elevar los controles para lograr la gestión de identidad de forma confiable y evitar su falsificación, alteración o uso fraudulento. Como parte del desarrollo tecnológico, el Ministerio del Poder Popular para las Relaciones Exteriores de la República Bolivariana de Venezuela (MPPRE), se encuentra inmerso en un proceso de transformación y modernización de la emisión de pasaportes diplomáticos, de servicio y acreditaciones que se lleva a cabo en el Área de Pasaportes Oficiales y en el Área de Inmunidades y Privilegios respectivamente. Teniendo en cuenta que estos documentos en su mayoría se emiten a personal diplomático y que por la función que desempeñan tiene ciertos privilegios, la protección de los datos es un factor clave y fundamental debido al valor que representa.

Las vulnerabilidades en la emisión de estos documentos pueden socavar la integridad de los mismos, de ahí la necesidad de garantizar un nivel de seguridad desde su concepción, y en cada uno de los procesos que se realizan para su emisión. Actualmente, los procesos de emisión de pasaportes y acreditaciones tienen lugar cuando el solicitante presenta en las oficinas un conjunto de recaudos que son necesarios para el trámite, los cuales son revisados para verificar su veracidad. Una vez revisados, se procede con el registro de los datos personales y con la captura de los datos biométricos, posteriormente se realiza el ensobrado y entrega de estos documentos.

El procedimiento actual presenta un conjunto de problemas que justifican cambios y modificaciones a lo que se está llevando a cabo en el ministerio. Para el caso de los pasaportes, tanto diplomáticos como de servicio, se utilizan dos sistemas informáticos, uno de ellos, desarrollado en Access, utilizado sólo para consulta, brinda información desactualizada, y su uso es inadecuado para la emisión de este tipo de documentos que requieren un tiempo de respuesta rápido. El otro sistema, desarrollado en Filemaker Pro, presenta limitaciones en el procesamiento de las búsquedas, lo que lo hace un sistema lento. Además, aunque permite al usuario registrar, eliminar y consultar los datos está obsoleto. Por otra parte, es necesario persistir la información de los pasaportes que son anulados para enviar un reporte al Servicio Administrativo de Identificación, Migración y Extranjería (SAIME), lo que ocasiona capacidad de respuesta lenta y deriva invertir mayor tiempo y esfuerzo, provocando inconformidad en la emisión de reportes. En el caso de acreditaciones, los trámites se realizan de forma manual porque el sistema informático con que cuentan presenta errores de validación y es ineficiente, lo que retrasa el proceso al tener que realizar búsquedas en archivos físicos. El documento de acreditación se registra en Microsoft Word, dificultando el manejo de los datos que se usan con mayor frecuencia.

Los mecanismos de seguridad que se tienen en cuenta para la emisión de estos documentos son insuficientes, pues no se realiza validación de los datos, ni de la identidad del ciudadano y no tienen un control estricto de los pasaportes y acreditaciones emitidos anteriormente. Ambos documentos no se ajustan a las normas internacionales en materia de documentos de viaje. Los clientes necesitan una estructura que permita almacenar la información de una forma eficiente y sencilla, que agilice el proceso de búsquedas y las operaciones que se realizan sobre ella; a su vez, necesitan lograr disponibilidad de la

información, debido a que en ocasiones se realizan trámites de pasaportes y acreditaciones con carácter urgente. Para ello cuentan con dos servidores de datos, por lo que es necesario establecer un mecanismo que permita mejorar la disponibilidad de la información, y ante fallas, restablecer el sistema en un tiempo razonable para continuar brindado servicio.

Teniendo en cuenta la situación actual se hace necesario plantear el siguiente **problema científico:** ¿Cómo garantizar la persistencia de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela?

Con el propósito de guiar y dar cumplimiento a la presente investigación se plantea como **objetivo general:** Mejorar la disponibilidad de los servidores de datos y diseñar el modelo persistente para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.

Los **objetivos específicos** quedan definidos de la siguiente forma:

- Modelar la base de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.
- Proponer un procedimiento para mejorar la disponibilidad de los servidores de datos.
- Validar la persistencia de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.

Objeto de estudio: Sistemas de bases de datos.

Campo de acción: La persistencia de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de La República Bolivariana de Venezuela.

Hipótesis: La implantación de una herramienta de replicación y el diseño de un modelo de datos permitirán la persistencia de los datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de La República Bolivariana de Venezuela.

Se definen las siguientes variables de la investigación:
Independientes: Herramienta de replicación
Modelo de datos

Dependiente: Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones.

En el **anexo 1** se encuentra la operacionalización de las variables.

Para evaluar el cumplimiento de los objetivos planteados se definen las siguientes **tareas de la investigación:**

- Identificar los requerimientos no funcionales para la persistencia de los datos de los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de La República Bolivariana de Venezuela.
- Identificar los elementos que tributan a las entidades del modelo de datos del módulo Solicitud.
- Modelar los conceptos de negocio del módulo Solicitud.
- Identificar los elementos que tributan a las entidades del modelo de datos del módulo Enrolamiento.
- Modelar los conceptos de negocio del módulo Enrolamiento.
- Conformar el Diccionario de datos a partir de las restricciones identificadas en el negocio.
- Modelar en el esquema de datos las entidades que permiten almacenar los datos tomados del SAIME.
- Diseñar la gestión de identidad de los ciudadanos del sistema, permitiendo separar los datos de los ciudadanos venezolanos y extranjeros.
- Indexar los principales campos de búsqueda dentro del modelo de datos.

- Caracterizar las técnicas de replicación para los sistemas de bases de datos.
- Caracterizar herramientas que permitan la réplica de datos para el Sistema gestor de bases de datos PostgreSQL.
- Identificar la técnica de replicación para la estructura persistente del Sistema de Emisión de Pasaportes Diplomáticos, de servicio y Acreditaciones de la República Bolivariana de Venezuela.
- Identificar la herramienta que permita la replicación de los datos del Sistema de emisión de pasaportes diplomáticos, de servicio y acreditaciones de la República Bolivariana de Venezuela.
- Establecer un procedimiento para aumentar la disponibilidad de los datos del Sistema de emisión de pasaportes diplomáticos, de servicio y acreditaciones de la República Bolivariana de Venezuela.
- Montar los gestores de Bases de datos.
- Instalar la herramienta de replicación en los servidores.
- Configurar la herramienta de replicación y los gestores de datos.
- Montar el esquema persistente en los servidores de datos.
- Insertar los datos de carga en las bases de datos.
- Aplicar pruebas para validar la solución propuesta y el diseño de la base de datos.

Métodos de investigación:

Teóricos:

- *Histórico-Lógico*: Permitirá identificar buenas prácticas para el diseño de bases de datos, las diferentes técnicas que garanticen la replicación de la información entre los servidores de datos y los mecanismos de recuperación ante fallos.
- *Modelación:* Permitirá identificar en los requerimientos del software entidades o elementos que conformarán el modelo de datos y sus relaciones. Se modelará la distribución de los nodos físicos de base de datos y la comunicación entre ellos.

Empíricos:

- *Observación:* Se observará el comportamiento de los servidores de base de datos en cuanto a replicación de los datos, balanceo de carga, cantidad de conexiones presentes, consultas

realizadas, con el objetivo de identificar elementos críticos que pueden dificultar su funcionamiento.

El documento se encuentra estructurado en tres capítulos. En el primer capítulo se definirán conceptos claves que guiarán la investigación. Se analizarán diferentes modelos de diseño de bases de datos para determinar el más adecuado para el diseño de la base de datos. Se analizará la arquitectura de un sistema gestor de bases de datos y se expondrán las principales características que presenta el sistema gestor de bases de datos PostgreSQL, además, serán caracterizadas las diferentes técnicas de replicación de datos y las soluciones de replicación que ofrece PostgreSQL. Se definirán las herramientas para el desarrollo de la investigación.

En el segundo capítulo se caracterizarán diferentes herramientas de replicación de datos para identificar la que se ajuste a las necesidades del sistema. Se identificarán los conceptos en los requerimientos del software que tributarán a posibles entidades para la conformación del modelo de datos. Se modelará la base de datos del sistema a partir de los conceptos identificados y las relaciones entre ellos, aplicándose patrones de diseño de bases de datos. Se documentará el modelo con el objetivo de caracterizar las entidades, atributos y las restricciones del mismo.

En el capítulo tres se definirá el ambiente donde será desplegada la solución. Se describirá el procedimiento para la restauración de un servidor de bases de datos. Será validado el diseño del modelo y la solución propuesta, teniendo en cuenta aspectos como integridad, replicación y recuperación, plasmándose los resultados obtenidos. Además se realizará un análisis donde se valorarán los resultados alcanzados, en base a las mejoras y ventajas respecto a la solución existente en el MPPRE.

CAPÍTULO 1

SURGIMIENTO Y DESARROLLO DE LAS BASES DE DATOS

En un inicio la información era procesada de forma manual, donde los procesos de búsqueda de datos eran engorrosos al tener que buscar en archivos físicos. El surgimiento y desarrollo de las bases de datos dio solución a este problema, pues la información era almacenada en equipos de cómputo, donde el acceso a la misma era más cómodo, rápido y seguro. El acelerado avance en las tecnologías de la información y las comunicaciones, unido a la necesidad de obtener mejores rendimientos en los sistemas y elevar la disponibilidad de la información, propició la ampliación de soluciones que lograran solventar las necesidades del usuario en cuanto a estos términos. Es por esto que es necesario conceptualizar elementos claves en la investigación, para poder realizar un buen diseño de la bases de datos y garantizar factores que son de vital importancia en la emisión de los documentos de pasaportes y acreditaciones.

1.1. Surgimiento de las Bases de datos

La creciente necesidad de almacenar grandes volúmenes de datos, propició el surgimiento de sistemas automatizados que permitieran agilizar el tratamiento de la información y persistieran los datos para su posterior manipulación, lo que se conoce hoy en día como base de datos. Una base de datos es un *"(…) conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora".* (GARCIA, 2005)

Los métodos de almacenamiento de datos han evolucionado desde los métodos manuales para archivar, procesar y conseguir un volumen importante de información, hasta los sistemas de almacenamiento de datos con que se cuenta en la actualidad. Con el surgimiento en 1890 de la Máquina tabuladora de Herman Holleric[1] se procesaba la información en tarjetas que permitían almacenar los datos en código binario. Más tarde, en la década de 1950 la información era almacenada en cintas magnéticas las que permitían generar copias de respaldo de la información. Aunque las cintas se leían en forma ordenada, la secuencialidad en el proceso de lectura era desventajoso, pues era necesario realizar un recorrido completo para obtener un dato que tal vez, estuviese al final de la cinta, lo cual fue resuelto años más tarde con el surgimiento de los discos duros y con la evolución de los ordenadores, obteniéndose de esta forma los primeros modelos de diseño de base de datos jerárquico y de red.

Ya para los años 70 aparece el modelo relacional, que dio origen a las bases de datos relacionales, propuesto por Edgar Frank Codd[2]. El modelo relacional permite minimizar la información redundante, lo que garantiza una mayor integridad de los datos. Proporciona un acceso eficaz a los datos y una fácil compresión y manipulación de los mismos, lo que lo convierte en uno de los modelos de base de datos más utilizados y difundidos en la actualidad. Para los años 80 se inician investigaciones relacionadas con las Bases de datos orientadas a objetos en busca de compensar las deficiencias de los modelos anteriores y se crea el lenguaje estructurado de consulta, de sus siglas en inglés Structured Query Language (SQL), que permite realizar diferentes operaciones sobre las bases de datos relacionales como acceder, modificar y recuperar información. Este lenguaje es ampliamente utilizado en el desarrollo de software a nivel mundial.

[1] Herman Hollerith (1860 -1929) estadístico estadounidense que inventó la máquina tabuladora.
[2] Edgar Frank Codd (1923-2003) científico informático inglés reconocido por sus aportes a las bases de datos relacionales.

1.2. Modelos de diseño de bases de datos

Las bases de datos deben estar respaldadas por un diseño que garantice una buena representación y fácil manipulación de los datos, es por ello que han surgido diferentes modelos de diseño de bases de datos. Un modelo de datos es una abstracción de un conjunto de reglas y conceptos del mundo real. Existen diferentes modelos de datos y su utilización tiene lugar durante el proceso de diseño de una base de datos. Los modelos conceptuales permiten crear una representación de la realidad con un elevado nivel de comprensión, sin tener en cuenta la estructura de almacenamiento de los datos; mientras que en los modelos lógicos la estructura de los datos está muy ligada a la estructura de almacenamiento y al sistema gestor de bases de datos que se utilice. Inicialmente se utiliza el modelo conceptual para modelar una representación de la realidad y posteriormente el modelo conceptual es transformado en un modelo lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad.

Modelo jerárquico

Este modelo organiza los datos en una estructura similar a la de un árbol, donde en el nodo superior se encuentra el padre y en los nodos inferiores los hijos como se presenta en la **figura 1**. En este tipo de base de datos las relaciones entre los nodos es de tipo padre/hijo. Los nodos representan los tipos de registro o entidades, y los arcos las relaciones jerárquicas entre los nodos. Los datos son almacenados en ficheros compuestos por registros, que a su vez tienen un conjunto de campos. En este modelo, los datos solo pueden ser consultados desde un nodo hijo hacia un nodo padre, en caso contrario habría que realizar una búsqueda secuencial por todos los registros de la base de datos. La poca flexibilidad de este modelo genera redundancia de la información para una situación de la vida real que no responda a una jerarquía. No permite representar interrelaciones (n:m)[3] y las actualizaciones pueden generar problemas debido a las restricciones del modelo. Uno de los sistemas más populares basados en el modelo jerárquico fue IMS (Information Management System) de International Business Machines (IBM).

[3] Relación de muchos a muchos.

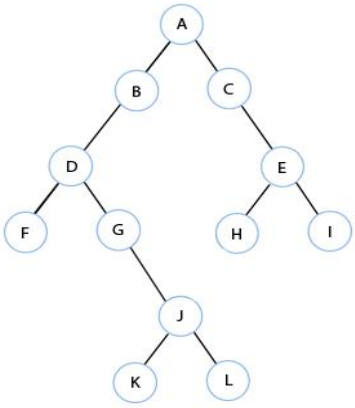


Figura 1. Representación del modelo jerárquico

Modelo de red

Este modelo es muy similar al modelo jerárquico con la diferencia de que los nodos hijos pueden estar relacionados y tener varios padres, formando una estructura de grafo como se muestra en la **figura 2**. Fue una mejora al modelo jerárquico, ya que ofrecía una solución al problema de la redundancia de los datos, aunque existen dificultades a la hora de administrar la información almacenada en una base de datos de red.

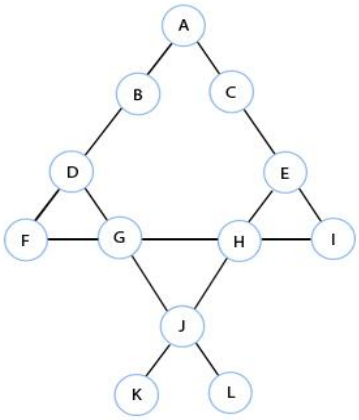


Figura 2. Representación del modelo de red

Modelo relacional

El modelo relacional fue propuesto en los años 70 por Edgar Frank Codd y *"(…) es uno de los modelos matemáticos más importantes y actuales para la representación de bases de datos"* (GARCIA, 2005). Se basa en la teoría matemática de las relaciones, donde *"una relación es un conjunto de tuplas bien definida, con una cantidad de atributos fijos, donde cada valor dado para cada atributo de cada tupla pertenece al dominio de valores para dicho atributo"*. (ROZIC, 2004)

En este modelo tanto las entidades como las relaciones que se establecen, se representan a través de tablas como lo muestra la **figura 3**, las cuales pueden variar en el transcurso del tiempo y poseen un nombre único que las identifican. Las tablas están compuestas por filas o tuplas, las cuales representan las ocurrencias de los objetos almacenados en una relación, y por columnas o atributos que no son más que los valores pertenecientes a un dominio que puede tomar un objeto. Las tuplas se identifican mediante una llave o clave primaria, lo que las hace únicas dentro de la relación. En una relación no puede existir más de una columna con el mismo nombre, no existen dos tuplas iguales, el orden tanto de las columnas como de las tuplas no es significativo y todos los atributos que componen las tuplas son atómicos, o sea, en cada posición fila-columna solo existe un único valor, no un conjunto de valores.

Este modelo debe cumplir un grupo de restricciones o reglas con el objetivo de apoyar la integridad de la base de datos:

- Restricción de dominio: el valor de cada atributo debe ser un valor atómico del dominio al que se encuentra sujeto el atributo.
- Restricción de clave: no pude existir más de una tupla perteneciente a una relación con la misma clave.
- Integridad de la entidad: ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo.
- Integridad referencial: Si una relación A posee una clave foránea o ajena, que es la clave primaria de otra relación B, entonces el valor que posea dicha clave foránea debe concordar con el valor de la clave primaria de B o ser nulo.

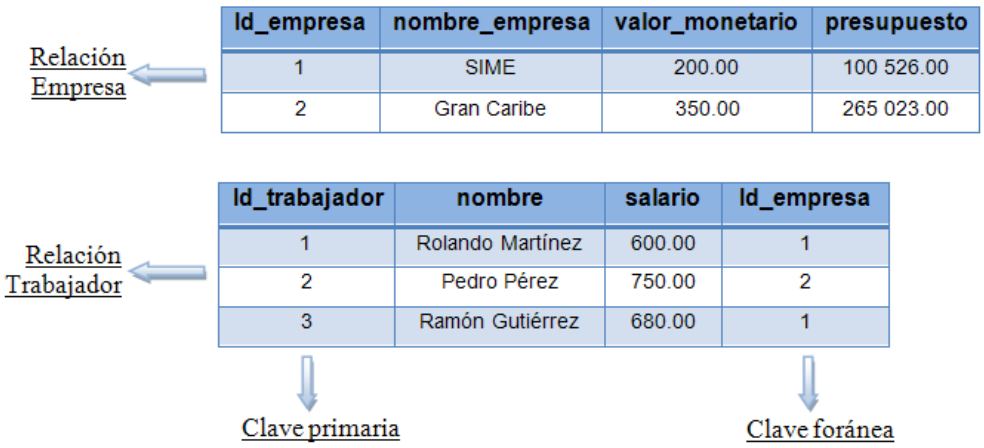


Figura 3. Representación del modelo relacional

Modelo entidad relación (Modelo ER)

Este modelo fue desarrollado por Peter P. Chen en el año 1976. Está basado en una percepción del mundo real, que consta de un conjunto de objetos básicos llamados entidades y de las interrelaciones que existen entre estos objetos. Según Peter P. Chen las entidades quedan definidas de la siguiente forma: *"(…) elementos que se pueden identificar claramente"* (ROZIC, 2004). Las entidades pueden variar según el fenómeno de la realidad que estén representando, por ejemplo, si se quisieran capturar los datos concernientes a un estudiante perteneciente a una universidad, tal vez sería necesario el nombre, el índice académico y su sexo, mientras que si el mismo estudiante fuese un cliente de un banco, los datos bien podrían ser, el nombre, su número de cuenta y su número de identificación personal. A este conjunto de datos se le conoce como atributos o propiedades. Una propiedad *"(…) es la unidad menor de información sobre un objeto"* (GARCIA, 2005), permite identificar a una entidad con respecto a otra y puede tomar diferentes valores pertenecientes a un dominio. Un dominio es el *"(…) conjunto de valores válidos que puede tomar una propiedad".* (ROZIC, 2004)

Este modelo usa diagramas para representar la estructura natural de los datos, donde los rectángulos representan las entidades y los rombos representan las interrelaciones, que son enlazadas con sus entidades, formando arcos, donde el grado de la interrelación es el indicado en el arco. Estas relaciones pueden ser de uno a uno (1:1) cuando una entidad A solo tiene una referencia de un objeto perteneciente

a una entidad B, de uno a muchos (1:n) y de muchos a muchos (n:m); también se les conoce como cardinalidad. La **figura 4** muestra las entidades empresa y trabajador, la relación y cardinalidad que existe entre ellas.

Figura 4. Representación del modelo entidad relación

Modelo orientado a objetos

Cuando se trata de aplicaciones complejas o sofisticadas como, experimentos científicos o sistemas multimedia, los requerimientos y las características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de gestión: la estructura de los objetos es más compleja, las transacciones son de larga duración, se necesitan nuevos tipos de datos para almacenar imágenes; por lo que las bases de datos orientadas a objetos (BDOO) fueron creadas para satisfacer estas necesidades. *"Las bases de datos orientadas a objetos son aquellas cuyo modelo de datos está orientado a objetos y almacenan y recuperan objetos en los que se almacena estado y comportamiento"* (Bases de datos Orientadas a Objetos y Bases de datos Objeto-Relacionales). Surgieron a raíz de los problemas originados por los modelos clásicos a la hora de representar y manipular cierta información, que aunque pueden representar una gran cantidad de datos, las operaciones que se pueden realizar sobre ellos son demasiado simples. Por otra parte, pretenden aprovechar las posibilidades que brinda el paradigma orientado a objetos como el encapsulamiento[4], la herencia[5] y el polimorfismo[6], donde las clases utilizadas en cualquier lenguaje de programación orientado a objetos serían las mismas clases utilizadas en una base de datos orientada a objetos, de esta forma se utilizaría el mismo modelo y se eliminarían las transformaciones entre modelos, como la transformación del modelo relacional al modelo orientado a objetos.

[4] "Mecanismo que oculta los detalles de cómo hacer cierta tarea y proporciona una forma de interactuar con un objeto". (BECERRIL, 1998)
[5] "Permite definir propiedades y métodos de un objeto a partir de las propiedades y los métodos de otros objetos". (BONANATA, 2003)
[6] "Capacidad que tienen los objetos para reaccionar de manera diferente en función del tipo de estímulo que reciben". (BECERRIL, 1998)

1.3. Normalización de bases de datos

La normalización se ha desarrollado con el propósito de crear estructuras de datos eficientes que eviten las anomalías que ocurren durante la actualización de los datos. Este concepto fue introducido por Edgar F. Codd y no solo es aplicable a sistemas relacionales. Este proceso permite eliminar la información redundante en una base de datos e involucra varias fases, que al completarse, se dice que la relación está en una de las formas normales que se presenta a continuación:

- Primera Forma Normal (1FN): Una relación se encuentra en 1FN cuando todos los atributos que componen las tuplas son atómicos.
- Segunda Forma Normal (2FN): Una relación se encuentra en 2FN si, y solo si, se encuentra en 1FN y los atributos no llaves, son funcional y completamente dependientes de la clave primaria, por lo general compuesta. Una relación que posea una única clave primaria se encuentra en 2FN.
- Tercera Forma Normal (3FN): Una relación se encuentra en 3FN si, y solo si, se encuentra en 2FN y no existe dependencia transitiva entre los atributos no llaves respecto a la llave primaria. Este procedimiento elimina la transitividad entre los atributos.
- Forma normal de Boyce-Codd (FNBC): "Una relación R está en FNBC si, y solo si, cada determinante[7] es una superllave[8] (candidata[9] o primaria)". (GARCIA, 2005)

Hoy en día la mayoría de los diseñadores de bases de datos normalizan los modelos hasta la 3FN debido a su fácil modo de aplicación. Sin embargo, las base de datos aún pueden ser vulnerables a algunas anomalías menos comunes, lo que hace que tenga que aplicarse una normalización más compleja y técnica, que puede terminar en complicados modelos de datos, difíciles de aplicar, de mantener y de usar. En algunos casos, pueden provocar un rendimiento más bajo que los diseños menos completamente normalizados. (STEPHENS, 2009)

[7] "Un determinante es cualquier atributo o conjunto de atributos del cual depende funcional y completamente cualquier otro atributo". (GARCIA, 2005)
[8] Una superclave es un conjunto de atributos que permite identificar a cada tupla de forma única.
[9] Una llave candidata es aquel subconjunto, formado a partir de una superclave, que no puede ser superclave.

1.4. Sistemas gestores de bases de datos

Un Sistema gestor de base de datos (SGBD) es aquel programa o conjunto de ellos que posibilitan la creación, el almacenamiento, el procesamiento y consulta de la información almacenada en una o varias bases de datos, de forma segura y eficiente, sin tener que conocer el modo de almacenamiento de los mismos, ni los métodos de acceso empleado.

Los SGBD deben cumplir un conjunto de objetivos que faciliten el tratamiento de los datos y el proceso de diseño de las aplicaciones. Estos objetivos son:

- Independencia de los datos y los programas de aplicación: es la capacidad de modificar la estructura de almacenamiento de los datos y el método de acceso a los mismos, sin que las aplicaciones se vean afectadas, de ahí que este sea el objetivo fundamental de los sistemas de base de datos.
- Minimización de la redundancia: minimizar la redundancia no significa eliminarla, pues aunque las bases de datos están definidas como no redundantes, en ocasiones existe una redundancia no significativa que permite disminuir los tiempos de respuesta al acceder a la información o para simplificar el método de direccionamiento.
- Integridad de los datos: consiste en garantizar la consistencia de los datos, con el objetivo de que durante todo momento sean correctos. La integridad está muy vinculada con la minimización de la información redundante, pues cuanta menor redundancia de los datos exista, mayor será la integridad de los mismos. Para aquellos casos en los que no se ha podido eliminar la redundancia, deben crearse los mecanismos que permitan la actualización de los datos repetidos de forma simultánea.
- Integración y sincronización de las bases de datos: un sistema de base de datos debe ser capaz de dar respuesta a las peticiones que son realizadas por diferentes usuarios sobre los mismos datos aunque estos estén almacenados con una determinada estructura y representación. A esto se le conoce como integración y está muy relacionada a la sincronización que consiste en garantizar un acceso múltiple y simultáneo a las bases de datos, de forma tal que los datos puedan ser compartidos por diferentes usuarios a la vez, pues es muy común que existan un conjunto de

usuarios accediendo a los mismos datos de forma concurrente, pero con diferentes puntos de vista.

- Seguridad y recuperación: la información que es almacenada en una base de datos puede tener un elevado valor para una organización, empresa o institución, de ahí que los SGBD deben contar con mecanismos de seguridad que garanticen el acceso a los datos de forma autorizada por diferentes usuarios y con los privilegios y permisos definidos. Por otra parte, deben poseer métodos que permitan realizar copias de respaldo a los datos y debe ser capaz de restaurar la base de datos llevándola a un estado consistente ante determinadas situaciones que afecten la integridad o disponibilidad de los mismos.
- Facilidad de manipulación de la información: un SGBD debe permitir a los usuarios realizar búsquedas sobre los datos de forma rápida y simple, aislándolo de las complejidades del tratamiento de los ficheros y del direccionamiento de los datos, donde los filtros de búsquedas empleados no afecten los tiempos de respuesta.

1.4.1. Arquitectura de tres niveles

El grupo ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee)[10] propuso en 1975, una arquitectura de tres niveles para los sistemas de bases de datos:

- Nivel interno: referido al almacenamiento físico de los datos en memoria. Se representa mediante el esquema interno que describe la estructura física de almacenamiento de la base de datos.
- Nivel externo: está relacionado con la forma en que los datos son percibidos por cada usuario de manera individual. Se describen varios esquemas externos o vistas de usuarios.
- Nivel conceptual: es un nivel intermedio entre los anteriores donde se describe la estructura total de la base de datos mediante el esquema lógico o conceptual.

La arquitectura de tres niveles tiene entre sus objetivos, separar los programas de aplicación de la base de datos física. Un nivel constituye una representación de forma abstracta de la información. Entre los niveles se observan correspondencias entre el nivel externo y conceptual, y entre el nivel conceptual e

[10] ANSI-SPARC es un estándar abstracto del diseño para Sistema de gerencia de base de datos (DBMS).

interno, como lo muestra la **figura 5**, que están dadas por la necesidad de lograr una independencia entre los datos, es decir que se puedan realizar cambios en cierto nivel sin que pueda afectarse el resto.

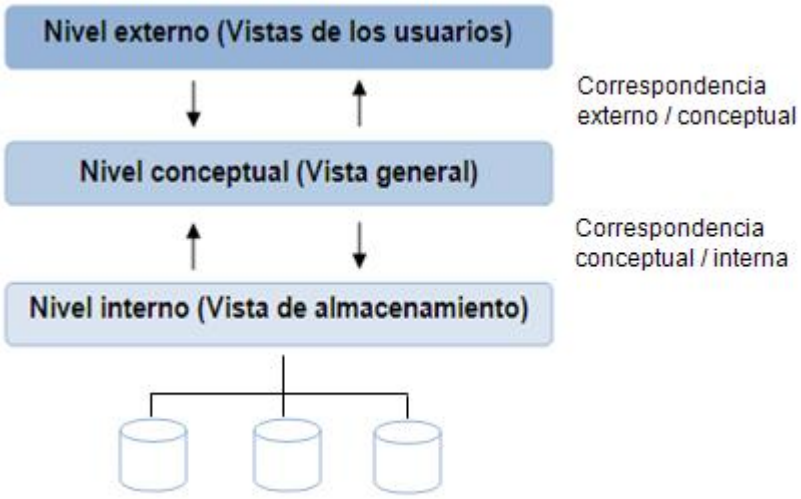


Figura 5. Arquitectura en tres niveles ANSI-SPARC

1.5. Sistema gestor de bases de datos PostgreSQL

PostgreSQL[11] es un sistema gestor de bases de datos objeto-relacional basado en la versión 4.2 de Postgres, desarrollado en la Universidad de California en Berkeley, EE.UU. Se encuentra distribuido bajo la licencia Distribución de Software Berkeley, de sus siglas en inglés Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Se caracteriza por su estabilidad, potencia, robustez y facilidad de administración. Presenta un buen rendimiento ante bases de datos que posean un elevado volumen de datos y una alta concurrencia de usuarios accediendo al mismo tiempo. Es el sistema gestor de bases de datos de código abierto más potente del mercado. Entre sus características se encuentran:

- Compatibilidad con tipos de datos especificados por el estándar SQL: bigint, bit, boolean, char, character varying, date, double precision, integer, interval, numeric, decimal, real, xml.

[11]La versión de PostgreSQL utilizada para la investigación es la 8.3.

- Provee un gran número de funciones y operadores para los tipos de datos: operadores lógicos, operadores de comparación, operadores y funciones matemáticas, funciones geométricas.
- Permite el manejo de consultas complejas, llaves foráneas, vistas, disparadores, reglas, sub-selects y procedimientos almacenados.
- Permite la herencia entre tablas, mecanismo que puede ser muy empleado por los diseñadores de base de datos.
- Permite definir funciones que pueden ser escritas en otros lenguajes genéricamente denominados lenguajes procedurales. Hay cuatro lenguajes procedurales actualmente disponibles en la distribución de PostgreSQL: PL/pgSQL[12], PL/Tcl[13], PL/Perl[14], PL/Python[15].
- Permite crear diferentes esquemas dentro de una base de datos, los que pueden ser utilizados para organizar los objetos que la conforman en grupos lógicos, y hacerlos más controlables.

Cumple con las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad):

- Atomicidad: es la propiedad que asegura que todas las operaciones que se ejecutan en una misma transacción se realizan o ninguna se realiza, en caso de que ocurra algún error.
- Consistencia: es la propiedad que asegura que una transacción solo termina si los datos son consistentes.
- Aislamiento: es la propiedad que asegura que una transacción es independiente de otras transacciones.
- Durabilidad: es la propiedad que asegura que cuando una transacción termina, el resultado es perdurable.

PostgreSQL provee herramientas para manejar el acceso concurrente a los datos. Internamente la consistencia es mantenida usando el modelo de Control de concurrencia muti-versión, de sus siglas en

[12] PL/pgSQL es un lenguaje procedural que puede ser usado para crear funciones y disparadores, hereda los tipos de datos, operadores y funciones de SQL.
[13] PL/Tcl es un lenguaje procedural que habilita el lenguaje Tcl, en inglés Tool Command Language que puede ser usado para escribir funciones y procedimientos de disparadores.
[14] PL/Perl es un lenguaje procedural que le permite a PostgreSQL escribir funciones en el lenguaje de programación Perl.
[15] PL/Python es un lenguaje procedural que le permite a PostgreSQL escribir funciones en el lenguaje de programación Python.

inglés Multiversion Concurrency Control (MVCC), que permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos, pues realiza copias de los datos de forma paralela, haciendo un control de concurrencia entre las distintas versiones que se van escribiendo. Además este gestor de bases de datos garantiza la integridad de sus datos a través del método Write-Ahead Logging (WAL), que consiste en almacenar y sincronizar todas las operaciones que alteran los datos en forma de log binarios, antes de que estos cambios sean aplicados a los datos, lo que garantiza la recuperación automática del sistema en caso de fallas. Estos ficheros son segmentos de 16Mb que se escriben y sincronizan en disco antes que se modifiquen los ficheros de las tablas, y su función principal es la recuperación a un estado consistente de forma automática, en caso de que ocurra algún fallo en el sistema, proceso conocido como Point-In-Time Recovery (PITR). Como máximo se almacenan 8 ficheros por defecto, y aquellos que no son necesarios, son eliminados, generando nuevos ficheros de WAL.

En cuanto al particionamiento de tablas, PostgreSQL lo maneja utilizando el mecanismo de la herencia. Este procedimiento consiste en dividir tablas muy grandes en piezas más pequeñas según criterios de agrupación, lo que permite reducir la cantidad de datos a recorrer en cada consulta, de forma que cada transacción realizada en una tabla padre se redirecciona de manera automática a una tabla hija que contiene un menor volumen de datos, con lo que se mejoran los tiempos de respuesta y aumenta el rendimiento.

PostgreSQL utiliza un modelo cliente/servidor en el cual existe un proceso servidor o postmaster (backend) y un grupo de aplicaciones clientes (frontend) realizando peticiones al servidor, el cual se encarga de atender y dar respuesta a dichas peticiones. Dentro de las ventajas de utilizar esta arquitectura se encuentra la independencia entre servidores y clientes, pues para los clientes es totalmente transparente la ubicación geográfica del servidor, así como la plataforma en la que se esté ejecutando, en este caso, las comunicaciones se realizan a través del protocolo de control de transmisión (de sus siglas en inglés Transmission Control Protocol, TCP) y el protocolo de internet (de sus siglas en inglés Internet Protocol, IP) comúnmente denominado TCP/IP; además tanto clientes como servidores no tienen por qué estar funcionando en un mismo host. El postmaster es el proceso principal de PostgreSQL y es el encargado de escuchar las conexiones entrantes de clientes, creando procesos hijos que se

encargarán de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.

Una base de datos en PostgreSQL presenta los límites mostrados en la **tabla 1**:

Límite	Valor
Tamaño máximo de una base de datos	Ilimitado, depende del sistema de almacenamiento que se utilice.
Tamaño máximo de una tabla	32 TB
Tamaño máximo de una fila	400 GB
Tamaño máximo de un campo	1GB
Número total de filas por tabla	Ilimitado
Máximo número de columnas por tabla	250-1600
Máximo número de índices por tabla	Ilimitado

Tabla 1. Límites de una base de datos en PostgreSQL[16]

1.6. Arquitectura de las bases de datos

Inicialmente los datos eran almacenados de forma centralizada en un ordenador central de alto rendimiento, encargado del control y acceso a los datos por parte del usuario en las diferentes terminales conectadas a él, de esta forma se puede administrar de manera centralizada todas las aplicaciones de un sistema, surgiendo así las bases de datos centralizadas, que almacenan sus datos en un sólo lugar físico, desde donde se gestionan la totalidad de los recursos. En un modelo centralizado, el rendimiento del sistema depende de la capacidad de procesamiento del ordenador central. Tienen como desventajas que al concentrar todos los recursos en un solo ordenador, si este falla, ya sea por problemas técnicos o por desastres naturales o provocados por el hombre, los clientes quedarían incomunicados y la disponibilidad de la información se vería afectada; además, devolver el sistema a su normal funcionamiento sería un tanto difícil. La **figura 6** muestra como están distribuidos los nodos y los datos en una arquitectura centralizada, en la cual los clientes acceden a un servidor central que se conecta a la base de datos y es el encargado de administrar la misma.

[16] Tomado de la FAQ de PostgreSQL.

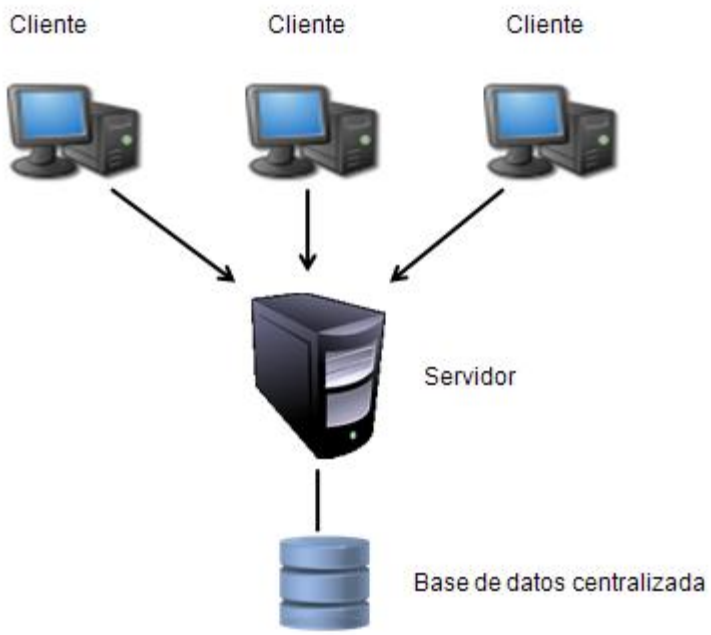


Figura 6. Arquitectura centralizada

Sin embargo después de experimentar un tiempo con este enfoque y a medida que el usuario necesitaba aplicaciones con mayor rendimiento y funcionalidades, surgió la necesidad de integrar información proveniente de diversas fuentes, es decir, manejar datos provenientes de varios sistemas de cómputo que se encontraban de forma independiente y comunicados a través de una red como muestra la **figura 7**. El enfoque distribuido es adaptable a la estructura de cada organización, donde se permite una autonomía local, haciendo que los usuarios tengan control local de los datos con los que interactúan, lo que conlleva a extremar las medidas de seguridad. Todas las acciones que se realicen para el manejo y distribución de la información son transparentes al usuario, proporcionando una independencia de los datos en el ambiente distribuido, tanto física como lógica. Los usuarios no necesitan tener conocimiento de la ubicación de los datos y los cambios realizados en la estructura lógica de la base de datos no deben afectar sus aplicaciones.

La distribución de los datos permite, que al estar ubicados en diferentes sitios de la red, una falla en uno de los servidores no implica el colapso total del sistema. Pero los datos duplicados pueden representar un problema, es ahí entonces la importancia de lograr la mayor consistencia de los mismos. Se obtienen

mejoras en el rendimiento, pues los datos son almacenados y usados donde son generados, lo cual permitirá distribuir la complejidad del sistema en los diferentes sitios de la red. Los datos generalmente se ubican cerca del sitio con mayor demanda y en ocasiones se distribuyen las cargas entre los servidores para optimizar su labor. Las consideraciones adicionales que supone el control de concurrencia, la implementación de mecanismos que garanticen la consistencia, actualización y seguridad de los datos suponen, como es de esperar, un aumento en los costes de mantenimiento, siendo esta una de las desventajas de los sistemas distribuidos.

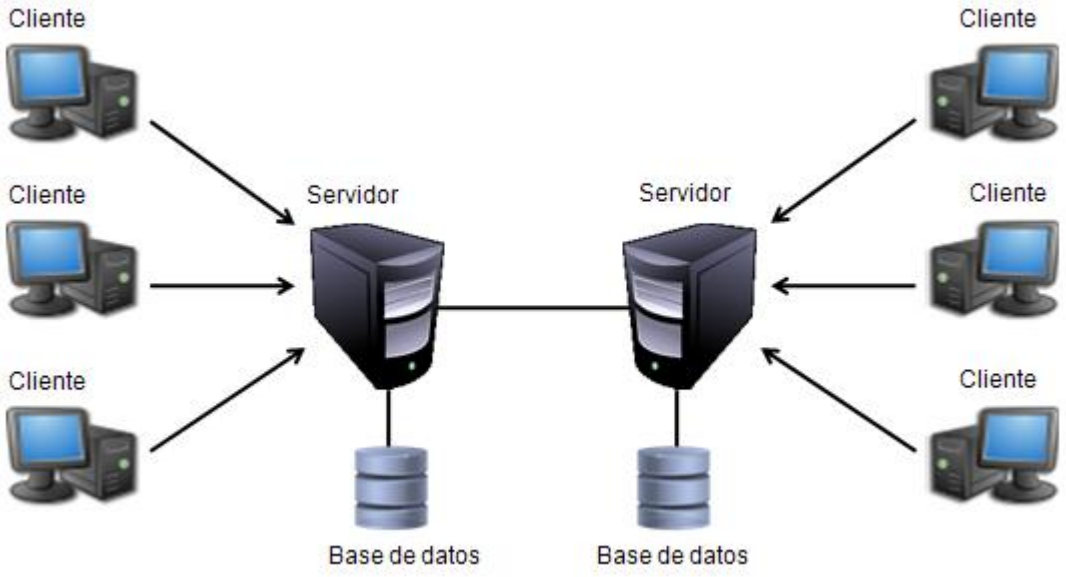


Figura 7. Arquitectura distribuida

1.7. Disponibilidad y réplica de datos

Actualmente la mayoría de las organizaciones demandan soluciones que generen tiempos rápidos de respuesta en las peticiones realizadas sobre los datos, por tal motivo ha surgido la necesidad de integrar un conjunto de ordenadores que operen como si fueran un único ordenador, denominado clúster, con el objetivo de garantizar disponibilidad, realizar réplica de datos, balancear las cargas entre los servidores y por consiguiente, elevar el rendimiento del sistema.

La disponibilidad está determinada por la capacidad de que varios servidores de bases de datos puedan trabajar en conjunto, permitiendo a un servidor secundario tomar rápidamente el control del sistema, ante algún fallo ocurrido en el servidor primario. Este procedimiento se encuentra definido en la documentación oficial de PostgreSQL como alta disponibilidad. (Group, 2008)

La replicación juega un papel fundamental en los sistemas de base de datos que se utilizan en aplicaciones de tiempo real. Es el proceso de copiar y mantener objetos de base de datos, como tablas, en múltiples bases de datos (ORACLE CORPORATION, 2002). Permite obtener mejoras en cuanto a disponibilidad de los datos, ya que la información tendrá al menos una copia como mínimo, que pueda ser usada para la recuperación del sistema ante un fallo ocurrido. Esta debe realizarse de forma tal, que garantice la consistencia de los mismos, en caso contrario, podría arrojar resultados erróneos, por tal motivo, los procesos de actualización deben realizarse en todos los nodos de bases de datos que contengan réplicas.

La réplica de datos puede realizarse de forma síncrona o asíncrona. La replicación síncrona garantiza la sincronización de los datos tanto en los sitios primarios como secundarios, en la que si se actualiza una réplica, las demás copias también serán modificadas dentro de la misma transacción; una transacción solo será aceptada cuando todos los nodos de bases de datos implicados estén listos para recibirla, en caso de que alguno falle, el proceso es anulado (ORACLE CORPORATION, 2002). Este proceso es transparente al usuario y desde el punto de vista lógico solo existe una versión de los datos, aunque es muy costoso en términos de latencia[17], ya que muchos mensajes circulan por la red. La replicación asíncrona, a diferencia de la anterior no permite una sincronización exacta de los datos porque la actualización de la información contenida en un nodo de bases de datos hacia los demás, no es realizada en la misma transacción, sino, pasado un período de tiempo, durante el cual es posible que los datos no sean idénticos.

La réplica de datos puede realizarse sobre los entornos de réplica maestro-esclavo o maestro-maestro, donde un servidor maestro (master server) es aquel en el que se pueden realizar consultas de lectura y escritura sobre los datos, mientras que en un servidor esclavo (slave server), se realizan consultas de solo

[17] Es la suma de retardos temporales dentro de una red.

lectura. En el entorno de réplica maestro-esclavo todas las modificaciones sobre los datos son enviadas al servidor maestro y este se encarga de enviar de forma asíncrona los cambios realizados hacia uno o más servidores esclavos. En cambio en el entorno maestro-maestro varios servidores se comunican entre sí con el objetivo de mantener la consistencia de los datos y la información puede replicarse de forma síncrona o asíncrona. En el primer caso, todos los servidores aceptan peticiones de escritura, pero las modificaciones de los datos son enviadas en una transacción desde el servidor original hacia los restantes. En el segundo caso los servidores trabajan de forma independiente y se comunican de forma periódica con otros servidores para resolver los conflictos de los datos. Ambos entornos de réplica permiten balancear la carga entre los servidores para evitar la sobrecarga en uno de ellos.

1.7.1. Soluciones de replicación en PostgreSQL

El gestor de bases de datos PostgreSQL hasta la versión 8.4 no permite realizar réplica automática de la información, pero si permite la implementación de mecanismos que garanticen la réplica de los datos, utilizando las técnicas de replicación maestro-esclavo y maestro-maestro, además brinda otras soluciones como:

- Shared Disk Failover o Almacenamiento compartido: este mecanismo permite tener un servidor actuando como maestro y varios servidores en estado de espera[18], donde todos en común comparten el mismo sistema de archivo o disco, todos los servidores están sincronizados y si el servidor principal falla, el servidor en estado de espera es capaz de iniciar rápidamente la base de datos sin pérdida de información, proceso conocido como failover. Tiene como limitaciones que si el disco falla o se corrompe, ambos servidores quedarían inhabilitados, además los servidores en estado de espera no pueden acceder al disco mientras el servidor maestro esté activo, lo que no permite balancear la carga entre ellos.
- Warm standby (Log shipping): esta forma de funcionamiento permite una alta disponibilidad, al contar con un servidor principal que envía de forma asíncrona los ficheros de WAL generados hacia el servidor en estado de espera que opera continuamente en modo de recuperación, así en caso de que el servidor principal fallase, el otro contendrá casi todos los datos del servidor activo y

[18] Un servidor en estado de espera (standby server) es aquel al cual no se tiene acceso hasta que su estado sea cambiado a servidor maestro.

podrá iniciarse como maestro. Este mecanismo no requiere de administración adicional y replica todo el servidor de base de datos. Tiene como limitaciones que en caso de fallas en el servidor principal, el servidor en estado de espera no contendrá las últimas transacciones realizadas y los cambios que se realicen a la base de datos que no son reflejados en los ficheros de WAL no serán replicados, como los índices hash. Además, el servidor en estado de espera no puede atender peticiones de los usuarios hasta que su estado no cambie a servidor maestro, es por este motivo que este mecanismo no permite balanceo de carga.

- Statement-Based Replication Middleware (Replicación basada en sentencias): su funcionamiento está basado en la utilización de un software intermediario que opera entre los clientes y los servidores de datos. El objetivo de esta herramienta es interceptar las sentencias SQL y enviarlas de forma síncrona hacia los servidores que conformen el clúster. Cada servidor trabaja de forma independiente, donde las sentencias de lectura-escritura son ejecutadas en todos ellos, mientras que las de solo lectura se ejecutan en único servidor, posibilitando balancear la carga entre los mismos. La utilización de funciones como now(), random(), tipos de datos como CURRENT_TIMESTAMP y la generación de secuencias puede ocasionar discrepancias entre los datos de los servidores.

1.8. Bases de datos y Sistemas de identificación

Actualmente se están desarrollando a nivel mundial varios Sistemas de Emisión de Documentos de Identificación, un ejemplo de ello lo constituye el SAIME, que surge como parte de las transformaciones llevadas a cabo por la República Bolivariana de Venezuela, con el fin de integrar los procesos de identificación, migración y extranjería con las más modernas tecnologías, logrando de esta manera la modernización de la Oficina Nacional de Identificación y Extranjería (ONIDEX). Se encuentra en funcionamiento desde marzo del año 2007. Para la persistencia de la información que manejan, utilizan un modelo de datos relacional, en el que emplean una estructura de tablas relacionadas, que permite separar los datos de los ciudadanos venezolanos, extranjeros, fallecidos y menores no cedulados, facilitando una gestión adecuada de la información de los ciudadanos. El sistema cuenta con una base de datos distribuida desplegada sobre el sistema gestor de bases de datos Oracle 10 g. Utilizan el modelo de diseño de bases de datos relacional para persistir la información que manejan. Existen en explotación

aproximadamente 83 servidores ubicados por todo el territorio venezolano y un clúster que centraliza toda la información que se almacena en estos. Dicha información es sincronizada a través de un servicio de réplica de datos.

Otro ejemplo de este tipo de sistemas lo constituye el Sistema de Identificación, Cédulas de Identidad y Pasaportes de la República de Chile, que tiene entre sus objetivos la verificación automática de la identidad de las personas y producir documentos de identidad que cumplan con los estándares internacionales y con un alto nivel de seguridad. Una de las principales innovaciones del proyecto fue la implementación de un sistema computacional central de identificación, que contiene una base de datos de identificación biométrica que almacena la fotografía, firma y huellas dactilares de la persona, sobre el sistema gestor de bases de datos Oracle 10 g. Para ello fue necesario digitalizar las fichas que contenían la información de los ciudadanos chilenos, a través del proceso de digitalización masiva. Esta base de datos contiene información que permite identificar a las personas e información referente a los antecedentes penales y delictivos. (GUERRA, 2005)

En Cuba, como parte de la política de informatización, el Ministerio del Interior (MININT), organismo encargado de preservar el orden interior y la seguridad del Estado, inicia la ejecución del proyecto "Identificación, Inmigración y Extranjería de la República de Cuba" de conjunto con la Universidad de las Ciencias Informáticas (UCI), como parte de un convenio de colaboración, donde uno de los objetivos a cumplir es *"Integrar las bases de datos que existen actualmente relativas a la identidad del ciudadano logrando un registro único de la población que garantice la seguridad en los datos"* (SANTIESTEBAN, y otros, 2010). En este proyecto se llevan a cabo procesos de identificación y control de los ciudadanos en el país. Muchos de estos procesos son de carácter secreto o confidencial y requieren una gran seguridad y protección. La arquitectura de bases de datos que utilizan es distribuida, donde cuentan con un Centro de Datos Nacional que intercambia información y se retroalimenta con bases de datos de oficinas. El principal mecanismo de intercambio de información es a través de réplica asíncrona con la herramienta Oracle Stream y la base de datos utiliza un modelo de diseño relacional. (SANTIESTEBAN, y otros, 2010)

Estos sistemas de identificación tienen en común que manejan información personal de los ciudadanos, por lo que un adecuado diseño para la persistencia de los datos de identidad, permitirá tener un mejor

control y organización de la información almacenada, además, utilizan el modelo relacional para la persistencia de los datos. El empleo de bases de datos para la gestión de información biométrica permite tener registrada este tipo de información, para su posterior utilización en procesos que requieran consultar estos datos.

1.9. Ambiente de desarrollo

En este apartado serán descritas las herramientas que permitirán el desarrollo del presente trabajo de diploma, las cuales están definidas por el proyecto “Transformación y modernización del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela (SEPYA)”.

1.9.1. Herramienta cliente para la administración del servidor de base de datos PostgreSQL

PgAdmin III es una herramienta libre publicada bajo la licencia de PostgreSQL, para el desarrollo y administración de bases de datos, mediante la cual se puede gestionar toda la información recogida en las bases de datos. Puede ser utilizado en Linux, FreeBSD, Solaris y Windows. Posee un editor de texto para sintaxis SQL con completamiento de código, sintaxis resaltada y depuración de errores. Permite la creación de reglas, índices, procedimientos almacenados, disparadores, secuencias, entre otros objetos. Brinda la posibilidad de realizar copias de respaldo a los datos y su restauración. Es desarrollado por la comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas.

1.9.2. Herramienta CASE

Dbwrench es una herramienta CASE multiplataforma, desarrollada en java sobre la plataforma Netbeans. Permite el diseño de bases de datos a través de diagramas entidad relación, en los cuales se pueden representar todo tipo de relaciones entre entidades, incluyendo relaciones recursivas. Presenta soporte para una amplia variedad de tipos de datos, dando la posibilidad de añadir nuevos tipos definidos por el usuario, además permite crear en el modelo, restricciones de unicidad, de anulabilidad, asignación de llaves primarias e índices a los atributos que compongan una relación. Brinda la posibilidad de realizar

ingeniería inversa de bases de datos, con lo cual se puede cargar un diseño mediante el establecimiento de una conexión sobre alguna base de datos. Otra de sus funcionalidades es que brinda la posibilidad de sincronizar la base de datos a partir de cambios aplicados sobre el modelo y viceversa. Presenta un editor de texto con completamiento de código y sintaxis resaltada que permite ejecutar sentencias SQL directamente en el servidor de datos. Esta herramienta posibilita crear un diccionario de datos en formato web que contiene todos los objetos que componen la base de datos como tablas, llaves primarias y foráneas, índices, secuencias, procedimientos almacenados, triggers y vistas con sus descripciones establecidas por el usuario. Dentro de los sistemas gestores de bases de datos soportados se encuentran MySQL, Microsoft SQL Server, Microsoft Access y PostgreSQL.

1.9.3. Lenguaje procedural PL/pgSQL

PL/pgSQL es un lenguaje procedural para el sistema de bases de datos PostgreSQL, el cual permite la creación de procedimientos almacenados y disparadores, hereda todos los tipos de datos definidos, así como funciones y operadores. Es un lenguaje sencillo de utilizar. Su utilización permite obtener un mayor rendimiento, debido a que con PL/pgSQL se pueden agrupar un grupo de sentencias SQL que son ejecutas en el servidor al mismo tiempo, de esta forma no se sobrecargaría la red. Este lenguaje añade a la potencia de un lenguaje procedural, la flexibilidad y sencillez del SQL, pudiéndose utilizar todos los tipos de datos, columnas, operadores y funciones de SQL. La portabilidad es otra de las grandes ventajas que ofrece, ya que las funciones se ejecutan dentro del servidor sin tener en cuenta la plataforma.

1.10. Propuesta de solución teórica

La persistencia de datos para los módulos de solicitud y enrolamiento del SEPYA se realizará en una base de datos relacional, permitiendo agilizar la generación de reportes y la actualización de los datos de una forma eficiente, consistente y rápida, en ella se utilizará además una herramienta para la replicación de la información, habilitando un balance de carga para aumentar la escalabilidad de la arquitectura de base de datos.

1.11. Conclusiones

- Se utilizará el modelo entidad relación para el diseño del modelo conceptual y el modelo relacional para el diseño lógico de la base de datos.
- La normalización adecuada de las entidades del modelo disminuirá la redundancia de información y por consiguiente evitará las anomalías en la actualización.
- La utilización de una herramienta para la replicación de los datos permitirá mejorar la disponibilidad de los mismos.

CAPÍTULO 2

DISEÑO DE LA BASE DE DATOS Y PROPUESTA DE SOLUCIÓN

Las características funcionales y técnicas del SEPYA a partir de sus procesos, son la clave para la identificación de los elementos arquitectónicamente significativos, cada uno de ellos son puntos vitales a tener en cuenta para el diseño de la bases de datos y para la elección de una herramienta que permita mejorar la disponibilidad de la información. El análisis detallado de un conjunto de herramientas disponibles, permitirá la caracterización de cada uno de ellas, para de esta forma identificar la más idónea y que cumpla con las necesidades del sistema. En este capítulo se incluyen los patrones de diseño utilizados, y se generan los entregables Diccionario de datos y el Modelo de Datos del sistema.

2.1. Características funcionales y técnicas del sistema

El análisis de las características funcionales y técnicas del sistema es un elemento clave a tener en cuenta en la investigación, a partir de ellas se deben identificar las principales funcionalidades, características técnicas y definir sus elementos arquitectónicamente significativos. Este paso es crucial para la identificación, modificación o implementación de una herramienta, que permita resolver las necesidades primarias del sistema y que cumpla con las expectativas planteadas.

2.1.1. Descripción de los procesos del sistema

Los procesos relacionados con la solicitud, el enrolamiento de datos y personalización de los documentos de pasaportes diplomáticos, de servicio y acreditaciones del SEPYA, están encaminados a mejorar el mecanismo que se lleva a cabo actualmente en Venezuela para la emisión de estos documentos. Ellos se encuentran descritos de forma general de acuerdo a las especificaciones legales definidas para los documentos de identificación, con las particularidades especificadas por el país al cual se encuentran sujetos, teniendo en cuenta normas internacionales como la Organización de la Aeronáutica Civil Internacional (OACI)[19]. Su identificación se realizó a partir del levantamiento de requisitos y de las experiencias en la implantación del sistema SAIME, además, se tuvo en cuenta la investigación "Propuesta de procesos para el desarrollo de un SEDI", encargada de realizar una definición y descripción detallada de los procesos de negocio, con el objetivo de garantizar correctamente los pasos generales dentro del flujo de emisión de un documento.

La emisión de documentos de identificación como pasaportes diplomáticos y acreditaciones, requiere de tres pasos fundamentales: solicitud, captura de datos y personalización del documento. En el sistema de pasaportes diplomáticos, de servicio y acreditaciones, se requiere inicialmente de una solicitud online donde se registrarán un conjunto de datos personales del titular que puede ser tanto venezolano como extranjero, y los datos de la solicitud[20]. En el caso de pasaporte se pueden realizar solicitudes para funcionarios y para familiares de funcionarios, mientras que en el caso de las acreditaciones las

[19] Organización que dictamina las regulaciones para los documentos de viaje.
[20] Los datos de la solicitud incluyen tipo de solicitud (emisión, renovación), tipo de documento, motivo de la solicitud, entre otros.

solicitudes pueden ser para funcionarios, familiares de funcionarios y personal de servicio doméstico. Posteriormente se procede a la verificación de la información recogida, y para aquellas solicitudes donde el ciudadano titular sea cedulado, se procede a realizar una validación de sus datos a través del SAIME. Seguidamente se verifica si el titular tiene algún tipo de prohibiciones[21] u objeciones a partir de las cuales se pueda denegar la emisión del documento de identificación. Una vez realizada la validación, si los datos son correctos se aprueba la solicitud y se envía una notificación al ciudadano solicitante para que este reserve una cita, en caso contrario se deniega la solicitud y se envía una notificación explicando las causas.

El segundo paso es el enrolamiento de datos, este comienza con la reservación de la cita para que el ciudadano se presente en el área correspondiente para la captura de los datos del trámite, entre ellos se encuentra el registro de un conjunto de documentos de recaudo[22] y su información biométrica[23], la que será utilizada para la confección del documento a emitir. Para los trámites de pasaporte la captura de los documentos de recaudo es obligatoria, en cambio, en el caso de acreditaciones es opcional. Posteriormente se realiza una supervisión de la información captada, donde ante la ocurrencia de algún error se realiza un proceso de corrección de la misma. Una vez revisada la información capturada, y después de haber comprobado que no existe ningún problema, se desarrolla un proceso de validación de identidad con el Sistema Automático de Identificación de Huellas Dactilares (de sus siglas en inglés Automatic Fingerprint Identification System, AFIS) para aquellos ciudadanos cedulados mediante sus huellas dactilares, asegurando por esta vía que la persona es quién dice ser. Si la respuesta emitida por AFIS es negativa, se detiene el trámite hasta que se corrija el error identificado o se cancela.

El tercer paso es la personalización de los documentos, que recibe un conjunto de datos de entrada para iniciar el proceso. Posteriormente, estos datos sufren un conjunto de transformaciones necesarias para su impresión y más tarde el documento impreso pasa por un proceso de control de calidad. En caso de que el control de calidad genere algún error, el documento vuelve a ser personalizado, en caso contrario se procede al ensobrado y entrega del documento.

[21] Restricciones de extranjería relacionadas con la entrada y salida al país.
[22] Documentos legales utilizados para validar la información recogida.
[23] La información biométrica incluye las huellas, foto y firma necesaria para la emisión del documento de identificación.

2.1.2. Necesidades técnicas del sistema

La identificación de los elementos técnico-funcionales para la arquitectura de base de datos del SEPYA tuvo lugar a partir de la descripción de los procesos del sistema y a partir de los requerimientos no funcionales del software. Para la solicitud de los documentos y consulta del estado del trámite se dispondrá de un acceso desde el portal del MPPRE y para la ejecución del trámite se contará con una aplicación de escritorio desplegada en el local de Captación y supervisión de documentos. (Requisitos, 2010). La emisión de de este tipo de documentos requiere un alto nivel de seguridad y control, de ahí que la información que se maneje debe ser protegida contra accesos no autorizados y de esta forma garantizar la confidencialidad e integridad de la misma, además el acceso a los datos debe realizarse a través de comunicaciones seguras utilizando el protocolo Secure Socket Layer (SSL)[24].

En ocasiones se realizan trámites a funcionarios diplomáticos con carácter urgente, lo que requiere que el sistema permanezca disponible de lunes a viernes, en el horario comprendido desde las 0:00h del lunes hasta 23:59h del viernes. Se contará con dos servidores de base de datos sobre el SGBD PostgreSQL, pues los clientes solicitan que se utilice un gestor de bases de datos libre, además, producto de la disponibilidad que debe presentar el sistema, es necesario mantener una réplica de los datos para garantizar que ante fallas eventuales en una de las instancias de PostgreSQL, el otro continúe aceptando peticiones sin que el servicio se vea afectado y con la mínima pérdida de información. El menor tiempo promedio establecido dentro del sistema para la recuperación ante errores es de 4:00h (Requisitos, 2010), además, la lógica de negocio del sistema se automatizará con una herramienta para la administración de procesos de negocio (de sus siglas en inglés Business Process Management System, BPMS) y se utilizará un componente adicional para la gestión de los datos biométricos de los ciudadanos. (Requisitos, 2010), ambas herramientas requieren de una estructura persistente propia para su funcionamiento.

Partiendo de los requerimientos identificados se definen un grupo de elementos arquitectónicamente significativos a tener en cuenta en la solución:

- La solución requiere la utilización de un ambiente de despliegue centralizado.

[24] Secure Socket Layer es un protocolo que establece un canal de comunicación segura, cifrando el intercambio de datos entre dos equipos.

- La información deberá ser replicada con el menor retardo posible, para garantizar la continuidad del servicio ante fallas en uno de los servidores.
- Se requiere contar con un mecanismo de recuperación que permita restaurar el servidor caído y llevarlo a un estado consistente en un tiempo medio de reparación de 4:00h.
- Se emplearán dos bases de datos, una para la información de negocio del sistema, que en su composición tendrá un esquema utilizado por el componente de gestión de la información biométrica de los ciudadanos. La segunda base de datos permitirá almacenar la información generada por los procesos de negocio.

Un elemento adicional dentro de la arquitectura de base de datos será balancear la carga del sistema SEPYA entre los servidores de datos, permitiendo que ante el aumento de solicitudes a la BD, la arquitectura se mantenga escalable, además, es clave la correcta identificación de una herramienta que garantice la replicación de base de datos para PostgreSQL, por lo que es necesario evaluar diferentes soluciones que permitan satisfacer esta expectativa.

2.2. Herramientas de replicación

Existen herramientas capaces de garantizar la replicación de datos, algunas más complejas y con mayores funcionalidades que otras. La selección para el uso de alguna de ellas depende de las necesidades y restricciones que se tengan, es por esto que en la investigación se analizaron diferentes herramientas como Slony-I, Bucardo, CyberCluster, PgCluster y Pgpool-II.

2.2.1. Slony-I

Es un sistema de replicación asíncrono maestro-esclavo, implementado bajo licencia BSD, el cual soporta la replicación en cascada, dando la posibilidad que un servidor esclavo sea a su vez un servidor maestro para otros esclavos y puede ser utilizado en un ambiente donde existan diferentes versiones de PostgreSQL, sistema operativo y hardware. Permite replicar grandes bases de datos hacia un número limitado de servidores esclavos, cuya cantidad no debería sobrepasar los 12 ordenadores, pues una cantidad mayor aumentaría los costos de comunicación y disminuirían las ventajas de tener varios servidores en el clúster. Los datos no son replicados a los esclavos instantáneamente, pues los cambios

realizados sobre los mismos son almacenados en una tabla mediante disparadores o triggers, y un servicio es el encargado de enviar esos cambios al servidor donde se van a replicar, donde son leídos y ejecutados. Este método de replicación tiene como inconveniente que las últimas transacciones pueden verse comprometidas ante fallas ocurridas en el servidor maestro, debido a que este puede estallar y los cambios no ser replicados hacia los servidores esclavos.

Slony-I no posee funciones propias para la detección de errores ocurridos en un nodo de forma automática, de ahí que no permita realizar failover automático, aunque sí de forma manual, además, no permite balancear la carga entre los servidores, ni brinda un pool de conexiones[25]. Otra de sus limitantes es que no permite replicar sentencias de definición de datos y objetos lob[26] pues ni las sentencias de definición de datos, ni las operaciones sobre objetos lob, ni las sentencias TRUNCATE son capaces de tener disparadores adecuados para informar a Slony-I cuando este tipo de cambios tienen lugar, pues PostgreSQL por motivos de seguridad, no da la posibilidad de crear disparadores sobre las tablas del sistema. Como resultado los únicos objetos de base de datos que Slony-I puede replicar son las tablas y las secuencias, teniendo en cuenta que cada tabla de la base de datos debe tener una llave primaria para poder ser replicada.

2.2.2. Bucardo

Bucardo es un sistema de replicación asíncrono que puede ser usado sobre los entornos de réplica maestro-esclavo y maestro-maestro. Se encuentra desarrollado en el lenguaje de programación Perl bajo licencia BSD y su funcionamiento hace un uso intensivo de triggers y los lenguajes procedurales PL/PgSQL y PL/Perl. Para los entornos de replicación maestro-esclavo, la réplica se realiza en cascada, permitiendo a un servidor esclavo ser un servidor maestro para otros esclavos, con lo que se obtiene una alta escalabilidad y permite equilibrar las cargas entre los servidores; mientras que la replicación para los entornos maestro-maestro se realiza exclusivamente con dos servidores maestros, permitiendo alcanzar

[25] Un pool de conexiones o connection pooling, es un mecanismo que permite manejar un grupo de conexiones abiertas a la base de datos para que sean reutilizadas ante cualquier operación de consulta y modificación, permitiendo que existan menos conexiones físicas a la base de datos que clientes ejecutando la aplicación

[26] Objetos que almacenan valores cuyo tamaño excede el tamaño máximo de almacenamiento definido para los tipos de datos estándar.

una alta disponibilidad y realizar failover ante la caída en uno de los nodos, aunque estos procesos no se realizan de forma automática.

Como limitantes se encuentra que no realiza balanceo de carga entre los servidores de datos, y al igual que Slony-I, no permite la replicación de sentencias de definición de datos, ni objetos lob y no replica aquellas tablas de la base de datos que no posean llave primaria. Su funcionamiento requiere tener instalado PostgreSQL a partir de su versión 8.0 con los lenguajes procedurales PL/pgSQL y PL/Perl.

2.2.3. Cybercluster

Es una herramienta de replicación síncrona para los entornos de replicación maestro-maestro donde pueden intervenir más de 2 servidores maestros, desarrollada bajo licencia BSD. Las versiones 1.0 y 2.0 se encuentran basadas en las versiones de PostgreSQL 8.2 y 9.0 respectivamente. Dentro de las funciones que brinda, se encuentran el balanceo de carga, lo que permite distribuir las operaciones de lectura entre los diferentes servidores de datos y ante la ocurrencia de algún error en uno de los servidores, Cybercluster separa dicho nodo automáticamente del clúster, sin interrumpir el acceso a los datos, ya que se puede continuar tramitando peticiones con los servidores restantes.

La recuperación de un nodo caído se realiza a partir de un servidor maestro activo y una vez recuperado puede volver a formar parte del clúster. Esta operación no requiere de una interrupción de los servicios, si al menos se cuenta con 3 servidores de bases de datos, debido a que debe existir un nodo activo y uno que se encargue de realizar las sincronizaciones necesarias hacia el servidor caído. La replicación no presenta problemas con la utilización de secuencias, procedimientos almacenados, objetos lob y funciones volátiles[27] como now(), nextval() y setval(). La replicación de objetos lob tiene como restricción, que estos deben ser ubicados en un directorio que pueda ser leído por todos los nodos de base de datos. El funcionamiento de Cybercluster está determinado por los nodos que integran el clúster, compuesto por balanceadores de carga, servidores de replicación y servidores de datos. Cuando no se cuenta con balanceadores de carga, las aplicaciones pueden conectarse a cualquier nodo de la BD dentro del sistema. El balanceador de carga no está implicado directamente dentro del proceso de replicación ya que

[27] "Indica que la función puede devolver diferentes valores, incluso dentro de una consulta individual de una tabla" (MARTÍNEZ, 2009)

su funcionalidad es distribuir la carga, por tal razón, si no está presente, no se podrá obtener un mejor rendimiento de los servidores. Además, esta herramienta de replicación es lenta a la hora de distribuir las peticiones y su funcionamiento requiere modificar el código fuente de PostgreSQL, motivo por el cual se encontrará por detrás de la última versión del SGBD.

2.2.4. PgCluster

PgCluster es una herramienta de replicación síncrona para entornos de replicación maestro-maestro desarrollado bajo licencia BSD, donde varios servidores de datos pueden recibir peticiones de forma simultánea. Su composición está determinada por 3 tipos de servidores, servidores para balanceo de carga, servidores de replicación y servidores de datos. Los servidores de balanceo de carga se utilizan para distribuir la carga entre los servidores de datos y permitir que la replicación se realice lo más rápido posible. Esta herramienta posee un mecanismo de detección de errores, cuyo objetivo es separar del clúster aquel servidor que tenga problemas de conexión y continuar con el mecanismo de replicación sin que los restantes servidores se vean afectados. En caso de que el servidor de replicación falle, los servidores de datos entran en un modo llamado stand-alone en el cual el acceso se puede realizar de 2 vías, la primera permite realizar consultas de solo lectura, mientras que en la segunda se pueden realizar consultas de lectura y escritura, permitiendo la actualización de las bases de datos. La replicación se realiza utilizando los archivos que son manejados por el servidor de base de datos a través del comando rsync[28], lo que tiene como desventaja que como la replicación es síncrona, si fallase la conexión, pueden ocurrir errores en la transferencia de archivos.

Una completa implementación requiere como mínimo de 6 máquinas, distribuidas en 3 servidores de datos, 2 para balanceo de carga y una para el control de monitorización y failover. Además su funcionamiento requiere de modificaciones al código fuente de PostgreSQL, no es un sistema escalable y los procesos de instalación y configuración son muy complejos. Todo el desarrollo de PgCluster se ha detenido pasando a PgCluster2 debido a los problemas arquitectónicos que este presenta, motivo por el cual no está completamente maduro, aunque continúa siendo utilizado en producción por unos pocos sitios.

[28] Herramienta de transferencia incremental de archivos para sistemas UNIX.

2.2.5. Pgpool-II

Es un software intermediario que opera entre las aplicaciones clientes (frontends) y los procesos de postgres (backends) manejando las comunicaciones entre ellos, de forma tal que las aplicaciones clientes ven a Pgpool-II como si este fuese el servidor de base de datos, y los servidores de datos ven a Pgpool-II como uno de sus clientes. Se encuentra desarrollado en lenguaje C, bajo licencia BSD y funciona sobre Linux, Solaris, FreeBSD, la mayoría de las arquitecturas UNIX, mientras que para Windows no tiene soporte. Puede ser utilizado a partir de la versión 6.4 de PostgreSQL.

El aumento de conexiones concurrentes en un sistema trae consigo un incremento en el consumo de recursos y un impacto negativo en el rendimiento, por tal motivo, Pgpool-II admite un límite máximo de conexiones, y las conexiones extras se mantendrán en una cola, en lugar de devolver un error inmediatamente. También brinda un pool de conexiones con el objetivo de reducir la sobrecarga en las conexiones y mejorar la productividad global del sistema, ya que mantiene abiertas las conexiones a los servidores PostgreSQL y las reutiliza siempre que se solicite una nueva conexión con las mismas propiedades que una ya existente.

Pgpool-II puede manejar múltiples servidores y la replicación de los datos se realiza síncronamente. También brinda la posibilidad de continuar brindando servicio ante la caída de algún servidor. Permite realizar la recuperación y sincronización del servidor caído, así como la incorporación de nuevos nodos al clúster. Como la replicación es síncrona, la ejecución de una consulta de lectura en cualquiera de los nodos que integren el clúster arrojará el mismo resultado, y Pgpool-II se aprovecha de esta ventaja para distribuir las consultas de lectura entre los nodos, permitiendo balancear la carga y por consiguiente reducir el trabajo de los servidores. El balance de carga que realiza Pgpool-II está orientado a la sesión que se establece con uno de los nodos de datos, y no basado en las sentencias SQL que son enviadas a los servidores, lo que significa que la selección de un nodo para el balance de carga se realiza cuando se inicia una sesión, y todas las sentencias SQL serán enviadas el servidor seleccionado, hasta que la sesión abierta se haya cerrado. La selección del nodo se realiza aleatoriamente considerando el parámetro de configuración backend_weight, definido en el fichero de configuración de Pgpool-II para cada uno de los nodos de base de datos que son manejados por el middleware. De este modo la mayor carga de lectura

puede ser asignada a uno de los servidores que integren el clúster. El rendimiento mejorará proporcionalmente al número de servidores, y el balanceo de carga funcionará mejor para aquellas situaciones en las que exista un gran número de usuarios ejecutando consultas de forma concurrente. A partir de su versión 2.3.2 permite la replicación de objetos lob sobre la versión 8.1 de PostgreSQL o superiores.

Otra ventaja es que ofrece una interfaz para realizar algunas tareas administrativas a través de la red, como la recuperación de un nodo caído, conocer el estado en que se encuentran los nodos que conforman el clúster, así como añadir o eliminar nodos al clúster. Como desventajas presenta que al ser un software intermediario pueden existir problemas durante la utilización de funciones como now(), random(), tipos de datos como CURRENT_TIMESTAMP lo que puede ocasionar discrepancias entre los datos de los servidores.

2.3. Elección de la herramienta

Para la selección de la herramienta se tuvieron en cuenta algunos criterios principales, como tipo de replicación, failover automático, si posee un mecanismo de recuperación, replicación de objetos lob, número de servidores necesarios, además de otros criterios adicionales que complementan la solución, como balanceo de carga, modificación adicional al código fuente de PostgreSQL, pool de conexiones, calidad de la documentación disponible y respaldo de comunidades activas.

Como consecuencia de la disponibilidad que debe ofrecer el sistema, es necesario garantizar una replicación de la información entre los servidores de datos, de ahí la necesidad de contar con una herramienta que permita una sincronía de los datos en todo momento, para garantizar que ante el fallo en uno de los servidores, el otro pueda continuar brindando servicio de forma automática, sin pérdida de información; para lo cual debe existir un mecanismo que posibilite restaurar el nodo caído y llevarlo a un estado consistente. Otro elemento a tener en cuenta para la elección, es la replicación de objetos lob, esto se debe a la necesidad de incorporar un modelo persistente (modelo de procesos) que en su definición posee este tipo de objetos. Otro aspecto a tener en cuenta, es el número de servidores que cada

herramienta utiliza como mínimo para garantizar la réplica de datos, debido a que la solución debe ser desplegada sobre dos servidores físicos como parte de la propuesta de la solución técnica.

Adicionalmente se propone balancear la carga entre los servidores que conforman el clúster y utilizar un agrupamiento de conexiones o un pool de conexiones con el objetivo de obtener un mejor rendimiento y por consiguiente mejores tiempos de respuesta. Se tuvo en cuenta además, la calidad de la documentación disponible, pues es necesario contar con la documentación de las herramientas con las que se trabaja, y debido a que dichas herramientas son open source, en caso de que surjan errores, resulta importante contar con el respaldo de comunidades activas que puedan proveer soporte para preguntas y su solución. Otro elemento que se tuvo en cuenta fue la necesidad de algunas herramientas de modificar el código fuente de PostgreSQL para garantizar su funcionamiento, lo que las hace dependientes de versiones específicas, imposibilitando utilizar la última versión del gestor.

En la siguiente tabla se evalúan las características de las herramientas estudiadas respecto a los criterios de selección, se asigna (++) al requerimiento en caso de que cumpla con él ampliamente, (+/-) para los casos en que se cumple parcialmente y (- -) en los casos en los que no se cumple.

Criterio de selección	Slony-I	Bucardo	Cybercluster	PgCluster	PgPool-II
Replicación síncrona	--	--	++	++	++
Replicación asíncrona	++	++	--	--	--
Failover automático	--	--	++	++	++
Recuperación	--	--	++	++	++
Replicación de objetos lob	--	--	++	++	++
Por debajo de 3 servidores	++	++	--	--	++
Balanceo de carga	--	--	++	++	++
Pool de conexiones	--	--	--	--	++
Modificación adicional	--	--	++	++	--
Calidad de la documentación disponible	++	++	--	--	++
Respaldo de comunidades activas	++	++	--	--	++

Tabla 2 Comparación de herramientas

Con el análisis realizado de las herramientas se identificó que Slony-I y Bucardo contaban con buena documentación y pueden ser utilizadas con el número mínimo de servidores necesarios para implantar la

solución, pero no cumplen con los requerimientos necesarios y primordiales del sistema, en primer lugar realizan réplica asíncrona lo que puede provocar inconsistencia entre los servidores y ante fallas en el servidor maestro las últimas transacciones pueden verse comprometidas, además, no brindan la posibilidad de realizar failover automático, no permiten la replicación de objetos lob y adicionalmente no dan la posibilidad de balancear la carga entre los servidores y no brindan un pool de conexiones.

Tanto Cybercluster, PgCluster y Pgpool-II replican la información de forma síncrona, además permiten realizar failover automático, balancear la carga y brindan mecanismos para la recuperación de un nodo caído. Las diferencias están dadas por el número de servidores que necesitan para implantar la solución. En el caso de Cybercluster un despliegue requiere como mínimo de 3 servidores, dos servidores de datos y un servidor de replicación cuya funcionalidad es garantizar la réplica entre los servidores de datos; mientras que la creación de un clúster con PgCluster de acuerdo a (PostgreSQL Web Team, 2010), requiere como mínimo de 6 servidores, distribuidos en 3 servidores para base de datos, 2 para balance de carga y un servidor para monitorización y control de failover, aunque el sitio oficial de PgCluster expone que pueden ser utilizados 5 servidores sin tener en cuenta el servidor de monitorización. A partir de la versión 1.0.6cv4 es posible instalar el servidor de replicación unido a un servidor de datos, aunque para aquellos sistemas que necesiten balancear la carga y ofrecer una alta disponibilidad, esta opción puede resultar peligrosa, por lo que es recomendable tener el servidor de replicación de manera independiente respecto a los servidores de datos. A diferencia de Cybercluster y PgCluster, la creación de un clúster a través Pgpool-II requiere como mínimo de 2 servidores, motivo por el que se ajusta al número de servidores con que se cuenta.

Tanto Cybercluster como PgCluster modifican el código fuente de PostgreSQL para su funcionamiento, la calidad de la documentación disponible para ambas herramientas no es buena y en el caso de Cybercluster es escasa, además no cuentan con un eficiente respaldo de comunidades activas.

El resultado final de la selección se inclina hacia Pgpool-II como la herramienta necesaria para implantar la solución, pues, al los datos replicarse síncronamente, se garantiza que la información sea insertada, modificada o eliminada en ambos servidores al mismo tiempo, dando la posibilidad de continuar brindando servicio ante la caída de uno de los servidores sin que haya pérdida de datos, además, Pgpool-II ofrece un

método para la recuperación de un nodo caído, a través del cual el nodo puede ser restaurado y puesto en marcha nuevamente. También otra de sus ventajas es que permite la replicación de objetos lob, de esta forma se garantiza que la base de datos utilizada por la herramienta para la administración de procesos de negocio, sea replicada correctamente. Adicionalmente, ofrece otras funcionalidades que contribuyen a obtener un mejor rendimiento y una mejor gestión pues brinda un pool de conexiones, permite balancear la carga entre los servidores que integran el clúster, ofrece posibilidades de ser administrado remotamente y además posee buena documentación y un buen respaldo por parte de su comunidad.

2.4. Diseño de la base de datos

El diseño de la base de datos inicia a partir de la identificación de los conceptos de negocio más significativos en el dominio del problema, determinándose así las entidades, atributos y restricciones que son necesarios para modelar la base de datos. Posteriormente se normaliza el modelo llevándolo a una de las formas normales, según las necesidades del sistema y finalmente se realiza el diseño físico de la base de datos a partir del modelo construido.

2.4.1. Conceptos de negocio significativos

Para garantizar un buen diseño de la base de datos, es necesario identificar los principales conceptos de negocio, para ello se realizó el Modelo de dominio, cuyo objetivo es comprender y describir las clases más importantes dentro del contexto del sistema (JACOBSON, y otros, 2000). La **figura 8** muestra una representación conceptual del negocio, a través de los conceptos más relevantes y las relaciones entre ellos. La relación entre organismo y solicitante, significa que un organismo contiene un conjunto de ciudadanos solicitantes, encargados de realizar las solicitudes de pasaporte o de acreditación para un titular. Cada solicitud de pasaporte o de acreditación, si es aprobada, tiene asociada un trámite de pasaporte o de acreditación, donde se recogen un conjunto de documentos de recaudo[29] necesarios para iniciar el trámite, además, se capturan un conjunto de datos biométricos del titular, que incluyen fotografía, firma y huellas, utilizados para validar la identidad del mismo a través de un trámite con el sistema AFIS, y

[29] Partida de nacimiento, Acta de fe de soltería, Autorización LOPNA, Certificado de divorcio, Certificado de matrimonio, Certificado de adopción, Constancia de estudio, Certificado médico de capacidad diferente.

para la confección del documento de identificación a expedir. Una vez que el trámite haya sido revisado, se ejecuta una orden de impresión que contiene los datos necesarios para personalizar el documento.

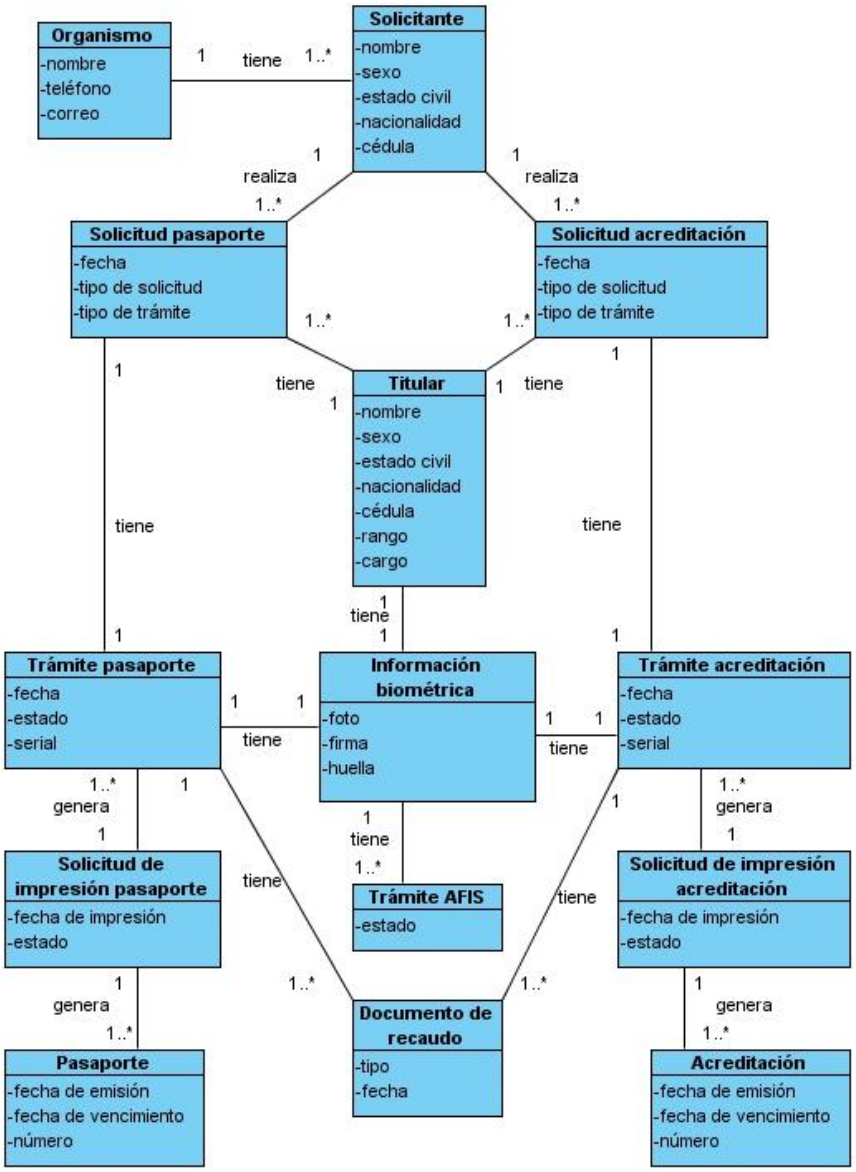


Figura 8. Modelo de dominio

2.4.2. Estructura general de la base de datos

La base de datos está compuesta por esquemas, que contienen un conjunto de tablas relacionadas entre sí, encargadas de almacenar toda la información que es gestionada durante los subprocesos de solicitud y enrolamiento de datos. Debido a la extensión del modelo, y para su mejor comprensión, se encuentra dividido en agrupaciones lógicas o sub-modelos, de acuerdo a ciertos criterios que se encuentran asociados en su mayoría con los procesos de negocio.

2.4.2.1. Vista por esquemas

La base de datos está compuesta por dos esquemas como se muestra en la **figura 9**. En el esquema público se almacena toda la información relacionada con las solicitudes y los trámites de pasaporte y acreditación, los datos referentes a la administración del sistema como tipo de organización, organismos y sus responsables, usuarios, roles, permisos de usuario según los roles y las funcionalidades que son asignadas a los roles. El esquema de biometría es utilizado para procesar toda la información biométrica de los ciudadanos que son tramitados y se presenta como parte de la propuesta de arquitectura del sistema.

Figura 9. Esquemas de la base de datos

2.4.2.2. Vista por sub-modelos

Esta vista representa la agrupación lógica de entidades, según definiciones del negocio, para una mejor comprensión del modelo. En la **figura 9** se muestran los sub-modelos presentes en cada uno de los esquemas existentes. El esquema público está compuesto por los sub-modelos solicitud, dirección, administración, ciudadano y trámite, mientras que el esquema biometría está formado por el sub-modelo biometría.

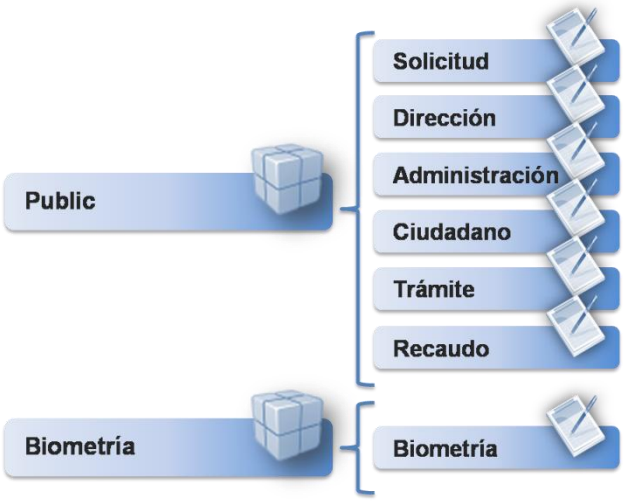


Figura 10. Vista por sub-modelos

Sub-modelo Solicitud

En el sub-modelo Solicitud se almacena toda la información necesaria para la solicitud de un pasaporte o de una acreditación, como datos personales del ciudadano, datos de contacto, si tiene o no prohibiciones y los datos propios de la solicitud, que incluyen tipo de solicitud, tipo de trámite y para el caso de solicitudes de acreditación, se recogen los datos del pasaporte y si el titular es extranjero se registran además sus datos laborales y los datos de la visa en caso de que posea. Inicialmente la información es almacenada en tablas estado, cuyo objetivo es evitar que los datos introducidos durante la solicitud se mezclen con los datos ya validados, pues esta pasa un proceso de revisión, con el objetivo de verificar que no existan errores en la información recogida. De esta forma se evita además, que se acceda directamente a la información de los ciudadanos. En este sub-modelo se encuentran también las entidades encargadas de la persistencia de las notificaciones que son enviadas a los ciudadanos una vez validada y aprobada la solicitud. El modelo de datos de este sub-modelo se muestra en el **anexo 2**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Dirección

En este sub-modelo se almacenan las direcciones de residencia y lugares de nacimiento de los ciudadanos, además de países, ciudades, estados y municipios, utilizados por diferentes entidades del modelo en general. El modelo de datos de este sub-modelo se muestra en el **anexo 3**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Administración

Este sub-modelo incluye las entidades para la gestión de usuarios, roles, permisos de usuarios, funcionalidades asociadas a los roles, los nomencladores del sistema, las organizaciones que realizan las solicitudes con sus responsables y los tipos de organizaciones. Contiene además diferentes entidades que permiten la planificación de las citas, las que son reservadas por los usuarios para iniciar los procesos de captura de datos e información biométrica una vez que la solicitud es aprobada. El modelo de datos de este sub-modelo se muestra en el **anexo 4**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Ciudadano

En este sub-modelo se registra la información de los ciudadanos, una vez que los datos introducidos durante la solicitud hayan sido validados. Está compuesto por un conjunto de entidades cuya finalidad es separar los ciudadanos venezolanos cedulados de los no cedulados, los extranjeros y los fallecidos, con el objetivo de lograr una mayor organización y control de los datos de los ciudadanos que son tramitados. El modelo de datos de este sub-modelo se muestra en el **anexo 5**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Trámite

Este sub-modelo almacena la información referente a los trámites de pasaporte y acreditación, registrándose cada una de las fases por las cuales transita un trámite. Además en él se registran los acompañantes de viaje y familiares de los funcionarios. El modelo de datos de este sub-modelo se

muestra en el **anexo 6**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Recaudo

Este sub-modelo incluye los recaudos necesarios para iniciar el trámite, como partida de nacimiento, certificado de acta de matrimonio, acta de defunción, autorización LOPNNA[30] , certificado de adopción, entre otros. El modelo de datos de este sub-modelo se muestra en el **anexo 7**. Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

Sub-modelo Biometría

Este sub-modelo se encuentra dentro del esquema biometría, y es utilizado por un componente cuya finalidad es la gestión de la información biométrica de los ciudadanos, además, tiene la responsabilidad de llevar a cabo el proceso de validación de identidad contra el sistema AFIS mediante las huellas dactilares. La representación del modelo se muestra en el **anexo 8.** Las descripciones de las entidades y sus atributos se pueden encontrar en el documento Diccionario de datos, generado como parte de los entregables del presente trabajo.

2.4.3. Nomenclatura

En el diseño de la base de datos se utilizó una nomenclatura sencilla que permite una mejor comprensión y organización de los elementos que componen el modelo. Los nombres utilizados para identificar los elementos de la base de datos como tablas, dominios, índices, y triggers, están formados por un prefijo seguido de un criterio de agrupación. Las palabras que lo componen se separan empleando el carácter guión bajo ‘’_”. Todos los elementos del modelo se encuentran en letra minúscula. Los prefijos se encuentran descritos en la siguiente tabla:

[30] Ley Orgánica Para Niñas, Niños y Adolescentes.

Prefijo	Elemento
tbl	Tablas
tbl_n	Tablas nomencladoras
pk	Llaves primarias
fk	Llaves foráneas
ncl	Llaves foráneas nomencladoras
dom	Dominios
seq	Secuencias
fun	Procedimientos almacenados
trg	Triggers
idx	Índices

Tabla 3. Prefijos del modelo de datos

Los nombres de las tablas que almacenan información propia de la solicitud comienzan con *tbl_solicitud*. Las tablas que almacenan la información de solicitud del ciudadano comienzan con *tbl_estado*. En el caso de las tablas de direcciones, estas comienzan con *tbl_direccion* y las que guardan la información asociada al trámite, inician con *tbl_tramite*, excepto las que almacenan datos asociados a los documentos de recaudo, que inician con *tbl_recaudo*. Las asociadas al registro de usuarios, roles y funcionalidades comienzan con *tbl_acceso* y las que registran información histórica terminan con *hist*.

Los nombres de las llaves primarias están compuestos por el prefijo *pk* seguido del nombre de la tabla sin el prefijo de esta última. Los nombres de las llaves foráneas están compuestos por el prefijo *fk* seguido del nombre de la tabla foránea, excluyendo las referencias de las tablas nomencladoras que inician con el prefijo *ncl* seguido del nombre del nomenclador. Para aquellas tablas que posean más de una llave foránea procedentes de una misma tabla, tendrán al final un nombre que las identifique en su contexto de negocio; ejemplo de esto son las nacionalidades de los ciudadanos que pueden ser nacionalidad originaria y nacionalidad adquirida. En este caso las entidades que almacenan los datos personales de los ciudadanos poseen dos llaves foráneas de la tabla *tbl_direccion_pais,* donde las llaves foráneas poseen los nombres *fk_direccion_pais_nacionalidad_originaria y fk_direccion_pais_nacionalidad_adquirida cada una.*

Los procedimientos almacenados se encuentran nombrados acorde a la funcionalidad que realizan. Aquellos procedimientos asociados con las solicitudes de pasaporte y de acreditación comienzan con

fun_sp y *fun_sa* respectivamente, en cambio los relacionados con el trámite inician con *fun_tp* y *fun_ta*. Los vinculados con el sub-modelo dirección inician con *fun_direccion* y aquellos procedimientos comunes al proceso de solicitud y de trámite, inician con *fun_comun*. De acuerdo al tipo de retorno, los que culminan con la palabra total retornan un valor entero, los de tipo obtener, devuelven un único elemento y los de tipo listar un conjunto de tuplas.

2.4.4. Indexación

Cuando se desea obtener una fila, PostgreSQL puede realizar la búsqueda de dos maneras diferentes. La primera consiste en realizar un recorrido secuencial sobre toda la tabla, hasta que el gestor encuentre la tupla que se desea obtener. Este tipo de búsquedas tiene como inconveniente, que se debe realizar una búsqueda sobre toda la tabla, implicando que el tiempo de ejecución de la consulta sea mucho mayor para aquellas tablas con un amplio volumen de datos, aspecto que podría ser mejorado si se realizara una búsqueda indexada. Un índice es una estructura auxiliar que permite acelerar el rendimiento de las búsquedas siempre que la consulta tenga condiciones asociadas a los índices y pueden ser creados utilizando una o más columnas de una tabla, aunque una incorrecta estrategia de indexado puede repercutir negativamente en el rendimiento, pues para aquellas tablas que posean un número pequeño de tuplas, resulta más costoso realizar una búsqueda indexada que secuencial, además las operaciones de inserción y actualización se tornan más lentas, ya que requieren de una modificación del índice. PostgreSQL ofrece cuatro tipos de índices: B-Tree, Hash, GIST y GIN. Cada uno de estos tipos utiliza un algoritmo diferente para determinados tipos de consulta. Si no se especifica el tipo de índice a utilizar, PostgreSQL crea por defecto los índices de tipo B-Tree, debido a que es el más utilizado.

- El panificador de consultas de PostgreSQL, puede considerar utilizar un índice de tipo B-Tree siempre que se realice una comparación sobre una columna utilizando los operadores <, <=, =, >=, >, además de la sentencia BETWEEN, IN y IS NULL.
- Los índices de tipo Hash solo pueden soportar operaciones de comparación utilizando el operador de igualdad (=) y no soportan las búsquedas cuando se utiliza la sentencia IS NULL. Este tipo de índice no ofrece un buen rendimiento, además de que no es registrado en los ficheros de WAL, lo

que trae como desventaja que una recuperación a partir de los ficheros de WAL, no permite la recuperación de los índices Hash, por lo que no es recomendable utilizar este tipo de índices.

- Los índices de tipo GiST son utilizados en operaciones sobre varios tipos de datos bidimensionales o datos geométricos como puntos, polígonos, círculos, entre otros.
- Los índices de tipo GIN, son índices que pueden soportar valores que contienen más de una llave, como es el caso de los arreglos.

En el modelo de datos del SEPYA, el tipo de índice utilizado fue el B-tree y fueron indexadas aquellas columnas que pueden llegar a contener un alto número de registros y sobre las cuales se realizan más operaciones de búsquedas, con el objetivo de hacer un uso eficiente del índice, un ejemplo de campos indexados son las columnas asociadas al número de cédula de los ciudadanos, el número del pasaporte, las llaves foráneas de las solicitudes existentes en las entidades asociadas a los trámites, los identificadores de los países que son referenciados desde la entidad que contiene las ciudades, entre otros. En el **anexo 9** se muestra el listado de los operadores.

2.4.5. Restricciones del modelo de datos

Este modelo cumple con un grupo de restricciones con el objetivo de apoyar la integridad de la base de datos:

- Todos los atributos se encuentran sujetos a un dominio con el objetivo de garantizar la consistencia y validez de los datos, exceptuando aquellos que son de tipo DATE o TIMESTAMP. En el **anexo 10** se muestran los dominios definidos para la base de datos.
- La integridad de la entidad se garantiza para todas aquellas entidades que poseen más de un atributo como llave primaria, pues ninguno de ellos puede tomar valores nulos.
- No existe más de una tupla perteneciente a una relación con la misma clave.
- Todas las relaciones entre entidades se encuentran sujetas a referencias entre entidades del modelo. Con ello se garantiza que una llave foránea apunte a una entidad con una tupla, fila o registro existente dentro de la base de datos, lo que permite minimizar la redundancia de los datos y con ello aumentar la integridad de los mismos.

2.4.6. Normalización

El proceso de normalización permite obtener menos información redundante, como resultado se reduce el espacio físico para almacenar los datos y la información se encuentra mejor organizada; además, permite eliminar las anomalías de actualización, pues cuando un elemento es insertado, modificado o eliminado, solo una tupla de una tabla necesita ser modificada. Al aplicar la normalización se fue comprobando que cada entidad cumpliera con las reglas y restricciones establecidas para cada una de las formas normales. De manera general, el modelo se encuentra normalizado hasta la 3FN, un ejemplo de esto son las entidades pertenecientes al sub-modelo dirección que se muestra en la **figura 11**. Estas entidades cumplen con las restricciones establecidas en la 1FN, pues poseen una llave primaria y sus dominios no tienen elementos que, a su vez, sean conjuntos. Cumplen además con las restricciones propuestas por la 2FN, pues toda relación que se encuentre en 1FN y que además su clave primaria no sea compuesta, se encuentra en 2FN y también se encuentran en 3FN pues no existen dependencias transitivas entre atributos no llaves.

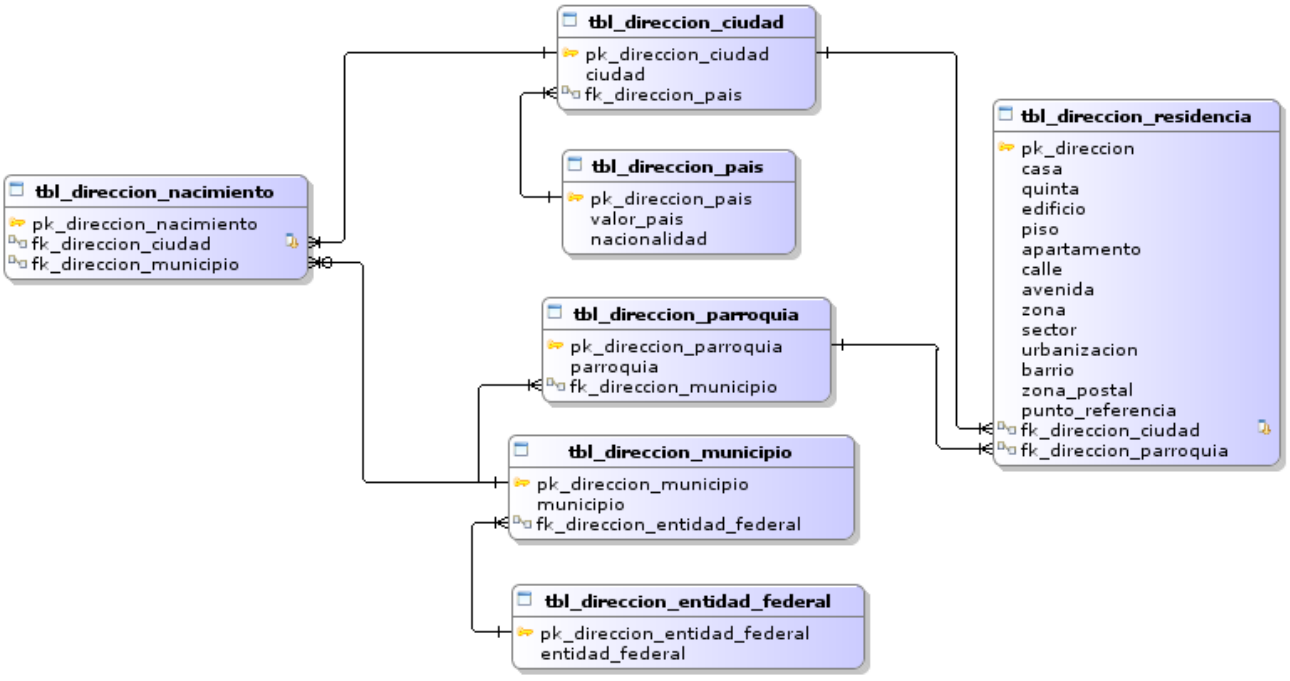


Figura 11 Sub-modelo dirección

Sin embargo en el modelo existen entidades, que por cuestiones de rendimiento se encuentran en 2FN. Un ejemplo de esto es la entidad *tbl_tramite_pasaporte*, la cual no cumple con las restricciones propuestas por la 3FN, que plantea que la relación debe estar en 2FN y además se deben eliminar las dependencias transitivas entre los atributos respecto a la llave primaria. La **figura 12** muestra la dependencia transitiva que existe entre los atributos *fk_identidad_ciudadano* y *fk_solicitud_documento_pasaporte,* debido a que una solicitud de pasaporte para un ciudadano tiene asociado un trámite, y con el objetivo de ganar en rendimiento, se decidió incluir en la tabla del trámite, el atributo *fk_identidad_ciudadano*, para de esta forma no tener que recurrir a la tabla *tbl_solicitud_documento* para obtener la información personal del ciudadano titular, cuyo proceso implicaría realizar otra búsqueda sobre otra tabla del modelo.

Figura 12. Dependencia transitiva entre atributos

2.4.7. Patrones de diseño utilizados

En el diseño y modelación de bases de datos, se presentan situaciones que se repiten en los modelos, que si son correctamente identificados llegan a convertirse en patrones de diseño. Un patrón de diseño es una solución a un problema determinado que surge durante el proceso de desarrollo de software que está probado y documentado; está definido como una *"solución común a un problema común de un determinado contexto"* (JACOBSON, y otros, 2000).

Durante el diseño de la base de datos para el SEPYA se utilizaron diferentes patrones de diseño como los patrones de asociación, patrón de llaves subrogadas y el patrón modelo para la seguridad de aplicaciones.

Los patrones de asociación representan las relaciones entre entidades del modelo, estos fueron utilizados para representar asociaciones de tipo muchos a muchos. Las bases de datos relacionales no soportan relaciones directas del tipo muchos a muchos. Esta situación puede ser resuelta añadiendo al modelo una tabla de asociación cuya funcionalidad es representar este tipo de relación entre varias entidades. La tabla resultante estará compuesta por las llaves primarias de las entidades a relacionar, las que a su vez formarán parte de la clave primaria y compuesta de la entidad resultante de la relación. En el modelo de base de datos que se propone, este patrón se pone de manifiesto a la hora de relacionar usuarios y roles del sistema, pues un usuario puede tener uno a varios roles mientras que un rol puede pertenecer a más de un usuario. Su representación se muestra en la **figura 13.**

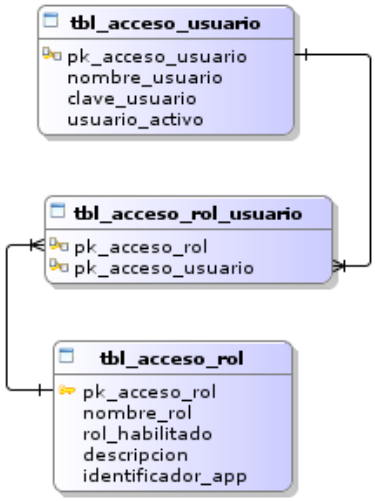


Figura 13. Patrón de asociación (muchos a muchos)

El patrón de llaves subrogadas es empleado cuando se desea generar una llave primaria única para cada entidad. Normalmente en los modelos de datos se utilizan números enteros en columnas auto-incrementales o identificadores del tipo UUID[31]. Este patrón es muy utilizado por las entidades que integran el modelo del SEPYA, ya que presentan en su mayoría identificadores de tipo UUID, además los

[31]Un UUID (Universally Unique Identifiers) es un entero de 128 bits representado en notación hexadecimal que se puede utilizar siempre que se requiera un identificador único. También se le conoce como GUID (Globally Unique Identifier).

números de serie de los trámites de pasaporte y de acreditación, son generados a través de secuencias auto-incrementales.

El patrón modelo para la seguridad de aplicaciones, a través del control de acceso basados en roles (RBAC), es sumamente utilizado en los sistemas de control de acceso. A través de este modelo, los permisos para ejecutar ciertas operaciones son asignados a roles específicos, y estos roles son asignados a los usuarios del sistema. El manejo de los permisos de cada usuario está determinado por la asignación del rol apropiado, ya que a los usuarios no se les asignan permisos directamente, sino que los adquiere a través de los roles que le son asignados. Este patrón se pone de manifiesto entre las entidades de la **figura 14,** donde un usuario que pertenece a una organización, puede poseer un conjunto de roles y las funcionalidades del sistema son asignadas al rol en específico.

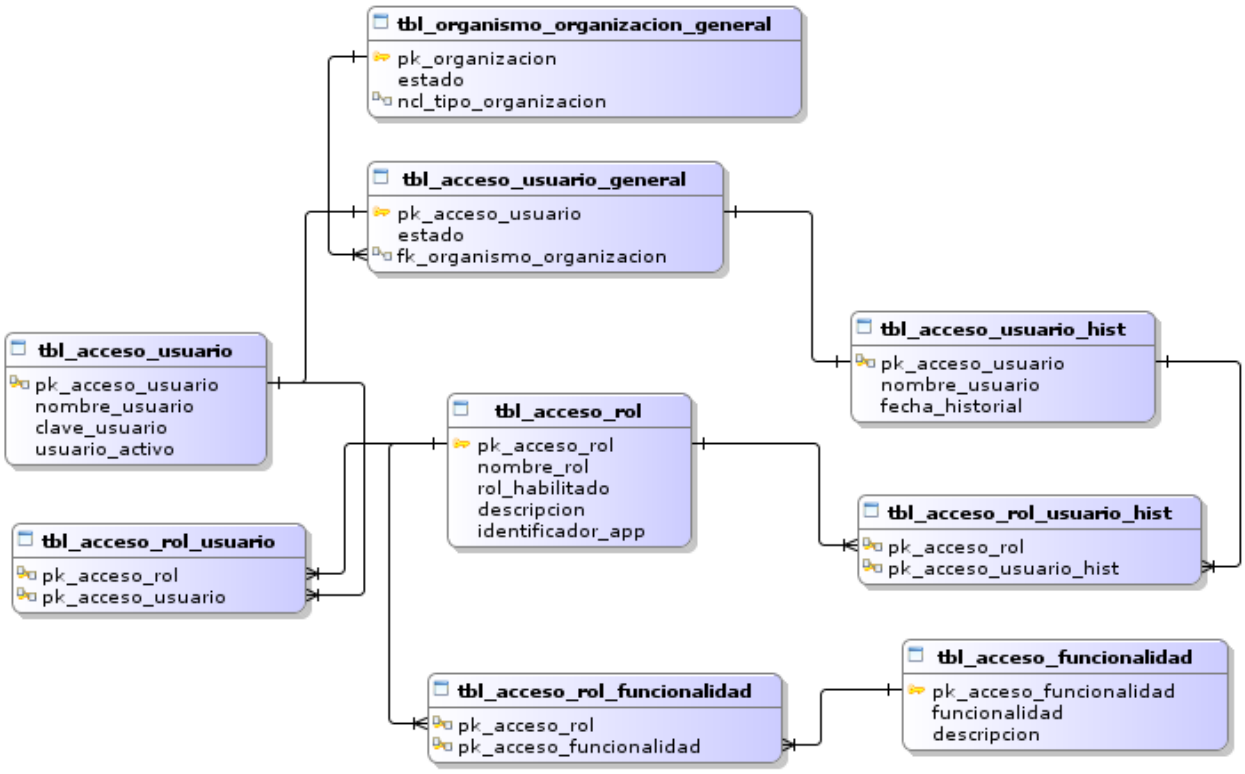


Figura 14. Patrón RBAC

2.4.8. Arquitectura de software

Una vez diseñado el modelo de datos del SEPYA y luego de haber identificado una herramienta para la replicación de los datos, es necesario establecer una arquitectura de software que garantice la correcta distribución y replicación entre los servidores. En el diagrama de software mostrado en la **figura 15** existen dos servidores de datos PostgreSQL. La comunicación entre las aplicaciones clientes y ambos servidores se realizará a través de Pgpool-II, quien actuará como software intermediario, interceptando las sentencias SQL y enviándolas síncronamente hacia los servidores; de esta forma se podrá balancear la carga debido a que Pgpool-II será capaz de distribuir aquellas sentencias que sean de solo lectura entre los servidores. Esta solución tiene como punto vulnerable, el nodo en que está instalado Pgpool-II, pues si este servidor falla, el software intermediario dejaría de funcionar.

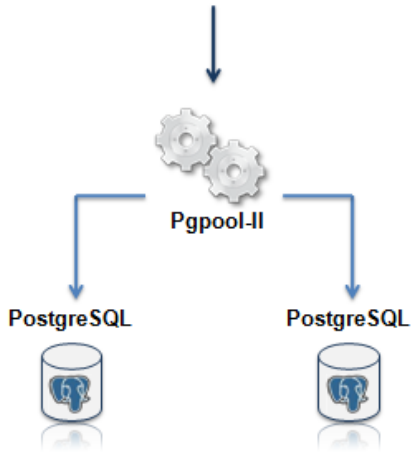


Figura 15. Diagrama de software

2.5. Conclusiones

- Las necesidades técnicas del sistema SEPYA permitieron identificar los elementos arquitectónicamente significativos.
- Se identificó a Pgpool-II como la herramienta para la replicación de los datos del sistema SEPYA.
- Se realizó el diseño de la base de datos del sistema SEPYA a partir de la descripción de los principales conceptos de negocio y de los requerimientos funcionales.
- Se estableció la arquitectura de software a utilizar para la base de datos del sistema SEPYA.

CAPÍTULO 3

CONFIGURACIÓN Y PRUEBA DE LA SOLUCIÓN PROPUESTA

A partir de la identificación de la herramienta de replicación y del diseño del modelo de datos, es el momento de instalar el sistema gestor de base de datos PostgreSQL y la herramienta PgPool-II, y realizar las configuraciones pertinentes, para lo que es necesario definir el ambiente donde será desplegado el software. La seguridad es un aspecto de vital importancia en las bases de datos, ya que la información puede perderse o alterarse, es por esto que es necesario tener en cuenta estos aspectos, tanto en el diseño, como a la hora de realizar una instalación. Una vez instalado el sistema, y con los requerimientos mínimos de seguridad, hay que contar con un procedimiento para la recuperación, que permita ante la caída de un servidor, restaurarlo y continuar brindando servicio. Las pruebas permiten comprobar si un sistema se comporta según lo esperado, por lo que aplicar pruebas de integridad al modelo y a la solución de replicación, permitirán verificar su funcionamiento y validez.

3.1. Propuesta de despliegue para la solución

"El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo" (JACOBSON, y otros, 2000). Se representa mediante un diagrama que muestra un conjunto de nodos y sus relaciones. "*Estos diagramas también pueden mostrar cómo se asignan los componentes ejecutables a los nodos*" (JACOBSON, y otros, 2000). El diagrama mostrado en la **figura 16** representa la distribución de los nodos físicos a utilizar en el sistema SEPYA y la comunicación entre ellos.

El sistema SEPYA se ejecutará sobre tres nodos servidores. El servidor de aplicaciones se conectará a Pgpool-II mediante el puerto 9999 a través del protocolo de comunicación Java Database Conection (JDBC), quien será el encargado de enviar las peticiones a los servidores de base de datos PostgreSQL 1 y 2 mediante el protocolo TCP/IP. Los servidores de datos se encontrarán aceptando conexiones por el puerto 5432. Los pasos para la instalación de PostgreSQL y Pgpool-II se encuentran descritos en los **anexos 11** y **12** respectivamente.

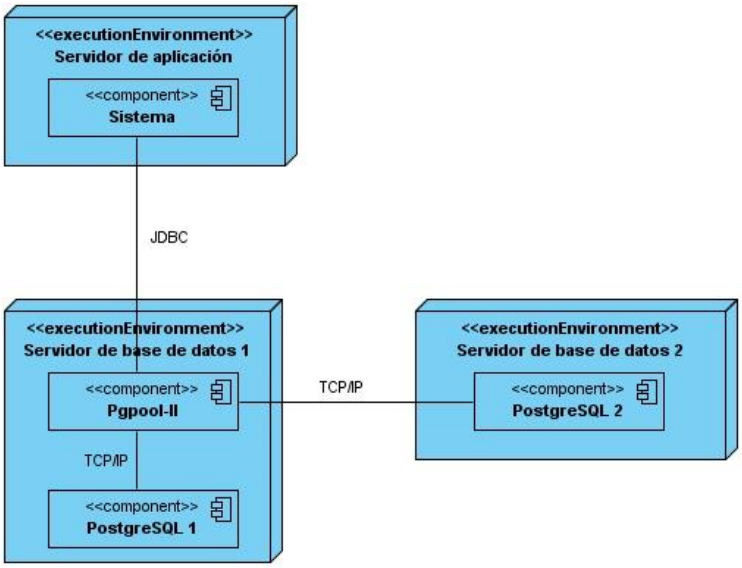


Figura 16. Diagrama de despliegue

3.2. Recuperación Online con Pgpool-II

El procedimiento de recuperación de un nodo brinda la posibilidad de volver a conectar nodos caídos al clúster, para ello es un requerimiento inicial que el servicio de PostgreSQL del nodo que se desee recuperar esté detenido. Para comprender este proceso, es necesario conocer cómo funciona el mecanismo de archivado de ficheros WAL. Todas las operaciones que alteran los datos de la base de datos son almacenados en ficheros de log binarios en el directorio pg_xlog, que se encuentra dentro del directorio data de PostgreSQL. El uso de estos ficheros viene activado por defecto, pero no su archivado o copia a otra ubicación. En PostgreSQL no es posible activar o desactivar el archivado de estos ficheros sin reiniciar el servicio, mientras que modificar el comando para su archivado, solo requiere de una recarga de la configuración del gestor; por esta razón se utiliza un comando de archivado nulo durante la ejecución normal del servidor, pues todo comando de archivado debe devolver al valor 0 si la ejecución es correcta.

3.2.1. Procedimiento de recuperación

Inicialmente es necesario crear un directorio en el que serán almacenados los ficheros de WAL generados durante la primera etapa de la recuperación, y debe ser creado fuera del directorio data de PostgreSQL. Para la presente investigación, el directorio fue creado dentro de /media/pgsql, teniendo como nombre pg_xlog_archive y como propietario el usuario postgres. Posteriormente, se modifica el fichero de configuración de PostgreSQL en ambos servidores, estableciendo los siguientes parámetros:

```
archive_mode = on
archive_command = 'exit 0'
```

El parámetro *archive_mode* establece el archivado de los ficheros y siempre debe estar activado, mientras que el parámetro *archive_command* especifica el comando a utilizar para el archivado. Cuando toma el valor *exit 0,* se indica que el comando devolverá un valor nulo. Una vez modificado dicho fichero se reinicia el servicio de PostgreSQL mediante el comando:

```
/etc/init.d/postgresql restart
```

El siguiente paso es hacer una llamada al comando *pcp_recovery_node*:
/opt/pgpool/bin/pcp_recovery_node <timeout> <hostname> <puerto> <usuario> <contraseña> <número de nodo>

- El parámetro *timeout* especifica el tiempo máximo que la llamada al procedimiento *pcp_recovery_node* debe esperar en caso que no se ejecute.
- El parámetro *hostname* especifica la dirección ip del host donde se encuentra instalado Pgpool-II.
- El parámetro *puerto* especifica el puerto a través del cual la interfaz PCP está aceptando conexiones.
- El *usuario* y la *contraseña* determinan los parámetros de autenticación con la interfaz PCP, coincidiendo con el usuario y la contraseña creada durante la configuración del fichero pcp.conf.
- El *número de nodo* especifica el número del servidor que se desea recuperar.

La ejecución del comando *pcp_recovery_node* parte de una llamada a un procedimiento almacenado en la base de datos, motivo por el cual todas las acciones que se realizan durante el proceso de recuperación, se ejecutan mediante el usuario postgres, divididas en tres etapas diferentes:

Primera etapa

Pgpool-II ejecuta SELECT pgpool_recovery('recovery_1st_stage_command', 'target', 'PGDATA') en el nodo maestro, produciendo una llamada al script */media/pgsql/data/base-backup*. Durante la ejecución de este script, Pgpool-II continúa aceptando peticiones de los clientes, y los ficheros WAL generados producto de estas peticiones, se irán almacenando en el directorio */media/pgsql/pg_xlog_archive*.

Los valores de los parámetros que se le pasan al procedimiento almacenado, se obtienen del fichero de configuración de Pgpool-II. El primer parámetro se adquiere de la directiva *recovery_1st_stage_command*, que tiene como valor el nombre del fichero *base-backup*, el parámetro *target* corresponde a la dirección ip del nodo que se desea recuperar y se consigue de la directiva *backend_hostname,* el tercero especifica la ruta del directorio data del nodo a recuperar, tomada de la propiedad *backend_data_directory.*

Inicialmente el script *base-backup*, activa el comando para el archivado de los ficheros WAL, que serán utilizados para la recuperación PITR y luego se solicita una recarga de la configuración. Seguidamente, se

crea un fichero llamado *recovery.conf*, que es enviado al nodo a recuperar. Cuando el servidor PostgreSQL del nodo caído inicie y realice una recuperación PITR, leerá dicho fichero y ejecutará el valor de la variable *restore_comand* tantas veces sea necesario para obtener los ficheros WAL generados, desde que se comenzó a realizar la copia de seguridad online, hasta que se dejaron de atender peticiones en la segunda etapa.

Posteriormente se inicia una copia de seguridad online mediante la ejecución de la consulta SELECT pg_start_backup('pgpool-recovery') en el nodo maestro forzando un checkpoint. Un checkpoint es un punto de control en la secuencia del log de transacciones, a partir del cual, todos los ficheros de datos serán escritos en disco y actúa como punto de inicio para una recuperación PITR. Posteriormente se ejecuta una copia del directorio */media/pgsql/data* desde el nodo maestro al nodo a recuperar, mediante el comando rsync sobre Secure Shell (ssh)[32], por lo que es necesario que la conexión sobre ssh no requiera de contraseña. Los pasos para establecer una conexión ssh sin contraseña se encuentran el **anexo 13.**

Finalmente se ejecuta la consulta SELECT pg_stop_backup(), para indicarle a PostgreSQL que el backup online a finalizado, desplazándose el puntero de inserción del fichero de log actual hacia el siguiente fichero de log, garantizándose que el fichero actual pueda rotarse y archivase, finalizando de esta forma la copia de seguridad. Este script se encuentra en el **anexo 14.**

Segunda etapa

Luego de finalizar la copia establecida en la primera etapa, Pgpool-II espera a que finalicen las peticiones que son ejecutas por los clientes durante un tiempo definido en la propiedad *recovery_timeout,* existente en el fichero de configuración de Pgpool-II. Si transcurrido este tiempo, las conexiones existentes no se han cerrado, no se ejecuta la 2da etapa de la recuperación, en caso contrario Pgpool-II deja de atender las peticiones de los clientes, quedando éstas acumuladas en una cola de peticiones pendientes, que serán atendidas una vez haya finalizado el proceso de recuperación. Luego se produce una llamada al procedimiento almacenado pgpool_recovery ('recovery_2nd_stage_command', 'target', 'PGDATA') en el

[32] SSH (**S**ecure Shell o intérprete de órdenes seguras) es el nombre de un protocolo y del programa que lo implementa, utilizado para conectarse a otro sistema Linux o Unix, que tenga un servidor ssh activo, y así poder ejecutar órdenes en él.

nodo maestro, para ejecutar el script *pgpool_recovery_pitr* almacenado dentro del directorio data del servidor. Los valores de los parámetros que espera este procedimiento se obtienen del fichero de configuración de Pgpool-II. El primer parámetro se adquiere de la directiva *recovery_2nd_stage_command*, que tendría como valor el nombre del fichero *pgpool_recovery_pitr*, el parámetro *target* corresponde a la dirección ip del nodo a recuperar y se consigue de la directiva *backend_hostname,* el tercero especifica la ruta del directorio data del nodo a recuperar tomada de la propiedad *backend_data_directory.*

Este script fuerza una rotación de ficheros WAL mediante la ejecución de la función *pg_switch_xlog().* Esta rotación de ficheros WAL se realiza con el propósito de generar un estado inalterado de la base de datos, que pueda ser recuperado por el nuevo nodo a partir del backup del directorio base, más los ficheros WAL generados por sucesivas rotaciones desde el inicio del backup, hasta esta última rotación explícita indicada a través de la ejecución de la función pg_stop_backup() de la primera etapa. Finalmente, se establece una comando de archivado nulo sobre la propiedad *archive_command* del fichero de configuración de PostgreSQL y se solicita una recarga de la configuración. El script *pgpool_recovery_pitr* se encuentra en el **anexo 15**.

Tercera etapa

En la tercera etapa Pgpool-II ejecuta una llamada al procedimiento almacenado pgpool_remote_start('target', 'PGDATA') en el nodo maestro, produciendo una llamada al script *pgpool_remote_start* almacenado dentro del directorio data del servidor. Los valores de los parámetros que espera el procedimiento almacenado se obtienen del fichero de configuración de Pgpool-II. El parámetro *target* corresponde a la dirección ip del nodo a recuperar y se consigue de la directiva *backend_hostname,* el segundo especifica la ruta del directorio data del nodo a recuperar tomada de la propiedad *backend_data_directory.* El objetivo de este script es iniciar PostgreSQL en el nodo que se está recuperando mediante una llamada al binario */usr/bin/pg_ctl*, y PostgreSQL hace una recuperación PITR al iniciar el servicio. Este script se encuentra en el **anexo 16.**

Una vez finalizado el proceso de PITR, el nodo recuperado está en condiciones de aceptar conexiones nuevamente. El clúster opera de nuevo con normalidad, todas las bases de datos son accesibles de nuevo y se procesan las peticiones acumuladas durante la segunda y tercera etapa de la recuperación en línea.

3.3. Seguridad en las bases de datos

En la actualidad uno de los puntos más críticos en las base de datos es la seguridad, pues muchos sistemas están expuestos a ataques, y en la mayoría de los casos no cuentan con una infraestructura de protección. La información almacenada en una base de datos está en constante riesgo de sufrir ataques que puedan provocar su modificación o pérdida, por ello es de vital importancia velar la seguridad de la misma, protegiéndola contra accesos no autorizados o cualquier acción que puedan violar la integridad y confidencialidad de los datos. “(…) La integridad radica en asegurar que los datos almacenados sean correctos, es decir verídicos (…)” (DEPARTAMENTO DE SEGURIDAD EN COMPUTO/UNAM-CERT , 2010). La disponibilidad de los datos requiere que la información esté en disposición de aquellos que deban acceder a ella, en el momento que la necesiten, por lo que contar con mecanismos que permitan recuperar la base de datos contra fallos, es un factor clave, para así no comprometer este principio.

3.3.1. Seguridad en PostgreSQL

En PostgreSQL la seguridad puede ser materializada en diferentes aspectos, en primer lugar debe garantizarse la seguridad en la manipulación de los ficheros, es decir debe establecerse al usuario del sistema postgres como el propietario de los ficheros y definirle los permisos de lectura, escritura y ejecución sobre el directorio DATA que contiene la información más crítica.

Otro aspecto importante a tener en cuenta es la seguridad en los accesos de los clientes, ya que es necesario definir que usuarios se conectarán a las bases de datos, desde que equipo y por cual método de autenticación lo harán. La seguridad en este nivel se realiza mediante el fichero *pg_hba.conf* (de sus siglas en inglés host based authentication), fichero de configuración para la autenticación de los clientes y usuarios, en él se define el acceso a las bases de datos del clúster.

El uso de conexiones SSL para cifrar las comunicaciones entre los clientes y el servidor permiten incrementar la seguridad, ya que la información viaja de forma cifrada. Para esto se requiere tener instalado OpenSSL en el servidor y PostgreSQL compilado con soporte SSL. Para activar el soporte SSL es necesario establecer el parámetro ssl en on, que por defecto viene en off, en el fichero de configuración *postgresql.conf*; se debe además contar con un certificado y su llave correspondiente firmados por una Autoridad de certificación.

A continuación se brinda una serie de pasos que pueden servir de guía para configurar PostgreSQL con soporte para conexiones seguras usando SSL:

1. Instalar OpenSSL en el servidor.
2. Instalar PostgreSQL con soporte para SSL. Durante la compilación se consigue utilizando el parámetro --with-openssl en. /configure.
3. Crear un certificado auto-firmado para el servidor en el directorio DATA:

Generar una solicitud de certificado:

- openssl req -new -text -out server.req

Generar una llave con algoritmo RSA:

- openssl rsa -in privkey.pem -out server.key

Borrar el archivo .pem creado, para evitar una posible pérdida de la llave privada e impedir que el servidor se vea comprometido:

- rm privkey.pem

Con el siguiente comando se genera un certificado auto-firmado X.509 y se almacena en el fichero server.crt:

- openssl req -x509 -in server.req -text -key server.key -out server.crt

A continuación se establece al usuario postgres como el propietario de los ficheros generados y se asignan permisos de lectura, escritura y ejecución sobre el fichero server.key:

- chown postgres:postgres server.*
  chmod og-rwx server.key

4. Establecer en el fichero de configuración postgresql.conf el parámetro ssl=on.
5. Actualizar las reglas de acceso, editando el fichero pg_hba.conf:
   TYPE DATABASE USER CIDR-ADDRESS METHOD

En el apartado TYPE se coloca el tipo de conexión que se usará: local y remota. La forma “local” se refiere a los usuarios que acceden desde el servidor en que se aloja el gestor de base de datos, en lo que respecta a “host” son aquellas conexiones remotas tanto de redes tipo ipv4 como soporte para ipv6. También está en la parte de host la opción con soporte para SSL “hostssl”.

El parámetro DATABASE es para especificar el nombre de la base de datos a la que se pueden conectar los usuarios, si se desea que el usuario se conecte a todas las bases de datos se puede utilizar la palabra reservada “all”.

El parámetro USER indica el usuario que tendrá acceso a las bases de datos que le sean permitidas. De igual forma se hace referencia a todos con la palabra reservada “all”.

CIDR-ADDRESS, es la columna de la IP con la máscara de red que puede acceder al servidor. Para el caso de una conexión local se deja en blanco.

METHOD, define el tipo de autenticación que el servidor debería usar para un usuario que intenta conectarse a PostgreSQL. Existen varios métodos posibles a utilizar, en este caso se usa md5, donde se envían los datos de autenticación con el algoritmo md5.

6. Reiniciar el servicio postgresql para que los cambios aplicados a los ficheros de configuración surjan efecto a través del comando: */etc/init.d/postgresql restart*

PostgreSQL contiene además privilegios de acceso y restricciones aplicables para cada objeto de la base de datos, como por ejemplo tablas, vistas y secuencias. Para esto hace uso de las sentencias REVOKE y GRANT en donde se revocan y otorgan privilegios respectivamente. Al hacer un buen diseño de la base de datos y especificar los tipos de datos y sus dominios, PostgreSQL es capaz de garantizar que en la base de datos pueda controlarse la redundancia a su mínima expresión, manteniendo la consistencia y así conseguir la integridad de los datos.

3.4. Pruebas

Para la validación teórica del diseño de la base de datos se realizarán pruebas de integridad al modelo y para la validación funcional de la propuesta de solución, se realizarán pruebas que permitan verificar que se realice la replicación de los datos, el balance de carga y la recuperación.

3.4.1. Pruebas de integridad al modelo

Con el objetivo de garantizar la integridad de los datos almacenados, se realizaron un conjunto de pruebas, entre las que se encuentran las pruebas de integridad de entidad, integridad de dominio e integridad referencial. Con las pruebas de integridad de entidad, se verifica que las claves primarias de las entidades del modelo, no puedan tomar valores nulos, ni duplicados; para lograr esto, PostgreSQL asigna automáticamente restricciones UNIQUE y NOT NULL sobre las claves primarias de las entidades. Un ejemplo de ello, se evidencia en la **figura 17**, donde, al tratar de insertar un valor nulo en la clave primaria de la entidad *tbl_nvinculo_familiar (pk_vinculo_familiar)*, el gestor lanza un error; igualmente sucede si se trata de insertar una clave primaria duplicada, como se muestra en la **figura 18.**

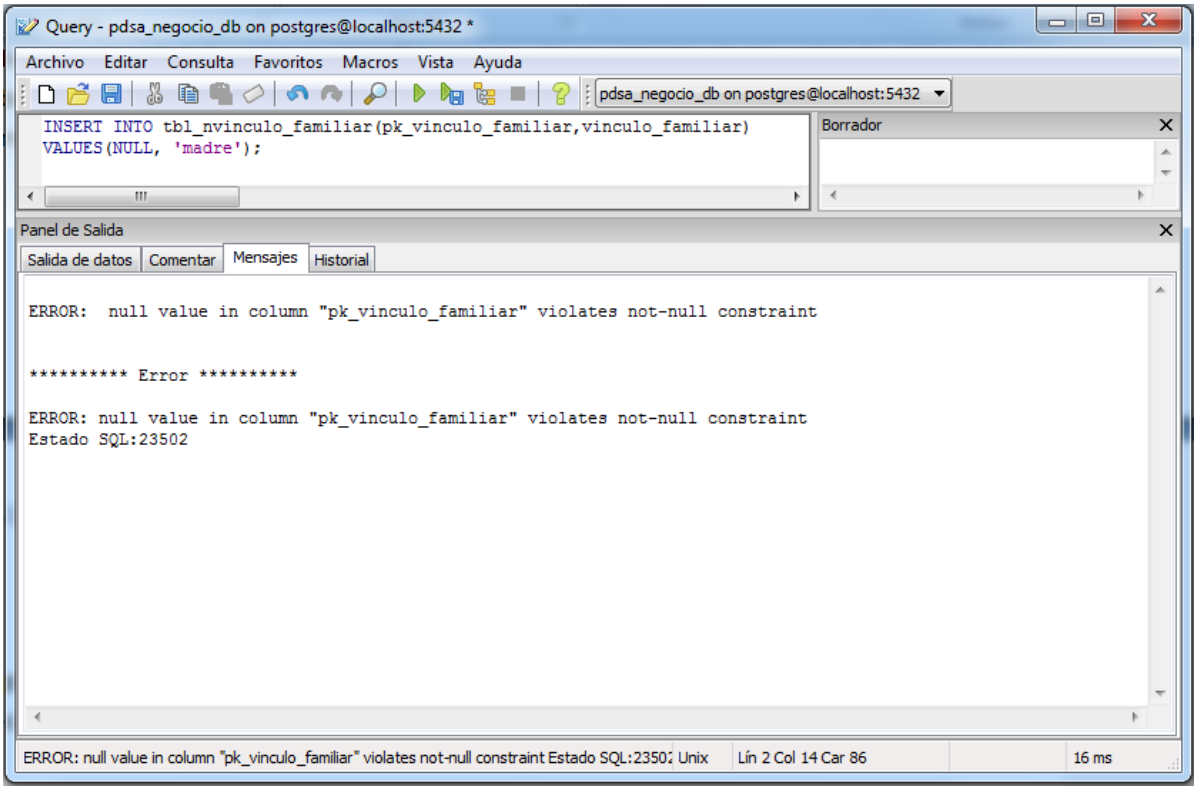


Figura 17. Error mostrado al tratar de insertar una clave primaria nula

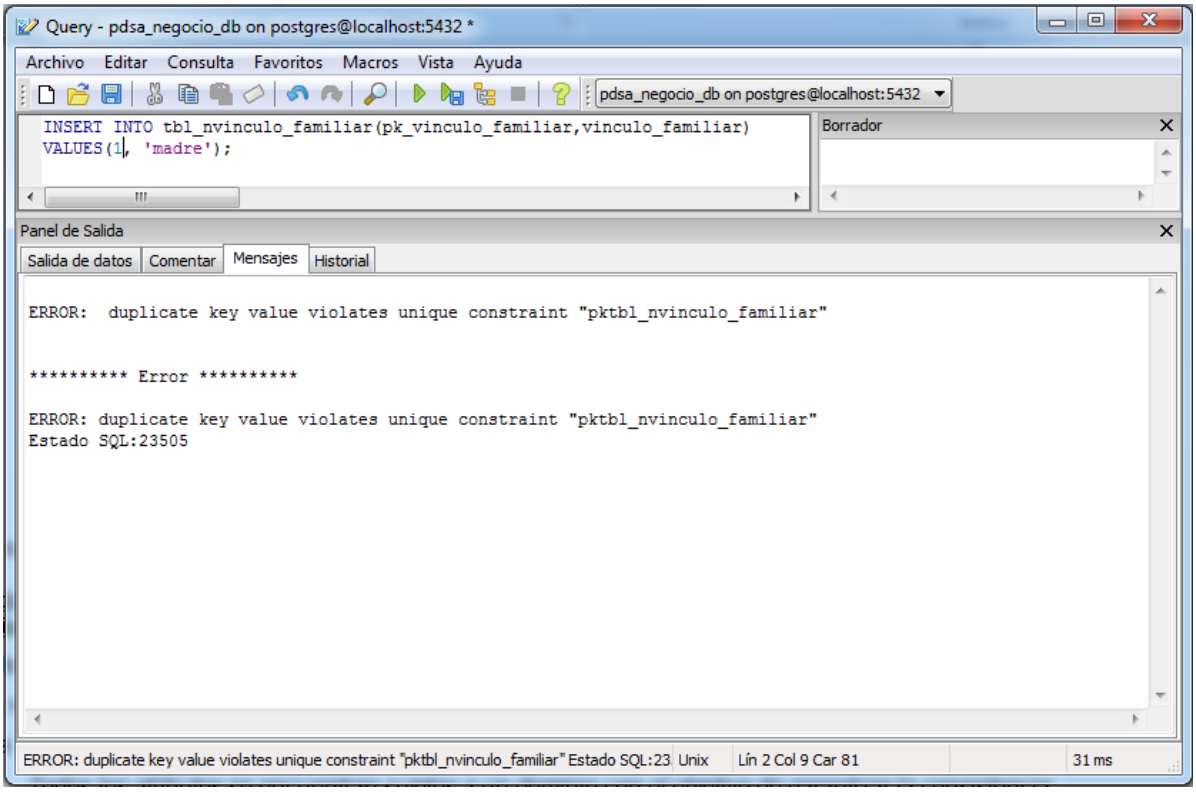


Figura 18. Error mostrado al tratar de insertar una clave primaria duplicada

Los campos de la base de datos, están sujetos a un conjunto de dominios, con el objetivo de garantizar la validez de las entradas para una columna determinada. Mediante las pruebas de integridad de dominio, se garantiza que cada valor que se desee insertar, cumpla con la regla definida en el dominio. Un ejemplo de ello, se muestra en la **figura 19**, en la cual, al tratar de realizar una inserción sobre la entidad *tbl_direccion_pais*, el gestor lanza un error, indicando que el valor viola la restricción implementada para el dominio *dom_cod_pais*, el cual solo admite letras mayúsculas de la A a la Z, y como máximo 3 caracteres.

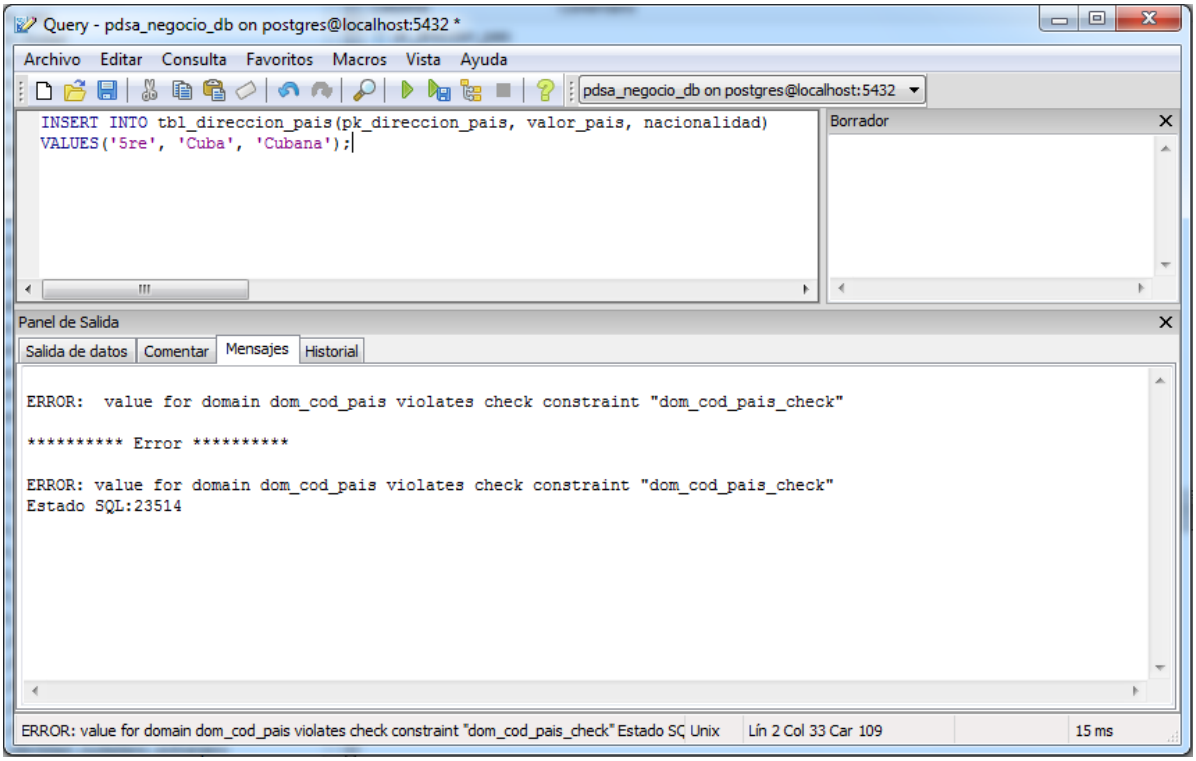


Figura 19. Error mostrado al tratar violar la restricción impuesta por el dominio dom_cod_pais

"*La integridad referencial garantiza que las relaciones entre filas de tablas relacionadas son válidas y que no se eliminan o se cambian datos relacionados de forma accidental*" (MICROSOFT, 2011). Con ella se garantiza que los valores de clave sean coherentes entre las diferentes tablas del modelo, o sea, al modificar una llave que es referenciada desde otras tablas, esta debe actualizarse en toda la base de datos, de igual forma, no deben existir referencias a valores inexistentes. Cuando se exige la integridad referencial, PostgreSQL impide de forma automática agregar o modificar filas en una tabla relacionada, si no existe una fila asociada en la tabla principal; de igual forma impide eliminar filas de una tabla principal, cuando haya filas de otras tablas asociadas a ella, a menos que las actualizaciones y eliminaciones se realicen en cascada. Un ejemplo de ello lo demuestra la **figura 20,** en la cual al tratar de eliminar una fila de la entidad *tbl_estado_identidad_ciudadano*, el gestor lanza un error, debido a que esta fila es referenciada en otras entidades del modelo.

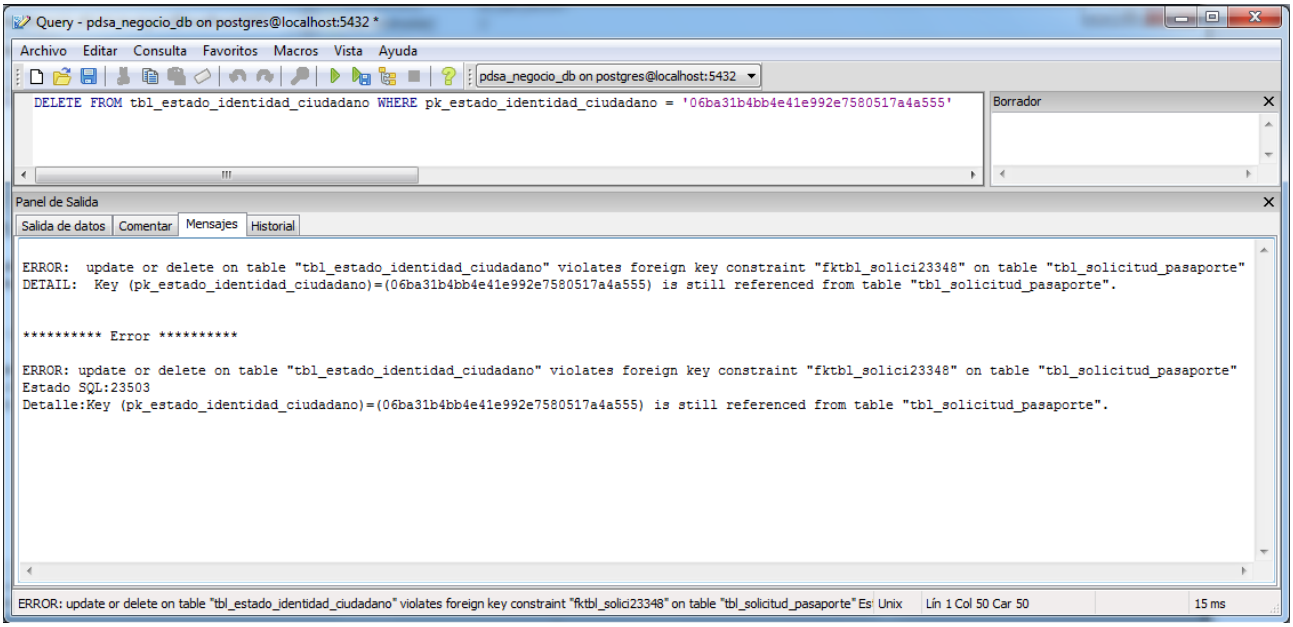


Figura 20. Error emitido por el gestor al tratar de violar la integridad referencial

3.4.2. Validación funcional de la solución propuesta

Con el objetivo de validar la herramienta de replicación seleccionada, se realizaron un conjunto de pruebas, para verificar el comportamiento de las funcionalidades que componen la propuesta de solución. Estas pruebas se encuentran enumeradas en la **tabla 4**:

Número	Objetivo
1	Comprobar el correcto funcionamiento de la réplica de los datos entre los servidores, ejecutando sentencias de inserción sobre la base de datos.
2	Comprobar el correcto funcionamiento del balanceador de carga, ejecutando sentencias de lectura sobre la base de datos.
3	Comprobar el procedimiento de recuperación en línea de un nodo caído.

Tabla 4. Pruebas funcionales

Para analizar el número de sentencias SQL ejecutadas sobre la base de datos, se utilizó una herramienta para analizar los ficheros de log del servidor denominada pgFouine. Esta herramienta permite determinar la cantidad de consultas de lectura, escritura, modificación y eliminación, así como las consultas más utilizadas y su tiempo de ejecución. Para ello se activó la propiedad log_statement = 'all' en el fichero de configuración de cada uno de los servidores, con el objetivo de almacenar en los ficheros de log todas las operaciones realizadas sobre la base de datos.

Prueba 1: Mediante esta prueba se realizaron un total de 5206 inserciones a la base de datos. Seguidamente se examinó el fichero de log de PostgreSQL a través de pgFouine, cuyo análisis arrojó que en ambos nodos fueron ejecutadas el mismo número de consultas, lo que demuestra que la replicación se ejecutó acorde a lo esperado.

Prueba 2: Para verificar el correcto funcionamiento del balanceador de carga, se utilizó una herramienta que ofrece PostgreSQL, denominada pgbench, que permite realizar pruebas de rendimiento a los servidores de datos, mediante la ejecución de un conjunto de sentencias previamente definidas. Esta herramienta permite además, simular un conjunto de conexiones concurrentes a la base de datos y el número de transacciones a ejecutar por cada conexión a la base. El script utilizado por pgbench se encuentra en el **anexo 17** y la sentencia ejecutada fue la siguiente:

pgbench -c 80 -n -f /home/pasaporte/Desktop/test.sql -t 100 -h 10.13.4.200 -p 9999 -U postgres pdsa_negocio_db

En la siguiente tabla se especifican los parámetros utilizados en el comando anterior:

Parámetros	Especificación
-c	Se establece el número de conexiones concurrentes a la base de datos.
-n	Esta opción se utiliza cuando no se emplean las tablas que por defecto se pueden crear a través de pgbench para realizar las pruebas.
-f	Se establece la ruta del fichero que contiene las sentencias SQL a ejecutar desde pgbench.
-t	Especifica el número de transacciones a ejecutar por cada conexión.
-h	Dirección IP del servidor sobre el que se realizará la prueba.

-p	Puerto mediante el cual se realizará la conexión.
-U	Usuario con el que se accederá a la base de datos.

Tabla 5. Parámetros de pgbench

De forma general fueron ejecutadas 100 transacciones por cada usuario concurrentemente, y como el script utilizado para la prueba contiene 4 consultas de lectura, deberían enviarse hacia los servidores de datos un total de 32000 consultas. Una vez finalizadas las transacciones generadas por pgbench, se realizó un análisis sobre el fichero de log de PostgreSQL, cuyo resultado permitió validar el correcto funcionamiento del balanceador de carga que brinda Pgpool-II. Con el análisis generado por PgFounie, se obtuvo que en el nodo 1 fueron ejecutadas 14 560 consultas y en el nodo 2 fueron ejecutadas 17 440 consultas, aunque estos resultados no siempre serán iguales, debido a la forma en que Pgpool-II realiza el balanceo de carga.

Prueba 3: Para realizar esta prueba, fue detenido el servicio de PostgreSQL en uno de los servidores, desarrollándose en primera instancia el proceso de failover automático y posteriormente se ejecutó la restauración y recuperación del nodo, cuya ejecución fue satisfactoria, quedando validada la solución propuesta para la recuperación.

3.5. Análisis de los resultados alcanzados en base a la solución existente

La solución propuesta ofrece mejoras significativas con respecto a la solución existente en el MPPRE, toda la información asociada a los trámites de pasaporte y acreditación, que en un inicio era procesada de forma manual y con algunos sistemas informáticos, ineficientes para la gestión de este tipo de documentos, será persistida en una base de datos relacional con el SGBD PostgreSQL, lo que permitirá agilizar los procesos de inserción, actualización y las operaciones de búsquedas sobre los datos, elevando la velocidad de respuesta y el número de peticiones realizadas a la base de datos. Además, mediante la utilización de este gestor, es posible elevar en gran medida el volumen de información almacenada, la cantidad de usuarios accediendo concurrentemente a los datos y la seguridad e integridad de los mismos.

Otra de las mejoras que se obtiene, es la posibilidad de validar la identidad de los ciudadanos que son tramitados, mediante la persistencia de sus datos biométricos, adicionándole una mayor seguridad al

proceso. El modelo además, contiene todas las entidades necesarias para almacenar los datos históricos, que son utilizados para consulta y en la generación de reportes, lo que permite llevar un mejor control de los documentos de identificación emitidos y anulados.

Un elemento importante es la disponibilidad, ya que es imprescindible contar con la información en el momento en que se necesite. En los trámites de pasaporte y acreditaciones al llevarse el proceso prácticamente de forma manual, esta se ve comprometida, pues el acceso a la información es engorroso al tener que realizar búsquedas en archivos físicos, lo que ralentiza el proceso, y en ocasiones puede perderse, teniendo en cuenta la vulnerabilidad que representa el factor humano. La solución propuesta eleva la disponibilidad de la información, mediante el empleo de Pgpool-II para la sincronización entre los servidores de datos, garantizando tener una réplica de la información en otro servidor, donde ante fallos en uno de ellos, se pueda continuar brindando servicio con el otro.

3.6. Conclusiones

- El procedimiento para la restauración de un nodo de bases de datos del SEPYA, permite conectar nuevamente el nodo al clúster, para que continúe brindando servicio.
- Las pruebas teóricas aplicadas al modelo de datos permitieron validar su integridad.
- Las pruebas funcionales realizadas a la propuesta de solución, permitieron validar el correcto funcionamiento de la réplica, el balance de carga y la recuperación.

CONCLUSIONES GENERALES

- Se diseñó una base de datos relacional para la persistencia de la información del SEPYA.
- El modelo de datos del SEPYA se encuentra normalizado y con la mínima redundancia de información.
- La propuesta de una herramienta para la replicación y un mecanismo para la restauración y recuperación de los nodos que integren el clúster, permitió una mejor disponibilidad de los datos.

RECOMENDACIONES

Durante el desarrollo del presente trabajo de diploma, se desarrollaron un conjunto de temas que dieron cumplimiento a los objetivos propuestos; a su vez existen otros temas que no fueron analizados y por su marcada importancia se consideran necesarios para darle continuidad a la investigación:

- Mantener actualizado el sistema gestor de base de datos PostgreSQL para fortalecer la solución propuesta.
- Investigar los elementos asociados a la optimización del gestor de datos y del sistema operativo en el que se despliegue, para de esta forma incrementar el rendimiento del sistema.
- Identificar herramientas y técnicas para la monitorización de los servidores de datos.

BIBLIOGRAFÍA

DEPARTAMENTO DE SEGURIDAD EN COMPUTO/UNAM-CERT . 2010. Implantación de Bases de Datos Seguras en PostgreSQL V 8.2.6. [En línea] 01 de 10 de 2010. [Citado el: 28 de 05 de 2011.] http://www.seguridad.unam.mx/doc/?ap=tutorial&id=204.

Bases de datos Orientadas a Objetos y Bases de datos Objeto-Relacionales. **MANZANEQUE, ALEJANDRO ALBERCA y DIAZ-TENDERO, JESUS GALVEZ.** Universidad de Castilla-La Mancha : s.n.

BECERRIL, FRANCISCO. 1998. *Java a su alcance.* México D.F : Editorial Mexicana, 1998.

BONANATA, MAXIMILIANO. 2003. *Programación y algoritmos.* Buenos Aires, Argentina : MP Ediciones , 2003.

2011. bucardo.org. *bucardo.org.* [En línea] 17 de 3 de 2011. [Citado el: 27 de 2 de 2011.] http://bucardo.org/wiki/Bucardo.

CASANOVA, JAIME. 2011. *Replicación en PostgreSQL.* 2011.

—. 2011. *Warm Standby o Log Shipping.* 2011.

GARCIA, ROSA MARIA MATO. 2005. *Sistemas de bases de datos.* Ciudad de la Habana : Editorial Pueblo y Educación, 2005.

Group, The PostgreSQL Global Development. 2008. *PostgreSQL 8.3.11 Documentation.* California : s.n., 2008.

GUERRA, ALDO SIGNORELLI. 2005. Convenio de colaboración entre el Servicio de Registro Civil e Identificación y el Ministerio de Relaciones Exteriores para la entrega de cédulas de identidad y pasaportes en los consulados de Chile en el exterior. [En línea] 2005. [Citado el: 25 de 01 de 2011.] http://www.iij.derecho.ucr.ac.cr/archivos/documentacion/inv%20otras%20entidades/CLAD/CLAD%20X/documentos/signorel.pdf.

HERRERA, ALVARO y CASANOVA, JAIME. 2011. *PostgreSQL Replicación, balance de carga y alta disponibilidad.* 2011.

Introducción a las Bases de Datos. Modelo Entidad-Relación (ER). **DEPARTAMENTO INGENIERIA Y GESTION DE SOFTWARE, UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS. 2006.** Ciudad de la Habana : s.n., 2006.

JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES. 2000. *El proceso unificado de desarrollo del software.* Madrid : Pearson educacion SA, 2000.

LEÓN, ROLANDO ALFREDO HERHÁNDEZ y GONZÁLEZ, SAYDA COELLO. 2002. *El paradigma cuantitavo de la investigación científica.* Ciudad de la Habana : Editorial Universitaria, 2002. 959-16-0343-6.

MARTÍNEZ, RAFAEL. 2009. PostgreSQL-es. *PostgreSQL-es.* [En línea] 6 de 6 de 2009. [Citado el: 8 de 3 de 2011.] http://www.postgresql-es.org/node/297.

—. 2009. PostgreSQL-es. *PostgreSQL-es.* [En línea] 6 de 5 de 2009. [Citado el: 10 de 3 de 2011.] http://www.postgresql.org.es/node/249.

—. 2009. PostgreSQL-es. *PostgreSQL-es.* [En línea] 29 de 3 de 2009. [Citado el: 15 de 3 de 2011.] http://www.postgresql.org.es/node/218.

—. 2011. PostgreSQL-es. *PostgreSQL-es.* [En línea] 29 de 3 de 2011. [Citado el: 25 de 3 de 2011.] http://www.postgresql.org.es/node/219.

—. 2009. PostgreSQL-es.org. *PostgreSQL y el uso de SSL.* [En línea] 1 de 12 de 2009. [Citado el: 26 de 2 de 2011.] http://www.postgresql.org.es/node/384.

MICROSOFT. 2008. MICROSOFT TECHNET. [En línea] 11 de 2008. [Citado el: 20 de 2 de 2011.] http://technet.microsoft.com/es-es/library/ms151198(SQL.90).aspx.

—. 2011. MSDN. *Cómo funciona la integridad referencial con una base de datos de Oracle.* [En línea] 2011. [Citado el: 16 de 2 de 2011.] http://msdn.microsoft.com/es-es/library/aa290252%28v=vs.71%29.aspx.

Modelos de datos. **Sevilla, Departamento de Lenguajes y Sistemas Informáticos E.T.S. Ingeniería Informática. Universidad de. 2005.** 2005.

ORACLE CORPORATION. 2002. Introduction to Advanced Replication. *Introduction to Advanced Replication.* [En línea] 2002. [Citado el: 10 de 5 de 2011.] http://download.oracle.com/docs/cd/B10501_01/server.920/a96567/repoverview.htm#12636. A96567-01.

—. 2002. Oracle 9i Advanced Replication. *Oracle 9i Advanced Replication.* [En línea] 2002. [Citado el: 10 de 5 de 2011.] http://download.oracle.com/docs/cd/B10501_01/server.920/a96567/repmaster.htm#32387. A96567-01.

PGCluster. 2005. PGCluster the multi-master and synchronous replication system for postgreSQL. *PGCluster the multi-master and synchronous replication system for postgreSQL.* [En línea] 7 de 3 de 2005. [Citado el: 20 de 2 de 2011.] http://pgcluster.projects.postgresql.org/index.html.

PgPool Global Development Group. 2011. Pgpool-II. *Pgpool-II.* [En línea] PgPool Global Development Group, 13 de 3 de 2011. [Citado el: 15 de 1 de 2011.] http://pgpool.projects.postgresql.org/.

PostgreSQL Web Team. 2010. PostgreSQL Wiki. *PostgreSQL Wiki.* [En línea] 8 de 3 de 2010. [Citado el: 26 de 2 de 2011.] http://wiki.postgresql.org/wiki/PgCluster.

—. 2010. PostgreSQL Wiki. *PostgreSQL Wiki.* [En línea] 25 de 9 de 2010. [Citado el: 20 de 2 de 2011.] http://wiki.postgresql.org/wiki/Replication%2C_Clustering%2C_and_Connection_Pooling.

—. 2010. PostgreSQL Wiki. [En línea] 3 de 12 de 2010. [Citado el: 18 de 3 de 2011.] http://wiki.postgresql.org/wiki/Slony.

—. 2010. PostgreSQL Wiki. *PostgreSQL Wiki.* [En línea] 21 de 8 de 2010. [Citado el: 18 de 3 de 2011.] http://wiki.postgresql.org/wiki/Pgpool-II.

—. 2010. PostgreSQL Wiki. *PostgreSQL Wiki.* [En línea] 8 de 3 de 2010. [Citado el: 26 de 2 de 2011.] http://wiki.postgresql.org/wiki/PgCluster.

POWELL, GAVIN. 2006. *Beginning Database Design.* Indianapolis : Wiley Publishing, Inc., 2006. 13: 978-0-7645-7490-0.

Requisitos, Catálogo de. 2010. *Catálogo de requisitos.* 2010.

ROBERTO MEDINA HERRERA. 2011. Contrato de prestación de servicios para la explotación del sistema de Identificación, cédulas de identidad y pasaportes para el servicio de Registro Civil e Identificación. [En línea] 6 de 1 de 2011. [Citado el: 25 de 5 de 2011.] http://www.registrocivil.cl/bases/PDF/Res_2_TratoDirecto_SrCeI_Sonda.pdf.

ROZIC, SERGIO EZEQUIEL. 2004. *Bases de datos* . Buenos Aires, Argentina : MP Ediciones, 2004.

Sabater, Jaume. 2009. PostgreSQL-es. *PostgreSQL-es.* [En línea] 6 de 7 de 2009. [Citado el: 10 de 10 de 2010.] http://www.postgresql-es.org/node/313.

SÁNCHEZ, JORGE. 2004. *Diseño conceptual de bases de datos.* 2004.

SANTIESTEBAN, ELIZABETA ROJAS y BOADO, ANGEL ERLYS BOLOY. 2010. *Arquitectura y diseño de la Base de Datos Única de los Sistemas de Identificación, Inmigración y Extranjería de la República de Cuba.* Habana : s.n., 2010.

Slony Development Group. 2008. Slony-I. *Slony-I.* [En línea] 12 de 9 de 2008. [Citado el: 25 de 2 de 2011.] http://slony.info/documentation/1.2/index.html.

SONDA. 2010. SONDA, Líder Latinoamericano de Servicios TI. [En línea] 2010. [Citado el: 09 de Marzo de 2011.] http://www.sonda.cl/.

STEPHENS, ROD. 2009. *Beginning Database Design Solutions.* Indianapolis : Wiley Publishing, Inc., 2009. 978-0-470-38549-4.

GLOSARIO DE TÉRMINOS

- Titular: Es la persona que recibe un documento de pasaporte o acreditación.
- Organismo: organización con facultades para solicitar los documentos de pasaporte y acreditación.
- Partida de nacimiento: documento legal, expedido por una institución autorizada que da fe del hecho del nacimiento, fecha en que tuvo lugar, sexo, y en su caso, de la hora en que se produjo el nacimiento del inscrito.
- Acta de fe de soltería: documento legal que debe presentarse cuando una hija de un funcionario tiene como estado civil soltera.
- LOPNA: Ley Orgánica de Protección al Niño y al Adolescente.
- Autorización LOPNA: normativa legal para trasladar a menores dentro y fuera del país.
- Certificado de Divorcio: documento legal que avala el cambio de estado civil a divorciado.
- Certificado de Matrimonio: documento oficial que acredita la celebración del matrimonio.
- Certificado de adopción: documento legal que debe presentarse cuando el titular del pasaporte es un menor adoptado.
- Certificado Médico de Capacidad Diferente: documento legal que debe presentarse cuando el titular tiene algún tipo de discapacidad.
- Constancia de Estudio: documento legal que debe presentarse cuando el titular está cursando estudios.
- Resolución de nombramiento: documento legal que avala el rango o categoría de una persona.
- Datos biométricos: se refiere a los datos que incluyen fotografía, firma y huella.
- Log: es un registro de actividad, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema.
- Transacción: sentencia o conjunto de estas que son ejecutadas sobre la base de datos.

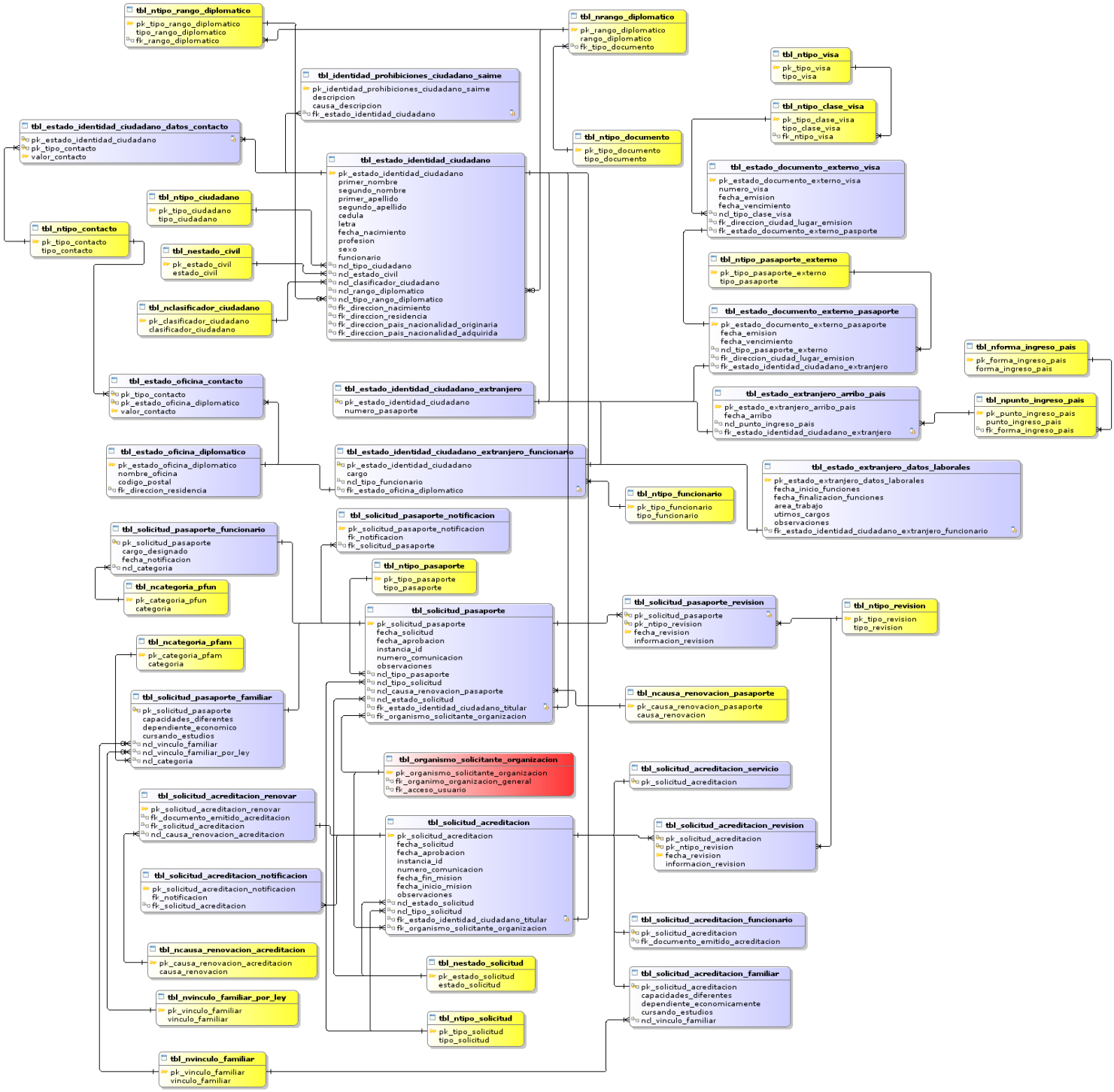
ANEXOS

Anexo 1: Operacionalización de las variables

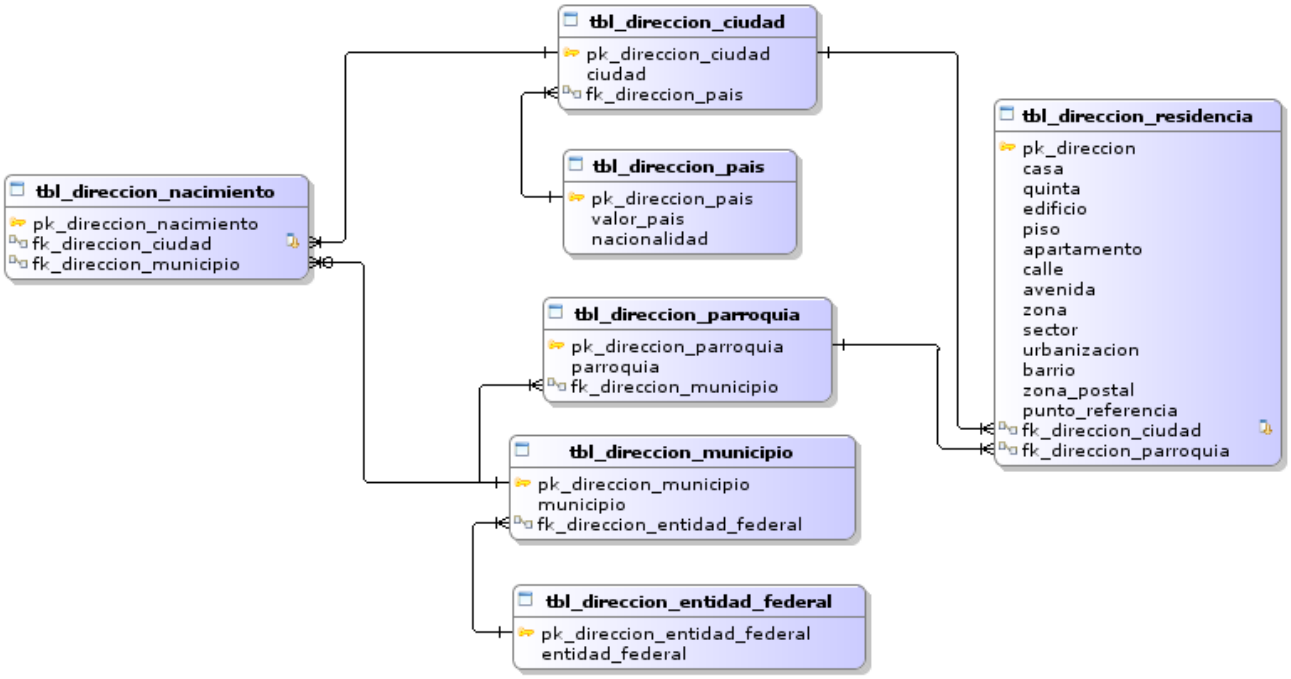
Variable conceptual	Dimensión	Indicadores	Unidad de medida
Herramienta de replicación	Datos	Replicación	Si
			No
	Rendimiento	Balanceo de carga	Si
			No
	Disponibilidad	Failover automático	Si
			No
Modelo de datos	Diseño	Normalizado	1FN
			2FN
			3FN
Sistema de emisión de pasaportes diplomáticos, de servicios y acreditaciones.	Servidores de datos	Sincronización	Si
			No
		Disponibilidad	Alto
			Medio
			Bajo

Tabla 6 Operacionalización de las variables

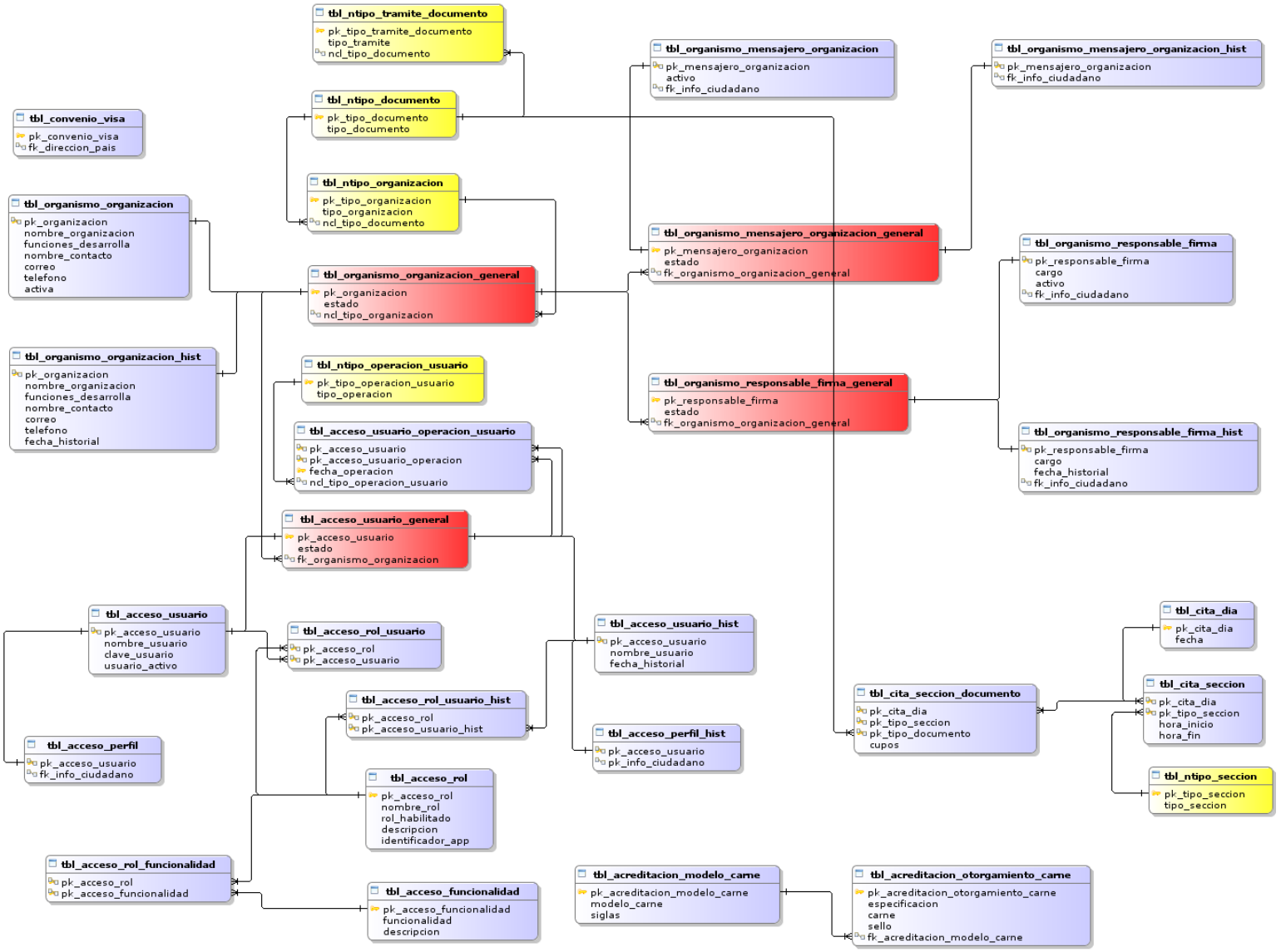
Anexo 2: Modelo de datos del sub-modelo Solicitud



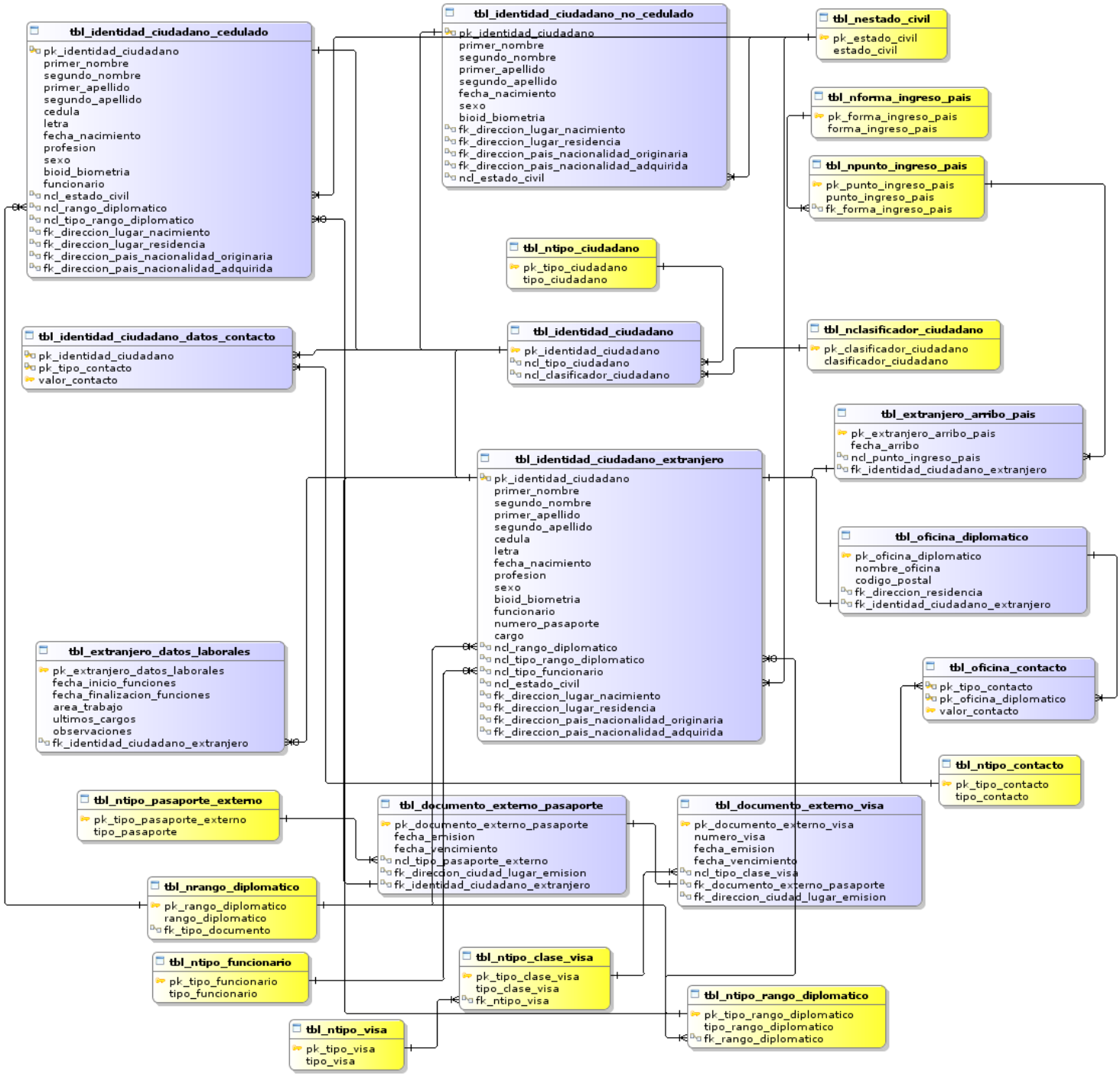
Anexo 3: Modelo de datos del sub-modelo Dirección



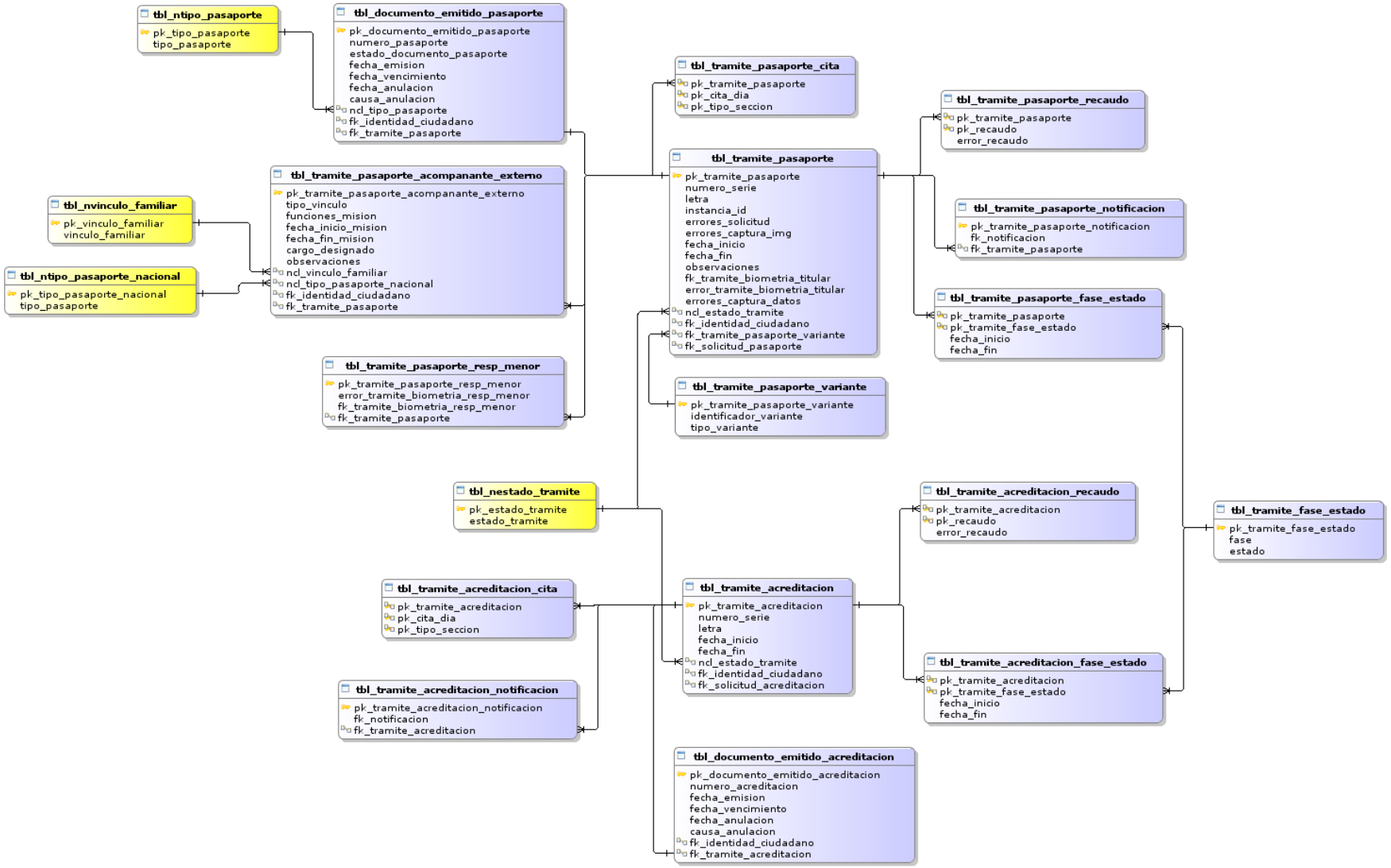
Anexo 4: Modelo de datos del sub-modelo Administración



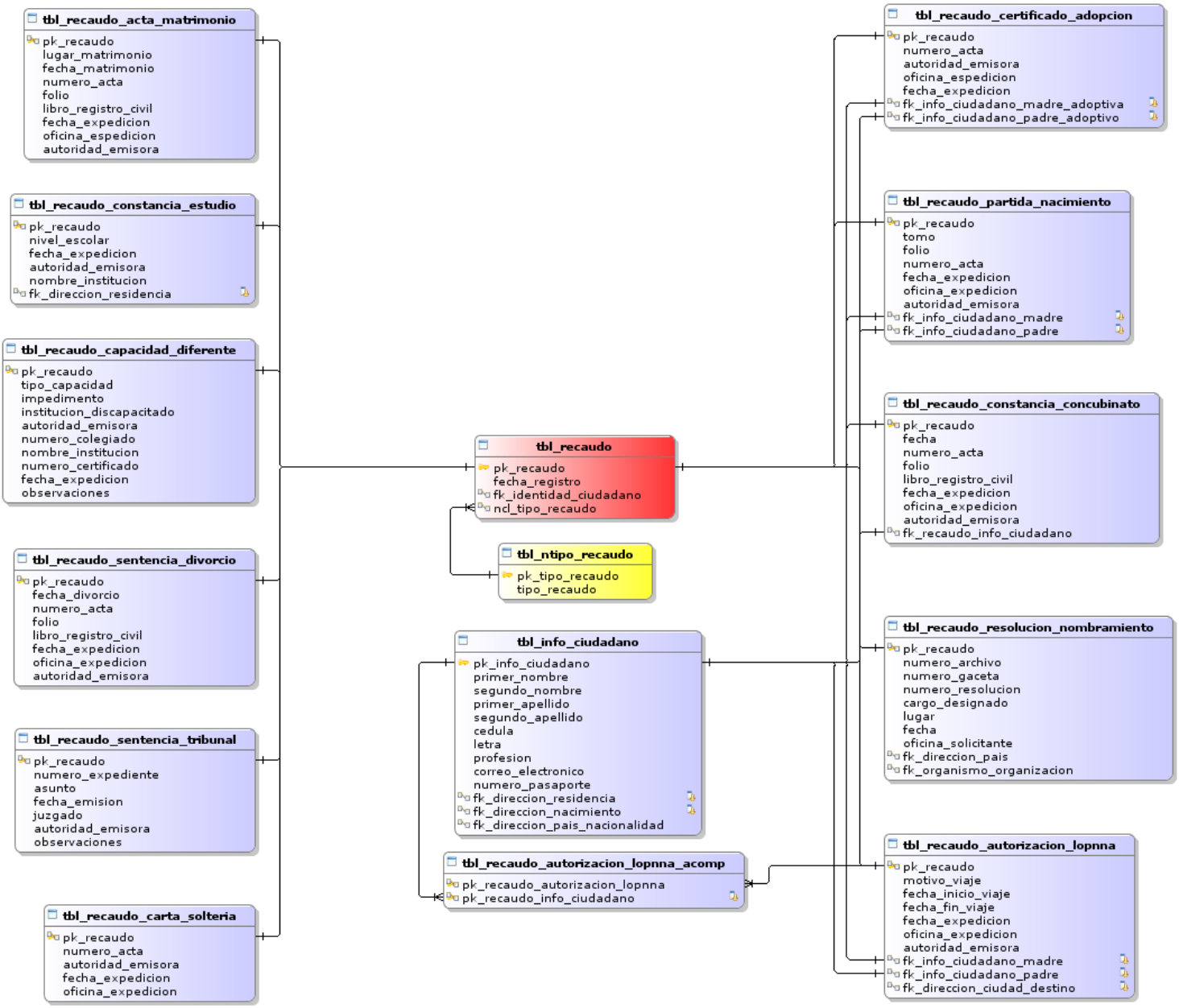
Anexo 5: Modelo de datos del sub-modelo Ciudadano



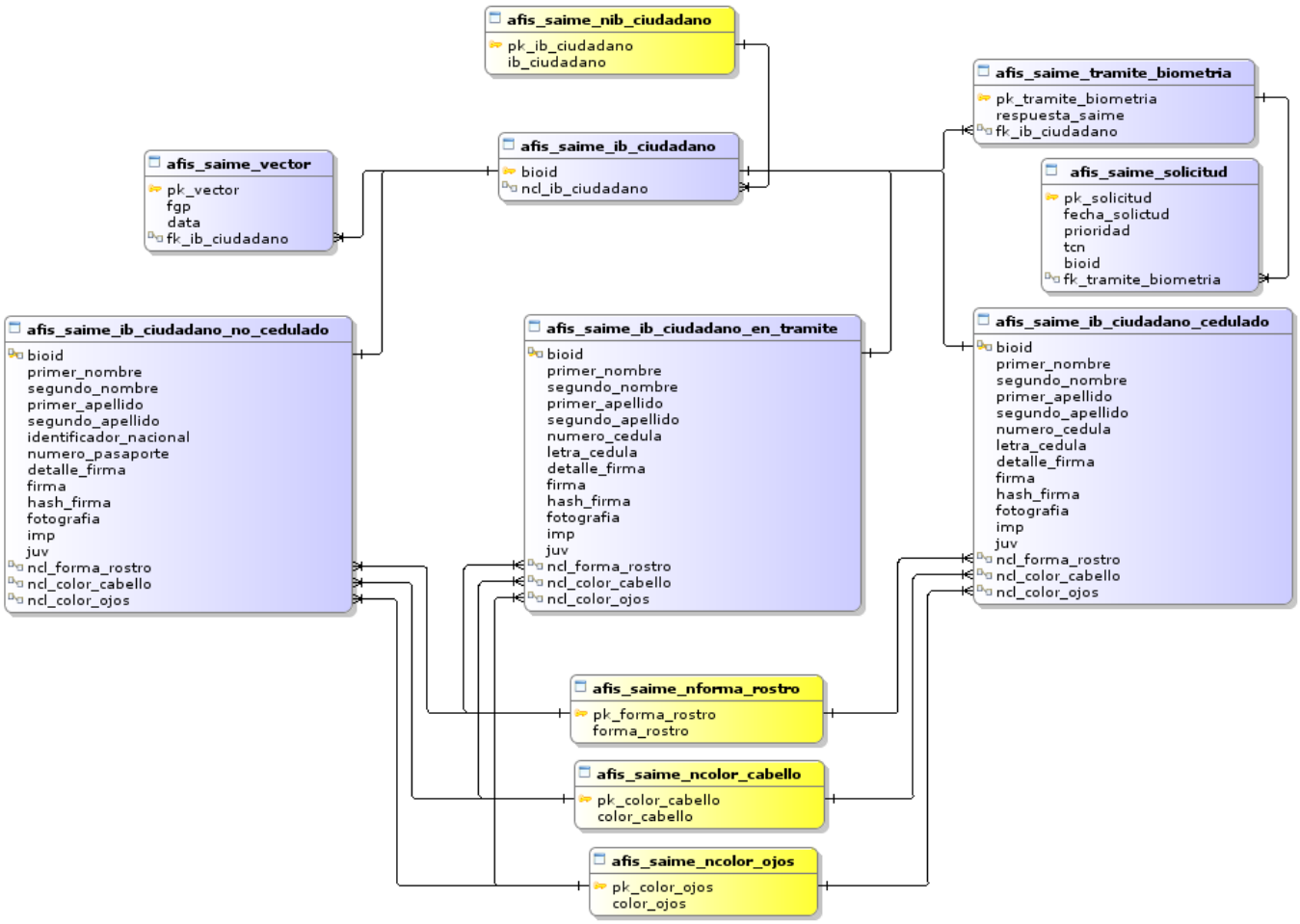
Anexo 6: Modelo de datos del sub-modelo Trámite



Anexo 7: Modelo de datos del sub-modelo Recaudo



Anexo 8: Modelo de datos del sub-modelo Biometría



Anexo 9: Dominios

Dominio	Tipo de datos	Descripción
dom_alfanumerico_100	VARCHAR(100)	Solo admite letras mayúsculas y minúsculas de la A-Z, dígitos del 0-9, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_alfanumerico_30	VARCHAR(30)	Solo admite letras mayúsculas y minúsculas de la A-Z, dígitos del 0-9, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_alfanumerico_50	VARCHAR(50)	Solo admite letras mayúsculas y minúsculas de la A-Z, dígitos del 0-9, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_cadena_100	VARCHAR(100)	Solo admite letras mayúsculas y minúsculas de la A-Z, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_cadena_30	VARCHAR(30)	Solo admite letras mayúsculas y minúsculas de la A-Z, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_cadena_50	VARCHAR(50)	Solo admite letras mayúsculas y minúsculas de la A-Z, vocales con tilde mayúsculas y minúsculas, carácter espacio y los caracteres (ü) y (Ü).
dom_cadena_binaria_20	VARCHAR(20)	Solo admite los dígitos 0 y 1.
dom_cadena_numerica_15	VARCHAR(15)	Solo admite números del 0 al 9.
dom_cod_alfanumerico	VARCHAR(20)	Solo admite letras mayúsculas de la A-Z y dígitos del 0-9.
dom_cod_pais	VARCHAR(3)	Solo admite letras mayúsculas de la A-Z.
dom_email	VARCHAR(100)	Solo admite direcciones de correo electrónico válidas.
dom_entero	INTEGER	Solo admite números enteros positivos.
dom_ip_dir	VARCHAR(15)	Solo admite direcciones IP válidas.
dom_letra_a	CHAR(1)	Solo admite el carácter A.
dom_letra_e	CHAR(1)	Solo admite el carácter E.
dom_letra_p	CHAR(1)	Solo admite el carácter P.
dom_letra_v	CHAR(1)	Solo admite el carácter V.
dom_letra_v_e	CHAR(1)	Solo admite los caracteres (V) y (E).
dom_sexo	CHAR(1)	Solo admite los caracteres (M) y (F).
dom_texto_100	VARCHAR(100)	Admite cualquier carácter.
dom_texto_30	VARCHAR(30)	Admite cualquier carácter.
dom_texto_500	VARCHAR(500)	Admite cualquier carácter.
dom_uuid	CHAR(32)	Solo admite letras minúsculas de la a-f y dígitos del 0-9.

Tabla 7. Dominios de la base de datos

Anexo 10: Instalación de PostgreSQL

El código fuente de PostgreSQL se encuentra disponible en el sitio oficial del gestor. Para su instalación, es necesario estar autenticado con el súper usuario del sistema root y ejecutar los siguientes pasos:

Paso 1:

Inicialmente deben ser instalados un conjunto de paquetes:

- GNU make es utilizado para determinar automáticamente qué piezas de un programa necesitan ser recompiladas y lanzar las órdenes necesarias para lograrlo.
- GCC para compilar el código fuente.
- tar para desempaquetar el fichero .tar.gz que contiene el código fuente del servidor.
- Librería GNU readline para el buffer de órdenes de psql.
- Librería de compresión zlib, utilizada para la compresión de backups generados a través de pg_dump y su restauración mediante pg_restore.

Existen otros paquetes que son opcionales y no son requeridos para una configuración por defecto, pero son necesarios cuando ciertas operaciones son habilitadas durante el proceso de instalación. Dentro de estos paquetes se encuentran:

- OpenSSL utilizado para el establecimiento de conexiones seguras utilizando SSL.

Para la instalación de estos paquetes se utilizará el gestor de paquetes yum a través del siguiente comando:

```
yum install make gcc gcc-c++ tar openssl-devel readline-devel zlib
```

Paso 2:

Copiar el fichero que contiene el código fuente hacia el directorio /usr/local/src mediante el siguiente comando:

```
cp [ubicación del fichero] /usr/local/src
```

Paso 3:

Moverse al directorio donde fue copiado el código fuente a través del comando:

```
cd /usr/local/src
```

Paso 4:

Descomprimir el paquete que contiene el código fuente:

```
tar -xzvf postgresql-8.3.14.tar.gz
```

Paso 5:

PostgreSQL por defecto se instala dentro del directorio /usr/local/pgsql, pero si se desea instalar en una ubicación diferente, es necesario crear el directorio donde se desee sea instalado. Para este caso se creará una carpeta nombrada *pgsql* dentro del directorio /media mediante el siguiente comando:

```
mkdir /media/pgsql
```

Paso 6:

Una vez creado el directorio donde PostgreSQL será instalado, es necesario ubicarse dentro del directorio que posee el código fuente que fue descomprimido durante el paso 4 y comenzar con la instalación mediante los siguientes comandos:

```
cd /usr/local/src/postgresql-8.3.14
./configure --prefix=/media/pgsql/ --bindir=/usr/bin/ --datadir=/usr/share/pgsql --sysconfdir=/usr/etc/  --libdir=/usr/lib/ --mandir=/usr/share/man/ --includedir=/usr/include/  --with-openssl
make
make install
```

En la siguiente tabla se describen los parámetros utilizados para la instalación de PostgreSQL:

Opciones	Descripción
--prefix=DIRECTORY	Con esta opción se indica el directorio donde PostgreSQL debe ser instalado.

--bindir=DIRECTORY	Especifica el directorio para los programas ejecutables como pg_dump, pg_restore, initdb, pg_ctl, entre otros.
--datadir=DIRECTORY	Especifica el directorio para archivos de datos de solo lectura, utilizado por los programas instalados. Los paquetes de instalación existentes en el módulo contrib serán ubicados en esta ubicación.
--sysconfdir=DIRECTORY	Directorio para varios ficheros de configuración.
--libdir=DIRECTORY	Directorio donde serán instaladas las librerías utilizadas por PostgreSQL.
--includedir=DIRECTORY	Directorio donde serán instalados los ficheros de cabecera en lenguaje C y C++.
--mandir=DIRECTORY	Directorio donde residirán las paginas man de ayuda que trae PostgreSQL.
--with-openssl	Instalación con soporte para conexiones utilizando SSL. Utilizar esta opción requiere tener instalado el paquete OpenSSL.

Tabla 8. Parámetros utilizados para la instalación de PostgreSQL

Paso 7:

Crear el usuario del sistema postgres. Con la opción -d se establece la carpeta de trabajo para este usuario cuando acceda al sistema, que será el directorio anteriormente creado en el paso 5. El comando es el siguiente:

```
useradd -d /media/pgsql/ postgres
```

Paso 8:

Crear el directorio donde se creará el clúster. El usuario postgres será el propietario de dicho directorio.

```
mkdir /media/pgsql/data
chown -R postgres:postgres /media/pgsql/
```

Paso 9:

Iniciar una sesión como usuario postgres y crear el clúster.

```
su - postgres
initdb -D /media/pgsql/data
```

Paso 10:

Una vez creado el clúster de PostgreSQL, es necesario iniciar el servidor, para lo que se utilizará un script desarrollado en bash nombrado postgresql. Este script da la posibilidad de iniciar, detener, recargar y reiniciar el servicio. Primeramente se debe copiar el script hacia el directorio /etc/init.d/, luego es necesario ubicarse dentro de este directorio y establecer a PostgreSQL como un servicio automático del sistema y finalmente se inicia el servicio. Los comandos para efectuar los procedimientos antes mencionados son los siguientes:

```
cp [fichero de inicio de PostgreSQL] /etc/init.d/
cd /etc/init.d/
chkconfig --add postgresql
chkconfig postgresql on
/etc/init.d/postgresql start ó service postgresql start
```

Paso 11:

Para la ejecución de procedimientos almacenados desarrollados en PL/pgSQL, es necesario tener instalado dicho lenguaje, el cual viene incluido dentro del código fuente del gestor. Para su instalación es necesario moverse hacia el directorio que contiene el código fuente e instalarlo. Una vez finalizado este proceso, hay que registrarse como usuario postgres y crear dicho lenguaje sobre template1, ya que todas las bases de datos que se creen heredaran de template1. Los comandos serían los siguientes:

```
cd /usr/local/src/postgresql-8.3.14/src/pl/plpgsql/
make
make install
su - postgres
createlang plpgsql template1
```

Anexo 11: Instalación de Pgpool-II

La versión de Pgpool-II utilizada será la 3.0.4 y solo será instalado en uno de los servidores que integran el clúster. Inicialmente deben ser instalados los paquetes gcc y gcc-c++ utilizados para la compilación de Pgpool-II. Para la instalación se seguirán los siguientes pasos como usuario del sistema root:

- Crear el directorio /opt/pgpool donde será instado Pgpool-II.

- Copiar el fichero pgpool-II-3.0.4.tar.gz dentro del directorio /usr/local/src.
- Moverse hacia ese directorio y desempaquetar el .tar.gz.
- Moverse hacia el directorio una vez descomprimido.
- Instalar Pgpool-II.

```
mkdir /opt/pgpool
cp [codigo fuente Pgpool-II] /usr/local/src
cd /usr/local/src
tar -xzvf pgpool-II-3.0.4.tar.gz
cd ./pgpool-II-3.0.4
./configure --prefix=/opt/pgpool  --with-openssl
make
make install
```

Luego de haber finalizado con la instalación, es recomendable instalar en ambos servidores el paquete pgpool_regclass que viene incluido dentro del código fuente. Este paquete es utilizado internamente por Pgpool en caso de que existan varios esquemas en la base de datos que contengan tablas con el mismo nombre, si este paquete no es instalado, el manejo de estas tablas puede causar problemas.

```
cd /usr/local/src/pgpool-II-3.0.4/sql/pgpool_regclass
make
make install
psql -U postgres -f pgpool_regclass.sql template1
```

Configuración de Pgpool-II

La configuración y puesta en marcha de Pgpool-II solo será realizada en el nodo en que fue instalado mediante los ficheros de configuración pcp.conf y pgpool.conf.

pcp.conf

Pgpool-II ofrece una interfaz de control a través de la cual el administrador puede conocer el estado de Pgpool-II y detener el proceso remotamente. El fichero pcp.conf es utilizado para la autenticación con esta interfaz. Durante el proceso de instalación, un fichero de ejemplo es creado dentro del directorio

/opt/pgpool/etc con el nombre pcp.conf.sample que puede ser utilizado para generar nuestro propio fichero pcp.conf. Para configurar dicho fichero inicialmente se deben ejecutar los siguientes comandos:

```
cd /opt/pgpool/etc
cp pcp.conf.sample pcp.conf
```

Una vez creado este fichero, es necesario generar la contraseña utilizada para autenticarse con la interfaz, utilizando un binario que viene incluido en la instalación de Pgpool-II cuya finalidad es cifrar una contraseña utilizando el método md5. Luego se edita el fichero de configuración y se copia una entrada en este fichero en el formato: USERID:MD5PASSWD. Para la generación de la contraseña se debe utilizar los siguientes comandos:

```
/opt/pgpool/bin/pg_md5 –p
password: <password>
34b339799d540a72bf1c408c0e68afdd
```

pgpool.conf

Las configuraciones de Pgpool-II se realizan en el fichero pgpool.conf. Al igual que pcp.conf, durante la instalación de Pgpool-II se crea un fichero pgpool.conf.sample mediante el cual se puede generar el fichero pgpool.conf a través de los siguientes comandos:

```
cd /opt/pgpool/etc
cp pgpool.conf.sample pgpool.conf
```

Todos los parámetros de configuración se encuentran explicados en el sitio oficial de Pgpool-II, por lo que solo serán analizados aquellos parámetros más importantes para la solución a implementar:

- Para habilitar la replicación en Pgpool-II es necesario establecer la propiedad *replication_mode* a true, igualmente sucede con la propiedad load_balance_mode, que permite balancear la carga entre los servidores.
- Para la replicación de objetos lob es necesario crear una tabla, la cual será utilizada por Pgpool-II para la generación de identificadores para dichos objetos, garantizando de esta forma que el id obtenido sea el mismo para todos los servidores que integran el clúster. La tabla se creará sobre

template1 para garantizar que todas las bases de datos que se creen, contengan dicha tabla. Se debe otorgar permiso total para dicha tabla.

- El parámetro lobj_lock_table especifica el nombre de una tabla, usada por Pgpool-II para controlar la replicación de objetos lob. Pgpool-II bloqueará la tabla especificada por lobj_lock_table y generará un id para el objeto lob, investigando en el catálogo del sistema pg_largeobject, seguidamente realizará una llamada a la función lo_create, con la cual se creará el objeto lob utilizando el id que fue generado previamente. Mediante este procedimiento se garantiza que Pgpool-II pueda obtener el mismo id de objeto lob en todos nodos de base de datos. La sintaxis para crear la tabla es la siguiente:

```
CREATE TABLE          pgpool_lobj_table();
GRANT ALL ON public. pgpool_lobj_table TO PUBLIC;
```

Autenticación md5 con Pgpool-II

Para habilitar la autenticación desde los clientes a la base datos a través de Pgpool-II utilizando el método de autenticación md5, es necesario realizar los siguientes pasos como usuario del sistema root:

1. Crear el usuario en el sistema con el cual se establecerán las conexiones a la base de datos a través de Pgpool-II.

```
useradd [usuario]
```

2. Autenticarse con el usuario creado anteriormente y ejecutar:

```
su - [usuario]
/opt/pgpool/bin/pg_md5 –md5auth [contraseña]
```

Esto creará el fichero pool_passwd, si no existe dentro del directorio /opt/pgpool/etc/, el cual contiene la entrada usuario: contraseña que será utilizada por Pgpool-II para realizar la autenticación.

3. Pgpool-II utiliza para la autenticación un fichero denominado pool_hba.conf. Durante el proceso de instalación un fichero de ejemplo es creado dentro del directorio /opt/pgpool/etc con el nombre pool_hba.conf.sample, que puede ser utilizado para generar el fichero pool_hba.conf. La

generación de dicho fichero se realiza de la misma manera que fueron generados los ficheros pcp.conf y pgpool.conf. Una vez creado este fichero, este debe ser editado y agregar los parámetros necesarios para la autenticación.

4. Habilitar la propiedad enable_pool_hba en el fichero de configuración de Pgpool-II.

Puesta en marcha de Pgpool-II

Dentro del código fuente de Pgpool-II se encuentran unos ficheros utilizados para iniciar el servicio. Se propone copiar dichos fichero hacia la carpeta donde fue instalado Pgpool-II, con el objetivo de tener una mayor organización. Una vez copiados dichos ficheros se puede iniciar el servicio mediante el siguiente comando:

```
/opt/pgpool/pgpool.init start
```

Anexo 12: Inicio de sesión SSH automática sin contraseña

Durante el proceso de restauración de un nodo, se hace uso del comando rsync para la copia de los datos entre los servidores, la información viaja de forma cifrada, posibilitándole una alta seguridad a la misma, pero es requisito que no se necesite la confirmación de una contraseña para comenzar la copia, los siguientes pasos son aplicables a los dos servidores a través del usuario del sistema root:

1- Cuando se instala PostgreSQL, se crea una cuenta de usuario postgres en el sistema, de la cual no se conoce su contraseña, por tal motivo, se necesita cambiar la contraseña para este usuario:

```
passwd postgres
```

Cuando se ejecuta el comando anterior, el sistema solicitará la nueva contraseña así como una confirmación de la misma.

2- Generar el par de llaves pública/privada para la conexión utilizando el algoritmo rsa. Es necesario autenticarse como usuario postgres, con el objetivo de que las claves generadas sean almacenadas dentro de su directorio home, que sería /media/pgsql/.

```
ssh-keygen -t rsa Generating public/private rsa key pair. Enter file in which to save the key (/home/user/.ssh/id_rsa):
# Presionamos enter
Enter passphrase (empty for no passphrase): # Presionamos enter
Enter same passphrase again: # Presionamos enter
Your identification has been saved in /media/pgsql/.ssh/id_rsa. Your public key has been saved in /var/lib/pgsql/.ssh/id_rsa.pub.
The key fingerprint is: 6f:c3:cb:50:e6:e9:90:f0:0f:68:d2:10:56:eb:1d:91 user@host
```

3- Copiar la llave generada hacia el servidor con el cual se establecerá una conexión segura:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub postgres@servidor
```

A partir de este momento se puede establecer una conexión SSH sin confirmación de contraseña.

Anexo 13: Script base-backup

```
#!/bin/sh
DATA=$1
RECOVERY_TARGET=$2
RECOVERY_DATA=$3
PGENGINE=/usr/bin
SSH=$PGENGINE/ssh
SCP=$PGENGINE/scp
RSYNC="$PGENGINE/rsync -arz --rsh=$SSH --delete"
PGCTL=$PGENGINE/pg_ctl
PSQL=$PGENGINE/psql
SED=/bin/sed
PGSQLPATH=/media/pgsql
PGCONF="postgresql.conf"
mkdir $PGSQLPATH/temporal
$SED -r -e "s/\s*archive_command\s*=.*/archive_command = '\/bin\/cp %p \/media\/pgsql\/pg_xlog_archive\/%f'/"
$RECOVERY_DATA/$PGCONF > $PGSQLPATH/temporal/$PGCONF
 chmod 644 $PGSQLPATH/temporal/$PGCONF
mv --force $PGSQLPATH/temporal/$PGCONF $RECOVERY_DATA/
rm -Rf $PGSQLPATH/temporal
$PGCTL reload -D $RECOVERY_DATA -s > /dev/null
psql -c "select pg_start_backup('pgpool-recovery')" postgres
echo "restore_command = '$RSYNC 10.13.4.181:/media/pgsql/pg_xlog_archive/%f %p'" >
/media/pgsql/data/recovery.conf
$RSYNC $DATA/base/ $RECOVERY_TARGET:$RECOVERY_DATA/base/
$RSYNC $DATA/global/ $RECOVERY_TARGET:$RECOVERY_DATA/global/
$RSYNC $DATA/pg_clog/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_clog/
$RSYNC $DATA/pg_multixact/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_multixact/
$RSYNC $DATA/pg_subtrans/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_subtrans/
$RSYNC $DATA/pg_tblspc/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_tblspc/
$RSYNC $DATA/pg_twophase/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_twophase/
```

```
$RSYNC $DATA/pg_xlog/ $RECOVERY_TARGET:$RECOVERY_DATA/pg_xlog/
psql -c "select pg_stop_backup()" postgres
$RSYNC $DATA/recovery.conf $RECOVERY_TARGET:$RECOVERY_DATA/
rm $DATA/recovery.conf
exit 0
```

Anexo 14: Script pgpool_recovery_pitr

```
#! /bin/sh
# Online recovery 2nd stage script#
datadir=$1              # master dabatase cluster
DEST=$2                 # hostname of the DB node to be recovered
DESTDIR=$3              # database cluster of the DB node to be recovered
archdir=/media/pgsql/pg_xlog_archive   # archive log directory
# Force to flush current value of sequences to xlog
psql -U postgres -t -c 'SELECT datname FROM pg_database WHERE NOT datistemplate AND datallowconn' template1|
while read i
do
  if [ "$i" != "" ];then
    psql -U postgres -c "SELECT setval(oid, nextval(oid)) FROM pg_class WHERE relkind = 'S'" $i
  fi
done
psql -U postgres -c "SELECT pgpool_switch_xlog('$archdir')" template1
PGENGINE=/usr/bin
PGCTL=$PGENGINE/pg_ctl
SED=/bin/sed
PSQL=$PGENGINE/psql
PGSQLPATH=/media/pgsql
PGCONF="postgresql.conf"
mkdir $PGSQLPATH/temporal
```

```
$SED -r -e "s/\s*archive_command\s*=.*/archive_command = 'exit 0'/" $DESTDIR/$PGCONF > $PGSQLPATH/temporal/$PGCONF
 chmod 644 $PGSQLPATH/temporal/$PGCONF
 mv --force $PGSQLPATH/temporal/$PGCONF $DESTDIR/
rm -Rf $PGSQLPATH/temporal
$PGCTL reload -D $DESTDIR -s > /dev/null
```

Anexo 15: Script pgpool_remote_start

```
#! /bin/sh
DEST=$1
DESTDIR=$2
PGCTL=/usr/bin/pg_ctl
ssh -T $DEST $PGCTL -w -D $DESTDIR start 2>/dev/null 1>/dev/null < /dev/null &
```

Anexo 16: Script de prueba utilizado por pgbench

```
SELECT count(*) FROM tbl_acceso_usuario INNER JOIN tbl_organismo_solicitante_organizacion ON
tbl_acceso_usuario.pk_acceso_usuario = tbl_organismo_solicitante_organizacion.fk_acceso_usuario INNER JOIN
tbl_solicitud_pasaporte ON tbl_organismo_solicitante_organizacion.pk_organismo_solicitante_organizacion =
tbl_solicitud_pasaporte.fk_organismo_solicitante_organizacion INNER JOIN tbl_solicitud_pasaporte_familiar ON
tbl_solicitud_pasaporte_familiar.pk_solicitud_pasaporte = tbl_solicitud_pasaporte.pk_solicitud_pasaporte INNER
JOIN tbl_nestado_solicitud ON tbl_solicitud_pasaporte.ncl_estado_solicitud =
tbl_nestado_solicitud.pk_estado_solicitud INNER JOIN tbl_estado_identidad_ciudadano ON
tbl_solicitud_pasaporte.fk_estado_identidad_ciudadano_titular =
tbl_estado_identidad_ciudadano.pk_estado_identidad_ciudadano LEFT OUTER JOIN
tbl_estado_identidad_ciudadano_extranjero ON tbl_estado_identidad_ciudadano.pk_estado_identidad_ciudadano =
tbl_estado_identidad_ciudadano_extranjero.pk_estado_identidad_ciudadano WHERE
tbl_acceso_usuario.nombre_usuario = 'solicitante' AND (tbl_nestado_solicitud.pk_estado_solicitud = 1 OR
tbl_nestado_solicitud.pk_estado_solicitud = 2);

SELECT * FROM tbl_ncausa_renovacion_acreditacion ORDER BY
tbl_ncausa_renovacion_acreditacion.causa_renovacion ASC;

SELECT * FROM tbl_ntipo_pasaporte;
```

```
SELECT tbl_direccion_ciudad.pk_direccion_ciudad, tbl_direccion_ciudad.ciudad FROM tbl_direccion_ciudad
WHERE tbl_direccion_ciudad.fk_direccion_pais = 'VEN';
```