Universidad de las Ciencias Informáticas

Facultad 4

Título: Sistema de réplica para bases de datos distribuidas en PostgreSQL.

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

Autor: Norge Fajardo Vega.

Tutor: Ing. Alexis Palma Espinosa

Junio, 2007

La utopía está en el horizonte. Camino dos pasos, ella se aleja dos pasos y el horizonte se corre diez pasos más allá. ¿Entonces para que sirve la utopía? Para eso, sirve para caminar.

Eduardo Galeano

Resumen

Las Fuerzas Armadas Revolucionarias (FAR) es una institución que desempeña un papel determinante en la defensa del país, siendo el Ministerio (MINFAR), su entidad rectora.

Entre sus principales objetivos se encuentra la total informatización del organismo, el cual no ha escapado como muchos otros en el país a la avalancha tecnológica de estos tiempos con el fin de elevar la eficiencia en los numerosos procesos que se realizan en todas sus entidades, por lo que el flujo de información es muy grande. Debido a la organización jerárquica de las mismas en toda la Isla, la información que sustenta el desempeño de cada una de las actividades debe tener un alto nivel de actualización en todas las estructuras. La solución vigente a este problema no satisface la necesidad de contar con la información cuando se deba tomar una decisión en un momento dado, fundamentalmente porque depende del factor humano. La información actualmente puede demorarse bastante tiempo en llegar a los diferentes niveles según la necesidad, pudiendo incluso no ser la adecuada.

El presente trabajo de diploma propone solucionar este problema a través de la implementación de un sistema de gestión de réplica de datos, tanto de espejo como de fragmentos según sea necesario, que permita la actualización de la información en los diferentes niveles íntegra y eficientemente.

Palabras claves: réplica, sistema gestor de base de datos, Slony.

Índice

Introducción

Las innovaciones tecnológicas producidas en los últimos años han promovido un cambio en la forma de observar los sistemas de información y en general las aplicaciones computacionales.

Aún cuando es posible que un usuario común no perciba los desarrollos relevantes de nuevos productos, para las aplicaciones existe una demanda permanente por mayor funcionalidad, mayor número de servicios, más flexibilidad y mejor rendimiento. Así, al diseñar un nuevo sistema de información o al prolongar la vida de uno ya existente, se debe buscar siempre formas para enlazar las soluciones ofrecidas por la tecnología disponible a las necesidades de las aplicaciones de los usuarios.

Un área en la cual las soluciones están integrando tecnología con nuevas arquitecturas o formas de hacer las cosas es, sin lugar a dudas, el área de los sistemas distribuidos de información. Ellos se refieren al manejo de datos que se encuentran en muchos sitios interconectados a través de una red de comunicaciones. Un caso específico de estos sistemas distribuidos es lo que se conoce como bases de datos distribuidas. [1]

Un sistema de bases de datos distribuidas (BDD) se compone de un conjunto de localidades, cada una de las cuales mantiene un sistema de base de datos local. Cada localidad puede procesar transacciones locales, es decir, aquellas que sólo acceden a datos que residen en esa localidad. Además una localidad puede participar en la ejecución de transacciones globales, es decir, aquellas que acceden a datos de varias localidades.

Las localidades pueden estar dispersas, ya sea por un área geográfica extensa (a lo largo de un país), llamadas redes de larga distancia; o en un área reducida (en un mismo edificio), llamadas redes de área local.

Cabe destacar que en un sistema de BDD tiene las siguientes características:

- ✓ Cada sitio es un sistema de base de datos completo por derecho propio, es decir, cada sitio tiene sus propias bases de datos reales, sus propios usuarios locales, su propio Sistema Gestor de Bases de Datos (SGBD) local y software de administración de transacciones y su propio administrador de comunicación de datos local. De esta forma, un usuario determinado puede realizar operaciones sobre los datos desde su propio sitio local, tal como si ese sitio no participa en el sistema distribuido.
- ✓ Los sitios han acordado trabajar juntos, a fin de que un usuario cualquiera pueda acceder a los datos desde cualquier lugar de la red, exactamente como si los datos estuvieran guardado en el propio sitio del usuario. [2]

En consecuencia, la llamada “base de datos distribuida” es en realidad una especie de objeto virtual, cuyas partes componentes se almacenan físicamente en varias bases de datos “reales” distintas, ubicadas en diferentes sitios. De hecho, es la unión lógica de esas bases de datos. [3]
La principal ventaja de los sistemas distribuidos es la capacidad de compartir y acceder a la información de una forma fiable y eficaz. Para sincronizar la misma, garantizando que ésta sea consistente, aumentando la eficacia, disponibilidad y tolerancia a fallos de los servicios prestados por los sistemas distribuidos como las BDD se usa la técnica de replicación.
Los sistemas distribuidos de información son ampliamente difundidos en las diversas ramas de la economía, dando solución a los más disímiles problemas de cada institución en el país.
El Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR) es una de las instituciones preocupada y ocupada en el desarrollo de soluciones informáticas aplicadas a su entorno. Dada la necesidad de compartimentación y vitalidad de la información, la totalidad de las aplicaciones informáticas son sistemas distribuidos a lo largo del país (sucursales), interconectados por una red con estructura de árbol, siendo el Ministerio el nodo primario de la jerarquía, como se muestra en la Fig. 1.1.

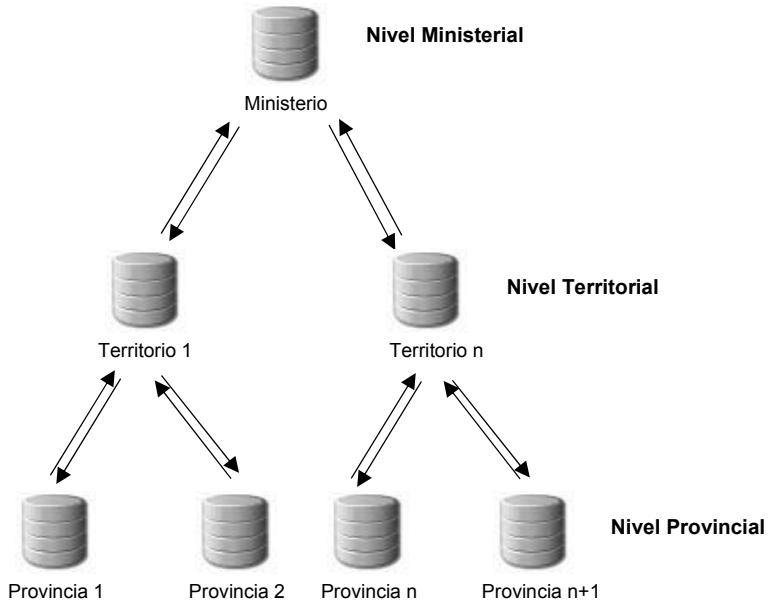


Figura 1 Distribución de datos en las FAR

Cada sucursal maneja sus propias bases de datos diseñadas en PostgreSQL (sistema gestor de base de datos), recibiendo y transmitiendo información (como se indica en la Figura 1), hacia uno y otro nivel de la jerarquía respectivamente.
Se tienen tres niveles principales en los cuales se concentra la información de las sucursales subyacentes, las que variaran en cantidad y tipo, según las necesidades del Ministerio. Un

primer nivel, denominado nodo Ministerial, un segundo nivel, denominado nodo Territorial y por último, un tercer nivel, denominado nodo Provincial.

Atendiendo a la situación anterior se debe garantizar que la información de cada sucursal se transmita al nivel superior cumpliendo con una serie de normas y reglas que garanticen la compartimentación de la misma, así como el cumplimiento de los plazos requeridos.

En estos momentos el mecanismo de actualización de la información a los nodos de nivel superior que se emplea no satisface totalmente las necesidades actuales del Ministerio, específicamente debido a los tiempos prolongados entre una actualización y la siguiente pues depende del factor humano, con el consiguiente incumplimiento de los plazos requeridos, provocando la demora de la gestión de la información, asimismo dificulta la toma eficiente y rápida de decisiones.

Luego de un análisis del mismo y tomando en cuenta la situación actual, surge el siguiente **problema**: ¿Cómo garantizar que la información distribuida en las bases de datos de las FAR se transmita a cada uno de los niveles inmediatos, tanto inferiores como superiores cumpliendo las normas establecidas y los plazos requeridos?

El presente trabajo se propone dar solución al problema existente mediante el desarrollo de un sistema de gestión de réplicas, que permita la actualización de la información en los diferentes niveles íntegra y eficientemente.

Por tanto el **objeto de estudio** es: el proceso de réplica en bases de datos distribuidas.

Delimitando así el **campo de acción**, siendo este, la aplicación del proceso de réplica en las bases de datos distribuidas de las FAR.

La investigación se sustenta en la siguiente **hipótesis:** si se tiene un sistema que permita la configuración, gestión y mantenimiento de una política de réplica de datos basada en reglas dinámicamente configurables, será posible transmitir la información cumpliendo las normas establecidas y los plazos requeridos entre los servidores de datos de las FAR.

El **objetivo general** de la tesis es: Elaborar un sistema informático que permita la configuración, gestión y mantenimiento de una política de réplica de datos basada en reglas dinámicamente configurables que garantice la transmisión de la información entre los servidores de datos de las FAR.

De él se derivan los siguientes **objetivos específicos**:

1. Caracterizar las reglas y procesos que intervienen en los flujos informativos en las FAR.
2. Diseñar el sistema que permita la configuración, gestión y mantenimiento de una política de replicación de datos entre los servidores de datos de las FAR.

3. Implementar el sistema informático diseñado.

Para la demostración de la hipótesis se proponen las siguientes **tareas**:

1. Realizar una búsqueda bibliográfica sobre el proceso de réplica entre bases de datos distribuidas, así como de los sistemas informáticos existentes para la ayuda al mismo.
2. Estudiar los procesos que intervienen en los flujos informativos en las FAR.
3. Estudiar las últimas tendencias y tecnologías que a nivel mundial se utilizan para construir una aplicación como la que se pretende desarrollar.
4. Seleccionar el proceso de desarrollo de software a utilizar.
5. Seleccionar las herramientas a utilizar para el desarrollo de la aplicación.
6. Realizar el diseño del sistema utilizando el proceso de desarrollo de software seleccionado.
7. Desarrollar una aplicación que facilite la agilidad del flujo informativo mediante una política de réplica de datos en las FAR.

Para realizar las tareas antes propuestas se utilizaron los métodos de nivel teórico y empírico. Dentro de los primeros se emplearon el Hipotético-Deductivo, de Análisis y Síntesis, como parte de los segundos la Entrevista y la Modelación.

El presente trabajo se encuentra dividido en cuatro capítulos. En el primero se tratan aquellos temas que constituyen la fundamentación teórica de la investigación a realizar. Incluye un estado del arte del tema tratado en el ámbito tanto nacional como internacional, de las tendencias, técnicas, tecnologías, metodologías y software existentes que de una forma u otra están relacionados con el tema que se aborda, profundizando en ellos.

El segundo capítulo se describe a profundidad el objeto de estudio, así como la modelación de los procesos de negocio y se da una descripción de la solución propuesta, definiéndose los requisitos que debe cumplir la misma.

En el tercer capítulo describe a profundidad la construcción de la propuesta de solución mediante los diversos artefactos que especifica el proceso de software utilizado.

Por último, el cuarto capítulo constituye la implementación y la prueba realizadas a la solución propuesta.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En el presente capítulo se brinda una visión general de los aspectos relacionados con la distribución y manejo de la información, y más específicamente, la aplicación de estos conceptos en las bases de datos de las Fuerzas Armadas Revolucionarias (FAR). Para ello se da una descripción de los principales conceptos asociados al dominio del problema.
Además constituye un acercamiento a las tendencias y tecnologías sobre las que se apoyará la propuesta.

1.2 Bases de datos distribuidas

Las Bases de Datos Distribuidas (BDD) surgidas como ya se ha comentado anteriormente con los avances en la tecnología de micro-procesadores y el amplio uso de redes de computadoras que impulsaron la investigación sobre la posibilidad de distribuir los datos en la red usando diferentes criterios como, por ejemplo, las frecuencias de utilización de los datos, son un avance distintivo en el almacenamiento de los datos por las ventajas que brindan.
Su objetivo principal es mejorar la eficiencia de su desempeño distribuyendo físicamente los datos de acuerdo a los requerimientos de uso y las capacidades computacionales que disponen los diferentes usuarios, pero manteniendo la visión del sistema como un solo componente. Para lograr este objetivo se desarrollaron diferentes algoritmos de distribución de datos, como la fragmentación y la replicación. [4]

1.2.1 Fragmentación de datos

El problema de fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red. Para ello se implementaron las técnicas de fragmentación de dos tipos: *fragmentación horizontal:* donde se distribuyen los registros de las tablas de la base de datos entre diferentes nodos de la red, como se observa debajo en las tablas 1.1, 1.2 y 1.3.
Una tabla T (País) se divide en subconjuntos, T1 (Provincia 3), T2 (Provincia 4),...Tn (Provincia n). Los fragmentos se definen a través de una operación de selección y su reconstrucción se realizará con una operación de unión de los fragmentos componentes.

idpers	numid	nombre	papell	idprovinc
1311911	91012938529	MARIO	RODRIGUEZ	3
1311912	1062138527	JOSE	VALDES	3
1311913	91050938505	OSMANY	HERNANDEZ	3
1411911	91081839140	NILO	NEGRIN	4
1411912	91012438561	LESKIER	ABELLA	4
1411913	91121838509	ARLEY	SIMON	4

Tabla 1.1 Base de Datos País

idpers	numid	nombre	papell	idprovinc
1311911	91012938529	MARIO	RODRIGUEZ	3
1311912	1062138527	JOSE	VALDES	3
1311913	91050938505	OSMANY	HERNANDEZ	3

Tabla 1.2 BD Provincia 3

idpers	sumid	nombre	papell	idprovinc
1411911	91081839140	NILO	NEGRIN	4
1411912	91012438561	LESKIER	ABELLA	4
1411913	91121838509	ARLEY	SIMON	4

Tabla 1.3 BD Provincia 4

fragmentación vertical: donde se distribuyen los atributos de acuerdo a la frecuencia de su uso, como se observa debajo en las tablas 1.4, 1.5 y 1.6.

idpers	numid	fnac	nombre	idestciv
1311911	91012938529	1991-01-29	MARIO	1
1311912	1062138527	1991-06-21	JOSE	0
1311913	91050938505	1991-05-09	OSMANY	0
1411911	91081839140	1991-08-18	NILO	1
1411912	91012438561	1991-01-24	LESKIER	1
1411913	91121838509	1991-12-18	ARLEY	0

Tabla 1.4 Base de Datos País

idpers	numid	fnac	Nombre
1311911	91012938529	1991-01-29	MARIO
1311912	1062138527	1991-06-21	JOSE
1311913	91050938505	1991-05-09	OSMANY
1411911	91081839140	1991-08-18	NILO
1411912	91012438561	1991-01-24	LESKIER
1411913	91121838509	1991-12-18	ARLEY

Tabla 1.5 BD Provincia 5

idpers	numid	nombre	idestciv
1311911	91012938529	MARIO	1
1311912	1062138527	JOSE	0
1311913	91050938505	OSMANY	0
1411911	91081839140	NILO	1
1411912	91012438561	LESKIER	1
1411913	91121838509	ARLEY	0

Tabla 1.6 BD Provincia 6

Una tabla T (País) se divide en subconjuntos, T1 (Provincia 5), T2 (Provincia 6),...Tn. Los fragmentos se definen a través de una operación de proyección.

Cada fragmento debe incluir la llave primaria de la tabla. Su reconstrucción se realizará con una operación de unión de los fragmentos componentes. [5]

En este contexto, una fragmentación "óptima" es aquella que produce un esquema de fragmentación que minimiza el tiempo de ejecución de las consultas de usuario.

Se puede también usar la fragmentación mixta que incluye una o varias secuencias de aplicación de las fragmentaciones anteriormente mencionadas.

1.3 Réplica de datos

La réplica de datos es una técnica que permite copiar y distribuir idénticamente las tablas de una base de datos en múltiples bases de datos ubicadas en diferentes nodos de la red. La replicación asegura que los datos correctos estén siempre disponibles en el momento y en el lugar necesario.

Posee las siguientes características:

- ✓ *Efectividad*: depende de la forma en la que los datos sean distribuidos y almacenados. A mayor efectividad, mayor será la disponibilidad de datos para ejecutar procesos paralelos.
- ✓ *Alta Disponibilidad*: es la razón de tiempo prudente en la que un servicio puede ser accedido. En el mejor de los casos puede ser de un 100%, a pesar de los fallos que se puedan presentar en el servidor, ya que debe existir un servidor adicional que posea alguna técnica de replicación que lo pueda suplantar en caso de ser necesario.
- ✓ *Tolerancia a fallos*: garantiza un comportamiento correcto, donde efectivamente pueden existir un número finito de fallos y tipos de fallos.
- ✓ *Coordinación*: cada una de las partes que conforman la base de datos distribuida, acuerda un consenso para realizar las invocaciones de los servicios a los objetos, que al final de la transacción debe realizarse tal y como fue solicitada, para lo cual debe utilizar algún tipo de ordenamiento. [6]

1.3.1 Entornos de Réplica

Los tipos de entornos de réplica son los siguientes:

Maestro-Esclavo (master-slave): o de solo lectura, permite a un solo maestro recibir consultas de lectura/escritura, mientras los esclavos solo pueden aceptar consultas de lectura.

Multi-Maestro (multi-master): también llamada par-a-par o la réplica de camino de n, permite múltiples sitios, actuando como pares iguales. Cada sitio en un ambiente de réplica de multi-maestro es un sitio de maestro, y cada sitio se comunica con otros sitios maestros. Esta capacidad tiene también un severo impacto en el desempeño debido a la necesidad de sincronizar los cambios entre los servidores.

Este tipo de entorno puede ser usado para mantener sitios recuperables ante posibles desastres o caídas, así como para proveer sistemas con alta disponibilidad y para balancear la carga de consultas a través de las distintas ubicaciones.

Existen dos modelos de distribución de datos esencialmente aplicados a cada uno de los entornos antes vistos:
Asincrónica: a menudo llamada almacena-y-reenvía, captura cualquier cambio local, los almacena en una cola, y, a intervalos regulares, propaga y aplica estos cambios en sitios remotos. Con esta forma de réplica, hay un período de tiempo antes de que todos los sitios alcancen la convergencia de datos.
Sincrónica: también conocida como la réplica en tiempo real, aplica cualquier cambio o ejecuta cualquier procedimiento reproducido en todos los sitios que participan en el ambiente de réplica como parte de una sola transacción. Si el procedimiento falla en cualquier sitio, entonces la transacción entera se anula. La réplica sincrónica asegura la consistencia de datos en todos los sitios en tiempo real.

Para cada modelo de distribución de datos existen varias tecnologías para llevar a cabo la replicación, cada una a su vez con las consiguientes limitaciones, como se muestra en la Fig. 3.

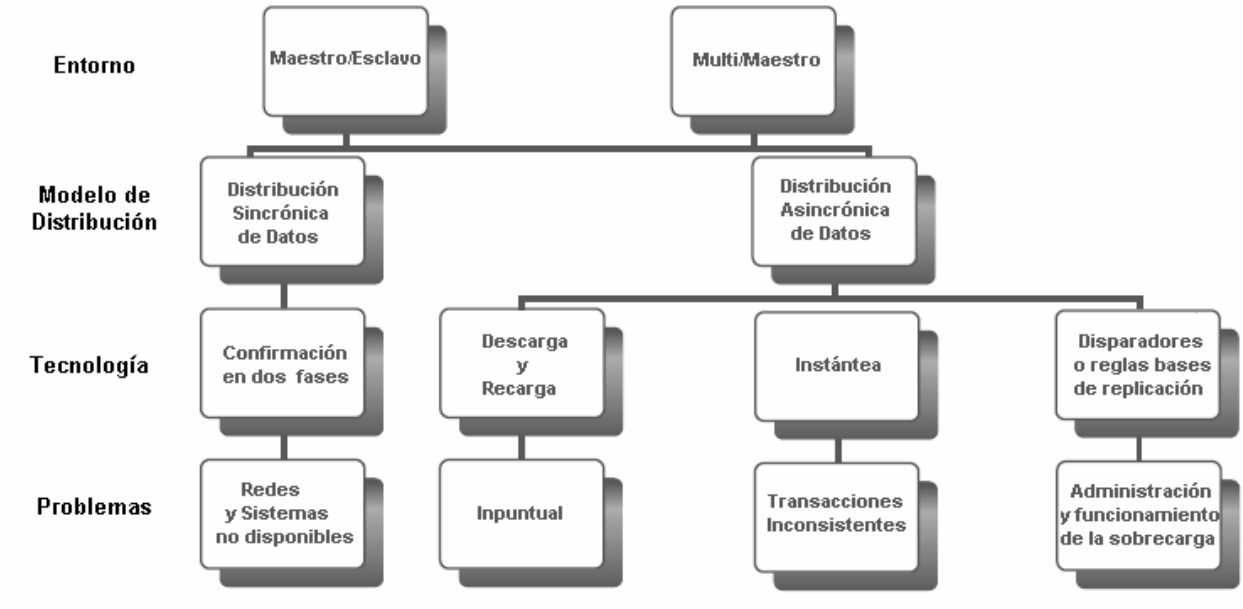


Figura 1.1 Esquema de réplica de datos

1.3.2 Técnicas de replicación

Sincrónica

Confirmación en dos fases (Two Phase Commit)

Esta tecnología surge a mediados de los años 80. Permite la sincronización de datos distribuidos. Cada transacción solamente es aceptada si todos los sistemas implicados en la réplica están conectados y listos para recibirla, si al menos uno falla, todo el proceso es anulado, como se muestra en la Fig. 4.

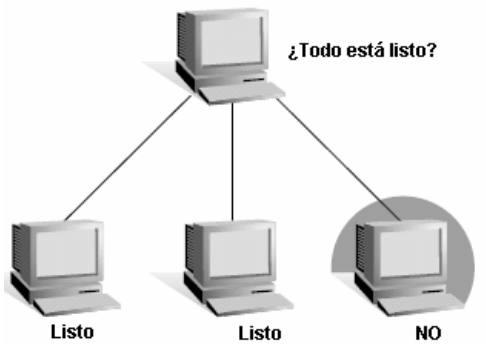


Figura 1.2 Confirmación en dos fases

Asincrónica

Descarga y Recarga (Dump and Reload)

Consiste en hacer un volcado de los datos, copiar la salva para un dispositivo de almacenamiento para luego distribuir la salva por los demás servidores. Esta técnica presenta el inconveniente de que en la mayoría de las ocasiones se consultaban datos que tenían semanas de desactualización, además de que el proceso se realizaba de forma manual.

Instantánea (Snapshot)

Consiste en hacer una instantánea de una tabla en un momento dado y esta “foto” es la que se replica en las demás bases de datos. Aunque mucho más avanzado que la técnica de Descarga y Recarga esta presenta el inconveniente de que las tablas esclavas son tablas de solo lectura. Se recomienda utilizar cuando: la mayoría de los datos no cambian con frecuencia; se replican pequeñas cantidades de datos; los sitios con frecuencia están desconectados y es aceptable un

periodo de latencia largo (la cantidad de tiempo que transcurre entre la actualización de los datos en un sitio y en otro).

Disparadores (Triggers)

La función del disparador es ejecutar una acción cuando ocurre un evento en la base de datos, los eventos pueden ser de inserción, actualización o eliminación.
Conjuntamente con las instantáneas, los disparadores son otro de los mecanismos asincrónicos que proporciona la base de datos como una manera de replicación de datos. [7]
Para mayor comodidad serán llamados triggers, ya que es el término por el cuál son conocidos.

1.3.3 ¿Por qué usar replicación?

Las instituciones utilizan la replicación en sus aplicaciones por varias razones, las cuáles pueden ser categorizadas de la siguiente forma:

- ✓ Distribución de datos a otras ubicaciones
- ✓ Consolidación de datos desde otras ubicaciones
- ✓ Intercambio bidireccional de datos con otras ubicaciones
- ✓ Alguna variantes o combinaciones de los anteriores

1.3.3.1 Distribución de datos a otras ubicaciones

La distribución de datos involucra el movimiento de todos o un subconjunto de datos de una o más ubicaciones.
La distribución es utilizada para proveer datos a aplicaciones desarrolladas en plataformas similares o diferentes. Esto puede ser tan simple como mantener una copia de los datos de producción en otro sistema similar, así como requerir una transformación compleja para adaptarse a los requerimientos de una nueva aplicación. Los datos que se copien pueden necesitar ser filtrados o transformados para la nueva aplicación. La distribución también puede ser usada para proveer coexistencia de dos sistemas, en la instancia de migración de uno al otro.

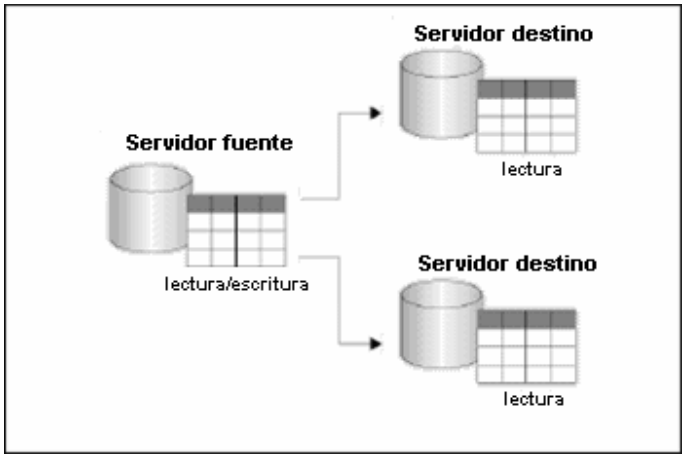


Figura 1.3 Replicación desde un servidor fuente hacia dos servidores destinos.

Los servidores destino podrían mantener un subconjunto diferente de los datos de acuerdo a las necesidades propias.

1.3.3.2 Consolidación de datos desde otras ubicaciones

Una institución puede tener sus datos en distintos sistemas distribuidos. A modo de ejemplo, una empresa de ventas puede tener sus datos en cada local, una empresa de manufactura los puede tener en cada planta de procesamiento, así como una compañía aseguradora podría tenerlos en cada una de sus oficinas o llegar incluso a cada computador personal de sus corredores. La replicación puede copiar los cambios de cada sitio distribuido al sitio central, para analizar, hacer reportes, o para el procesamiento central de los datos.

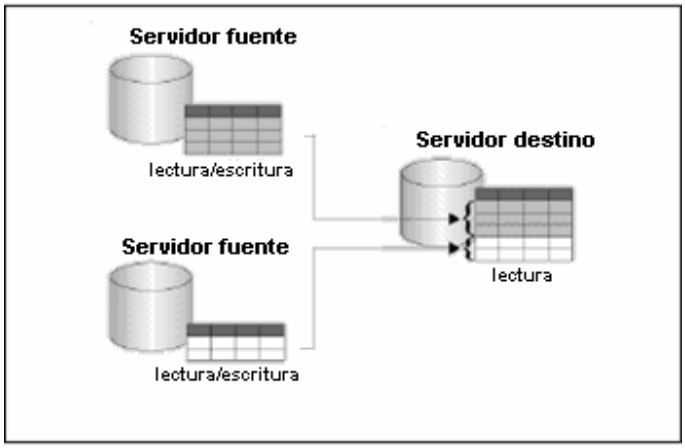


Figura 1.4 Replicación desde dos servidores fuentes, hacia un servidor destino.

La estructura de los datos en cada fuente es la misma. Los datos del destino son la unión de los datos de las distintas fuentes.

1.3.3.3 Intercambio bidireccional de datos con otras ubicaciones

Si los datos se pueden modificar en múltiples ubicaciones, entonces la replicación debe procesar los cambios realizados en cada uno de los sitios de forma coordinada. Uno de los servidores, es visto como el servidor maestro, quien se encarga de distribuir los cambios a todos los sitios. Los cambios realizados en los destinos fluyen hacia los otros sitios a través del servidor maestro.

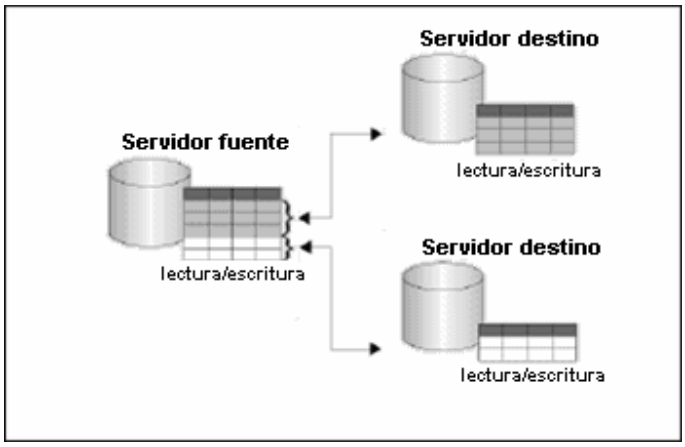


Figura 1.5 Réplica bidireccional, en la que tenemos un servidor maestro designado.

Otro tipo de replicación bidireccional, es aquella que no tiene designada un servidor maestro. Cada ubicación copia los cambios desde todos los otros sitios directamente. Esto habitualmente sucede en los entornos “multi-maestro” o “par-a-par”.

1.3.3.4 Otros requerimientos

Algunas aplicaciones podrían no requerir una copia completa de los datos pero si necesitar un registro de todos los cambios que han ocurrido. Esto es útil para mantener una auditoria del sistema y también puede ser útil para proveer los cambios para procesos especiales. Por ejemplo, los cambios realizados un sistema fuente pueden ser almacenados en una tabla intermedia, donde se analicen y validen, antes de procesarlos y enviarlos a los sitios destino. Esta es una variante de la técnica de distribución de datos, en la cual el destino sólo recibe los cambios.

Los cambios también pueden ser utilizados en escenarios de replicación multi-capas, donde son copiados desde una fuente central hacia un área de almacenamiento en otro sistema y finalmente desde esta área a los sitios destino. Esto es muy útil en caso de existir muchos sitios destino, ya que minimiza el impacto sobre el sistema fuente.

1.3.4 Conflictos de la replicación de datos

Existen varios tipos de conflictos a tener en cuenta, al tratar la replicación de datos. Los conflictos pueden ocurrir cuando estamos trabajando en un ambiente de replicación que permite actualizaciones concurrentes sobre los mismos datos en múltiples sitios.

- ✓ **Conflicto de actualización**

Un conflicto de actualización ocurre cuando se produce la replicación de una actualización (*update)* sobre un registro con otra actualización (*update)* sobre el mismo registro. Este conflicto ocurre cuando dos transacciones originadas desde distintos sitios actualizan el mismo registro, en forma cercana en el tiempo.

Resolución

- ❖ Prioridad: Cada servidor obtiene una prioridad única, y el servidor de mayor prioridad "gana", respecto de aquellos con prioridad menor.
- ❖ Timestamp: La más nueva o la más antigua de las modificaciones es la considerada correcta, y por defecto, sino se eligió ninguno de los criterios "gana" la más nueva.
- ❖ Particionamiento de datos: Se garantiza que cada registro sea manipulado por un único servidor, lo que simplifica la arquitectura.

- ✓ **Conflicto de unicidad**

El conflicto de unicidad sucede cuando la replicación de un registro intenta violar una restricción de integridad, ya sea por llave primaria o única (*Primary Key* o *Unique).* Por ejemplo considere lo que sucede, cuando dos transacciones originadas de dos sitios diferentes, cada una inserta un registro, en su respectiva tabla replicada, con el mismo valor de clave primaria. En ese caso sucede un conflicto de unicidad.

Resolución

- ❖ Para cada servidor brindar un rango distinto de números para los generadores de clave (secuencias).
- ❖ Agregar el identificador del servidor a la clave primaria.
- ❖ Replicar en tablas separadas, y acceder a los datos a través de una vista formada por la unión de ellas. Para resolver el conflicto de potenciales claves duplicadas en la unión se usará una pseudo columna que representa la base de datos fuente.

- ✓ **Conflicto de supresión**

Un conflicto de supresión ocurre cuando dos transacciones originadas de sitios diferentes, una de ellas intenta borrar un registro, y la otra actualizar o borrar el mismo registro, ya que en este caso el registro no existe, tanto para ser actualizado como borrado.

Resolución

Para evitar este tipo de conflictos, una posible solución es que los sitios marquen lógicamente los registros a ser borrados y que periódicamente el sitio maestro corra un proceso que realice el borrado ("delete") físico de los datos, es decir desde los sitios replicados no se puede ejecutar una sentencia para hacer el borrado de los datos (delete).

- ✓ **Conflicto de orden**

Los conflictos de orden pueden ocurrir en ambientes de replicación con tres o más sitios maestros. Si la propagación al sitio maestro X, está bloqueada por alguna razón, entonces la replicación de modificaciones en datos puede seguir siendo propagada a través de los otros sitios maestros; al finalizar la propagación estas modificaciones debieron ser propagadas al sitio X en un orden diferente a como ocurrieron en los otros sitios maestros, pudiendo producirse un conflicto.

Resolución

Este tipo de conflictos suelen resolverse asignándole distintas prioridades a los sitios maestros, de forma de ordenar las transacciones de acuerdo a esta. [8]

1.4 Tendencias y tecnologías actuales

1.4.1 Software libre

Software libre es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. Análogamente, el software gratuito (denominado usualmente Freeware) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de software no es libre en el mismo sentido que el software libre, al menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa. [9]

En 1984, Richard Stallman comenzó a trabajar en el proyecto GNU (es un acrónimo recursivo para "Gnu No es Unix"), fundando la Fundación de Software Libre (*FSF, Free Software Foundation*) un año más tarde. Stallman introdujo una definición para software libre (en inglés, free software), el cual desarrolló para dar a los usuarios libertad y para restringir las posibilidades de apropiación del software.

De acuerdo con tal definición, el software es "libre" si garantiza:

- ✓ la libertad para ejecutar el programa con cualquier propósito (llamada "libertad 0").
- ✓ la libertad para estudiar y modificar el programa ("libertad 1").
- ✓ la libertad de copiar el programa de manera que puedas ayudar a tu vecino ("libertad 2").
- ✓ la libertad de mejorar el programa, y hacer públicas tus mejoras, de forma que se beneficie toda la comunidad ("libertad 3").

Es importante señalar que las libertades 1 y 3 obligan a que se tenga acceso al código fuente.

1.4.2 Software libre y Software de código fuente abierto

El término *free software* resulta ambiguo en el idioma inglés, ya que *free* significa además de libre, gratis. Para evitar esta ambigüedad es acuñado entonces por Christine Peterson, el término “open source” para definir todo aquel software que fuera de código fuente abierto. “El significado obvio para *software de código fuente abierto* es: *usted puede mirar el código fuente*. Este es un criterio más pobre que *software libre*. *Software de código fuente abierto* incluye software libre, pero también incluye programas semi-libres”. [10]

El movimiento *Open Source* apareció en 1998 con un grupo de personas, entre los que cabe descatar a Eric S. Raymond y Bruce Perens, que formaron la Open Source Initiative (OSI). Buscaban darle mayor relevancia a los beneficios prácticos de compartir el código fuente e interesar a las principales casas de software y otras empresas de la industria de la alta tecnología en el concepto. [9]

El movimiento del software libre hace especial énfasis en los aspectos morales o éticos del software, viendo la excelencia técnica como un producto secundario deseable de su estándar ético. El movimiento *Open Source* ve la excelencia técnica como el objetivo prioritario, siendo la compartición del código fuente un medio para dicho fin. Por dicho motivo, la FSF se distancia tanto del movimiento *Open Source* como del término "*Open Source*".

1.4.3 Sistema Gestor de Base de Datos

1.4.3.1 PostgreSQL

PostgreSQL es un sistema gestor de base de datos objeto-relational (***O**bject **R**elational **D**atabase **M**anager **S**ystem*), basado en POSTGRES Versión 4.2, desarrollado en la Universidad de California, en el Departamento de Ciencias de la Computación de Berkeley.

POSTGRES es pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde.

PostgreSQL es un descendiente del "código abierto" (en inglés, open source) original de Berkeley. Soporta gran parte del SQL estándar y muchas modernas funcionalidades como:

- ✓ Consultas complejas
- ✓ Llaves foráneas
- ✓ Disparadores (en inglés, triggers)
- ✓ Vistas
- ✓ Integridad transaccional
- ✓ Control de versionado concurrente (MVCC en sus siglas en ingles). Estrategia de almacenamiento que permite trabajar con grandes volúmenes de datos.

Cumple completamente con las características ACID (acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español) para realizar transacciones seguras, es multiplataforma, está disponible para 34 plataformas en su última versión estable. Con la integridad referencial y posee interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY, además de traer soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos (*SSL, Secure Sockets Layer*).

Además, PostgreSQL puede ser personalizado por el usuario en muchas formas, según sus necesidades, por ejemplo, adicionando entre otros, un nuevo:

- ✓ Tipo de datos
- ✓ Funciones
- ✓ Operadores
- ✓ Funciones agregadas

- ✓ Lenguajes procedurales

Y debido a la liberación de la licencia, PostgreSQL puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos. [11]

1.4.3.2 PL/PGSQL

PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL. Permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones, dando mucho más control automático que las sentencias SQL básicas.

Desde PL/pgSQL se pueden realizar cálculos complejos y crear nuevos tipos de datos de usuario. Como un verdadero lenguaje de programación, dispone de estructuras de control repetitivas y condicionales, además de la posibilidad de creación de funciones que pueden ser llamadas en sentencias SQL normales o ejecutadas en eventos de tipo disparador (trigger).

Las funciones escritas en PL/pgSQL aceptan argumentos y pueden devolver valores de tipo básico o de tipo complejo (por ejemplo, registros, vectores, conjuntos o incluso tablas), permitiéndose tipificación polimórfica para funciones abstractas o genéricas (referencia a variables de tipo objeto). [12]

Con PL/pgSQL puede agrupar un grupo de computaciones y una serie de consultas dentro del servidor de bases de datos, teniendo así la potencia de un lenguaje procedural y la sencillez de uso del SQL, del cual puede usar todos los tipos de datos, columnas, operadores y funciones, pero ahorrando una gran cantidad de tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor, lo cual puede redundar en un considerable aumento del rendimiento.

Debido a que las funciones PL/pgSQL corren dentro de PostgreSQL, estas funciones funcionarán en cualquier plataforma donde PostgreSQL corra. Así se podrá rehusar el código y reducir costes de desarrollo. [13]

1.4.4 Python: ¿Alternativa para el desarrollo?

Es un lenguaje interpretado, orientado a objetos de propósito general. Permite mantener de forma sencilla la interacción con el sistema operativo y resulta muy adecuado a la hora de manejar ficheros de texto. Es un lenguaje que se destaca por su rapidez de desarrollo.

Python es extensible, resulta fácil añadir una nueva función o módulo al intérprete, para realizar operaciones críticas a la máxima velocidad.

Se puede enlazar el intérprete Python a una aplicación escrita en C y utilizarlo como lenguaje de macros para dicha aplicación.

Es soportado en las siguientes plataformas: Unix, Windows, OS/2, Mac, además de él mismo poder integrarse con otras plataformas de desarrollo como Java, para lo cual usa Jython, una implementación de Python para la máquina virtual de Java.

La implementación de Python es bajo la licencia de código abierto que permite el uso libre y distribución para un uso comercial. La licencia de Python es administrada por la Fundación Python de Software. [14]

1.4.5 Interfaz Gráfica de Usuario

1.4.5.1 GTK

GTK (GIMP Toolkit) es una biblioteca para crear interfaces gráficas de usuario. Ofrece un juego completo de elementos para construir una interfaz de usuario a la medida.

Se llama GIMP toolkit porque fue escrito para el desarrollo del Programa de manipulación general de imagen (GIMP: General Image Manipulation Program), pero ahora GTK se utiliza en un gran número de proyectos de programación, incluyendo el proyecto de entorno de escritorio multiplataforma para Linux GNOME (GNU Network Object Model Environment en su significado en ingles).

GTK es software libre y parte del proyecto GNU, sin embargo, los términos de la licencia para GTK, la licencia pública menor de GNU (LGPL:Lesser General Public License), se podrán desarrollar programas con licencias abiertas, gratuitas, libres, y hasta licencias comerciales no libres.

GTK es basado en tres librerías desarrolladas por el equipo de GTK:

Glib es la biblioteca de bajo nivel que constituye la base principal de GTK y GNOME. Suministra el manejo de estructura de datos para C, cubierta de portabilidad, e interfaces para funcionalidades de tiempo de ejecución, bucle de evento, hilos, carga dinámica, y un sistema de objeto.

Pango es una biblioteca para el diseño y la versión del texto, con un énfasis sobre la internacionalización. Constituye el punto principal de texto y tipo de caracteres respondiendo para GTK +-2.0

La biblioteca ***ATK*** suministra un juego de interfaces para la accesibilidad. Soportando las interfaces de ATK, una aplicación o juego de herramientas pueden ser usados con tales herramientas como lectores de pantalla y dispositivos de entrada alternativos, entre otros.

GTK ha sido diseñado desde el principio para respaldar un rango de leguajes, no solamente C/C++. Usando GTK desde lenguajes como Perl y Python (especialmente en combinación con Glade, constructor de Interfaz Gráfica de Usuarios) brinda un método eficaz del desarrollo rápido de aplicaciones. [15]

1.4.5.2 Glade

Glade es una herramienta de desarrollo rápido de aplicaciones (RAD, en sus siglas en ingles), para desarrollar fácil y rápido interfaces de usuario para GTK/GNOME como entorno de escritorio, bajo la licencia pública general (GNU GPL, General Public License).

La interfaz de usuario diseñada en Glade es salvada como un fichero de lenguaje de marcado extensible (*XML, e**X**tensible **M**arkup **L**anguage*), y usando la librería libglade puede ser cargado dinámicamente por las aplicaciones que lo necesiten.

Al usar libglade, los ficheros XML de Glade pueden ser usados por numerosos leguajes de programación incluyendo C, C++, Java. Perl y Python, entre otros, además de no ser difícil añadirle soporte para otros muchos. [16]

1.4.6 Proceso de desarrollo de software utilizado

Para controlar y planificar la propuesta que presenta este trabajo, se decidió utilizar como proceso de desarrollo de software el Proceso Unificado de Rational (RUP), por sus características y las facilidades que aporta a todo el proceso.

1.4.6.1 El Proceso Unificado de Rational

El Proceso Unificado de Rational (*RUP, Rational Unified Process*) es el producto final de tres décadas de desarrollo y uso práctico. Su desarrollo sigue un camino desde 1967 con la Metodología Ericsson (Ericsson Approach), una aproximación de desarrollo basada en componentes, que introdujo el concepto de caso de uso; pasando por el proceso Objectory de Rational (publicado en 1997) hasta el Proceso Unificado de Racional, publicado en 1998. [17]

El Proceso Unificado es una propuesta de proceso para el desarrollo de software orientado a objeto que utiliza el Lenguaje Unificado de Modelado (*UML, Unified Model Language)* para describir todo el proceso. Está basado en componentes, lo cual quiere decir que el sistema en

construcción está formado por componentes interconectados a través de interfaces bien definidas.

Sus características principales son:

1. Guiado/Manejado por casos de uso.
2. Centrado en arquitectura.
3. Iterativo e Incremental.
4. Desarrollo basado en componentes.
5. Utilización de un único lenguaje de modelación.
6. Proceso Integrado. [17]

La creación sólida de software de calidad requiere el conocimiento específico de las tareas que deben llevarse a cabo en cada entorno. Ahí radica la importancia de aplicar un proceso de desarrollo flexible y adaptado a cada objetivo de desarrollo. RUP combina un conjunto básico de mejores prácticas aprobadas por el sector con una serie de complementos opcionales del proceso a fin de dar cabida y soporte a proyectos de cualquier envergadura o alcance. Cualquier tipo de proyecto (incluidos los pequeños, los basados en web, aquellos fundamentales para un proyecto y los proyectos integrados) permite obtener resultados más acordes con las previsiones gracias a la aplicación de RUP.

1.4.6.2 UML

UML se comenzó a gestar en la empresa Rational cuando Booch y Rumbaugh decidieron unir sus métodos para conseguir un lenguaje estándar y sólido. Más tarde se incorporó Jacobson, lo que dio lugar a la versión 0.9 de UML en 1996. En ese mismo año el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares. Posteriormente se creó un consorcio con varias organizaciones interesadas en UML. La versión 1.0 de UML surgió en 1997 con la contribución de IBM, HP, Oracle, Microsoft y otras organizaciones y como una respuesta a esta petición en enero de 1997. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997.

El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. Proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases escritas en un

lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables. [18]

1.4.7 Propuesta de desarrollo

Todas las herramientas descritas anteriormente cumplen con la política de software libre seguida para el desarrollo de aplicaciones informáticas en las Fuerzas Armadas Revolucionarias, rectorada por el Centro Principal de Automatización (CPA) del MINFAR. Además de algunas de ellas tener definido su uso por la entidad, otras se integran perfectamente a aplicaciones desarrolladas para la institución, como es el caso del lenguaje de programación Python, incluido en todos los sistemas GNU Linux.

En la aplicación se integra Glade para construir la interfaz de usuario con Python como lenguaje de desarrollo. En la escritura de los disparadores el lenguaje procedural PL/PGSQL, como parte del sistema gestor de base de datos PostgreSQL para el cual se desarrolla la aplicación.

Toda la construcción guiada por RUP como proceso de desarrollo de software.

1.5 Conclusiones

En este capítulo se han introducido conceptos indispensables para la comprensión del proceso de distribución de la información, con el uso de las tecnologías actuales. La mejor solución al problema de actualización de la información es la aplicación de la técnica de réplica de datos. Se hace necesaria la construcción de un software que ayude a configurar esta técnica en los servidores de datos de las FAR.

Se detalla un estudio de algunas de las últimas tendencias que a nivel mundial se vienen utilizando en el mundo del software. No se podía por tanto dejar de hablar de software libre cuando nuestro país, como muchos otros, ya se ha trazado la meta de implantarlo en numerosas esferas a nivel nacional, siendo un ejemplo de ello las FAR, por lo es uno de los aspectos que se ha considerado en el momento de escoger las herramientas a utilizar para construir la solución que se propone.

Capítulo 2 Características del sistema

2.1 Introducción

El presente capítulo constituye el resultado del análisis del objeto de estudio y los procesos que tienen lugar actualmente del negocio que tiene que ver con el objeto de estudio. Para ello se definen conceptos que se agrupan en un Modelo de Negocio. A partir de este estudio se definen los requisitos funcionales y no funcionales que deberá satisfacer la propuesta de solución, dándose además una descripción detallada de la misma.

2.2 Descripción del objeto de estudio

2.2.1 Situación Problémica

La institución, máxima responsable de la defensa del país y su soberanía, ha apostado por la rama de la informática para darle solución a problemas cruciales en su desempeño por lo que es de máxima prioridad para ellos viabilizar el proceso de gestión de la información entre los niveles de mando de manera rápida y confiable.

Los sistemas diseñados para informatizar los procesos de las FAR, usan bases de datos distribuidas por los sitios de la red, para los cuales esté destinada la misma.

La información se fragmenta siguiendo una política de fragmentación horizontal, partiendo del mayor número de acceso a los datos en cada sitio, es decir, cada sitio maneja sus datos a través de la misma aplicación igualmente distribuida, la cual emplea PostgreSQL como sistema gestor de base de datos.

El proceso de actualización de la información no se lleva a cabo eficientemente, pues esta no sigue un curso de centralización para su análisis posterior.

Los especialistas del MINFAR, deben tener conocimiento de la información de cada una de las entidades del país donde se trabaje en su especialidad.

Actualmente, la transmisión de la información se realiza a través de una funcionalidad en los sistemas informáticos que se desarrollan que permite generar un fichero (despacho) con los cambios producidos en la base de datos entre el último realizado y la fecha en curso. Este despacho es enviado por el usuario que lo genera al nivel inmediato superior vía correo electrónico.

Por su parte, el oficial inmediato superior debe incorporarlo a su base de datos para su posterior actualización mediante otra funcionalidad del sistema diseñada para ello.

Es un proceso análogo a la técnica de replicación vista anteriormente “Descarga y Recarga”, los problemas en ambos casos no difieren, el retardo en la actualización de la información es un aspecto clave en este sentido, dificultando el proceso de toma de decisión en su amplio espectro.

Esto implica un mayor tiempo de desarrollo en las aplicaciones, así como es una solución parcial al problema, partiendo del punto que la información llegará a la instancia superior, sólo que no con la puntualidad necesaria.

2.2.1.1 Softwares existentes relacionados

Existen en el mundo múltiples soluciones que permiten llevar a cabo la réplica de datos enfocadas al gestor de base de datos utilizado en cuestión. Pueden citarse entre ellos Postgres-R, PgReplicator, Usogres, ErServer, aunque ninguno de ellos implementa una solución de réplica para bases de datos fragmentadas.

Estas aplicaciones se dividen de acuerdo a los entornos de réplica donde pueden ser aplicadas, para el entorno “maestro-esclavo”, la más popular es:

Slony I

Slony es el plural de elefante en el idioma ruso. La mascota para Slony, Slon es una muy buena variación del usual elefante mascota de Postgres.

Es un sistema de replicación “maestro a múltiples esclavos”, soporta la réplica en cascada y permite a un esclavo ser a su vez maestro para otro servidor.

Incluye los tipos de funcionalidades necesarias para replicar bases de datos grandes a un número razonablemente limitado de sistemas esclavos. “Razonable,” en este contexto, un orden de una docena de servidores. Si el número de servidores crece más allá de ese, el coste de comunicaciones aumentará prohibitivamente, y las ventajas incrementales de tener servidores múltiples fallarán en ese punto.

Además permite indicar qué cambios replicar de un servidor a otro.

Slony-I implementa la réplica asincrónica, usando disparadores para determinar las actualizaciones de las tablas, donde un solo “origen” (maestro) se puede replegar a los “suscriptores múltiples” (esclavos) incluyendo suscriptores conectados en cascada. [19]

Slony I realiza una réplica de espejos, exactamente igual al origen de datos, no es posible actualizar los datos a medida que se produce algún cambio en ellos. Puede ser útil en determinadas situaciones, pero no da una solución al problema planteado inicialmente.

Para el caso del entorno “multi-maestro” es:

PgCluster

Es un sistema de replicación síncronico, “multi-maestro” para PostgreSQL
PGCluster consiste en tres tipos de servidores, un servidor para balance de carga, cluster DB y un servidor de réplica.
Tiene dos funciones principales:

Compartir carga

- ✓ La carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones Web donde existe gran demanda por el número de peticiones.

Alta disponibilidad

- ✓ Cuando ocurre un fallo en el Cluster DB, el servidor de balance de carga y el de replicación separan el fallo del sistema, y continúa el servicio que usa el DB restante.
- ✓ El Cluster DB cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio.
- ✓ Los datos son copiados automáticamente a DB restaurada o añadidos desde otra DB. [20]

Este último con objetivos definidos para soportar una alta disponibilidad, para controlar los fallos de los servidores, es un sistema propietario además.

Ambos sistemas son internacionales, en Cuba no se ha dado ninguna solución al respecto, así como tampoco en la Universidad, donde sí se ha trabajado en el tema, pero para el gestor de base de datos Oracle.
El problema va por el lado de elegir la mejor opción para replicar bases de datos en PostgreSQL y que cumpla con las funcionalidades que se necesitan desarrollar.

2.2.2 Objeto de automatización

Se automatizaran varios procesos, todos con el fin de darle solución a la situación problémica planteada anteriormente, entre ellos:

Configuración de Slony, como herramienta para replicar espejo

Como se explicó anteriorme Slony es una herramienta provechosa, aunque no resuelve el problema planteado en su totalidad puede ser útil en muchas situaciones, un ejemplo es el caso de los nomencladores (más del 95%) siempre se actualizarán desde el Ministerio.

Configuración de un esquema de réplica para replicar bases de datos fragmentadas (ver Figura 2.1).

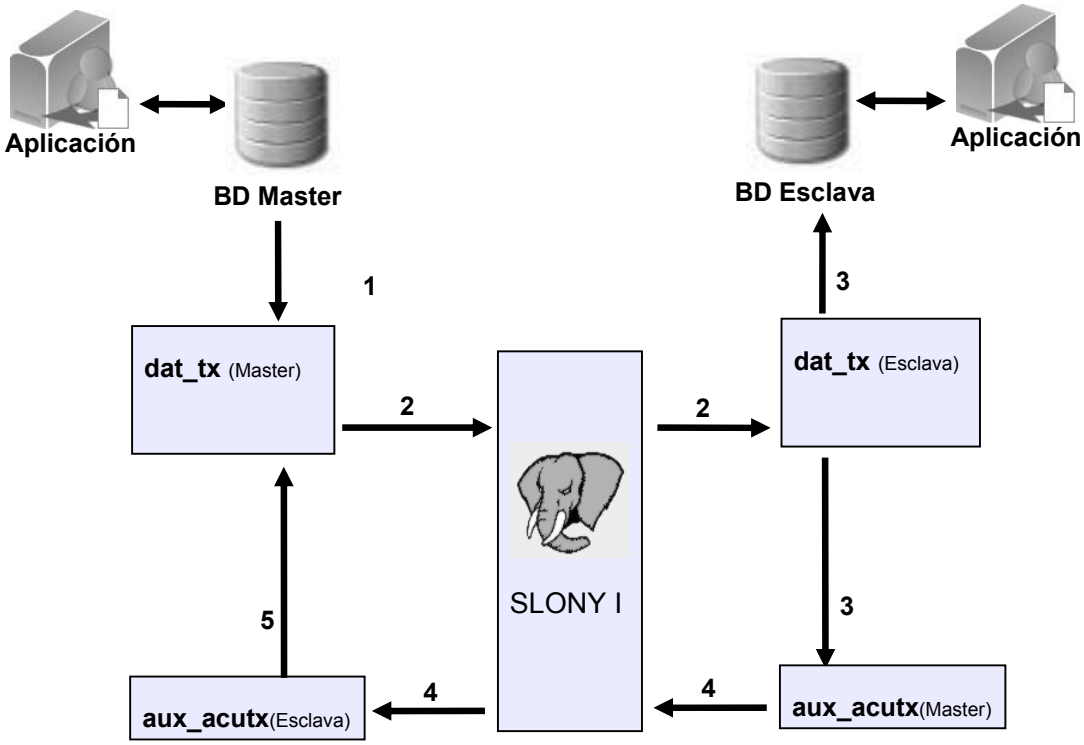


Figura 2.1 Esquema general de replicación de BD para base de datos fragmentadas

A continuación se detallan la responsabilidad de cada flujo en la propuesta.

Flujo 1

✓Será un trigger escrito en pl/pgsql que se dispare siempre luego de uno de los eventos de inserción, borrado o actualización (INSERT, DELETE o UPDATE), reconstruirá la sentencia ejecutada y la insertará en la entidad dat_tx (master).

✓Según las reglas definidas decidirá si la sentencia sql es o no propagada al nodo esclavo.

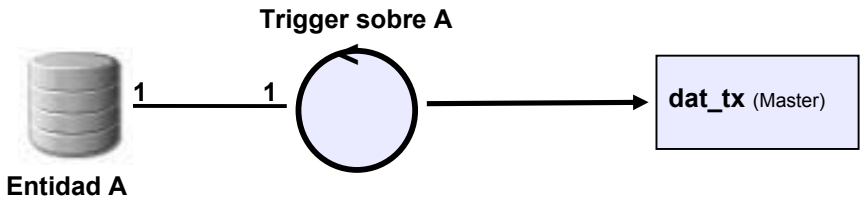


Figura 2.2 Descripción del Flujo 1 de la propuesta

Flujo 2

Slony con su capacidad para crear espejos replicará cada tupla que pongamos en dat_tx (master) en dat_tx (esclava o espejo). Aún cuando se haya perdido la conexión, éste en cuanto se restablezca realizará la actualización de las mismas.

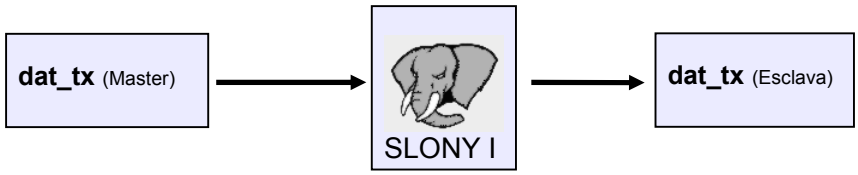


Figura 2.3 Descripción del Flujo 2 de la propuesta

Flujo 3

Será un trigger que se dispare siempre luego de una inserción (after INSERT) sobre la entidad dat_tx (esclava). Su cuerpo estará implementado en pl/pgsql, posibilitando así la conexión a la "BD esclava" y la ejecución de las instrucciones sql recibidas. De esta última haberse ejecutado satisfactoriamente será insertado su identificador (llave) en la entidad dat_acutx (master).

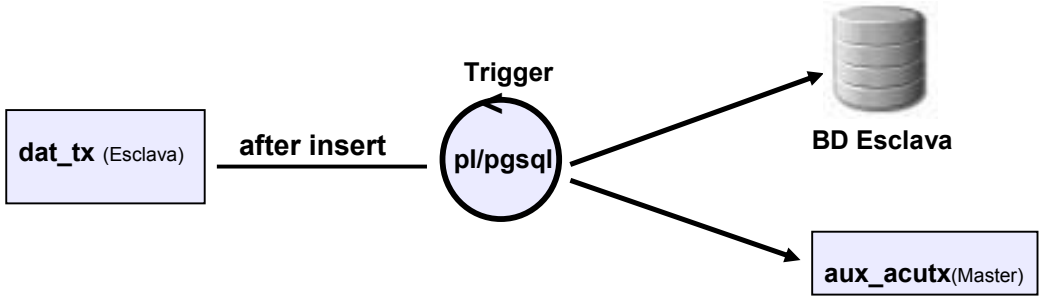


Figura 2.4 Descripción del Flujo 3 de la propuesta

Flujo 4

Slony replicará cada tupla que pongamos en dat_acutx (master) en dat_acutx (esclava o espejo). Esta es la propuesta para mantener la entidad dat_tx (master) vacía, evitando así la lentitud por sobrecarga del sistema.

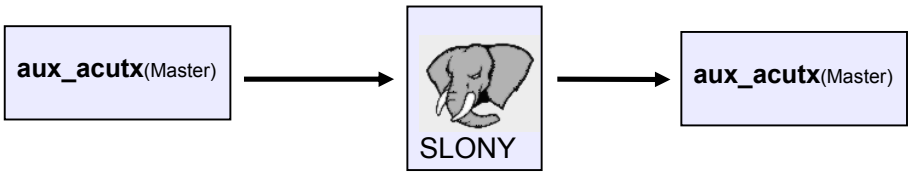


Figura 2.5 Descripción del Flujo 4 de la propuesta

Flujo 5

Será un trigger (pl/pgsql) que se dispare siempre luego de una inserción (after INSERT) sobre la entidad aux_acutx (esclava), este tendrá la responsabilidad de ejecutar el borrado de todas las tuplas que estén en dat_tx (master) en la medida en que lleguen a él sus identificadores (así mismo se vacía dat_tx (esclava)).

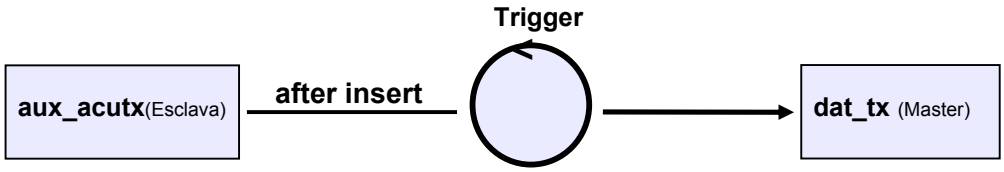


Figura 2.6 Descripción del Flujo 5 de la propuesta

Esta propuesta fue sometida a una prueba de factibilidad y tolerancia a fallos, se escogió una tabla para la réplica, se crearon los triggers y se configuró Slony para dos servidores de datos independientes. Al insertar 5500 registros aproximadamente la propuesta funcionó correctamente, a mitad del proceso se desconectó la red del servidor maestro y al ser conectada nuevamente los datos que estaban en cola llegaron sin problemas al servidor esclavo, por lo que se demostró que presenta una correcta tolerancia a fallos.

Creación de reglas para la réplica

Esta opción permitirá definir los filtros de la réplica de datos. Asegurará en cada servidor los datos que se necesiten.

2.2.3 Propuesta de Sistema

El sistema es una herramienta de gestión de réplica para bases de datos distribuidas en PostgreSQL, gestor usado en el desarrollo de aplicaciones para las FAR.
En este campo de desarrollo existen diversas herramientas, ya vistas anteriormente. Todos son sistemas a base de ficheros de configuración manuales.

Una propuesta para la réplica de fragmentos de datos específicamente, que permita replicar la información en función de reglas definidas previamente, no es conocida, luego de la investigación previa realizada al comienzo del análisis del sistema.

2.2.3.1 Modelo de Dominio

Teniendo en cuenta las descripciones de los procesos en el comienzo del capítulo, nos damos cuenta de que el negocio que estamos estudiando, tiene muy bajo nivel de estructuración. Por tanto trataremos de dar un enfoque a todo el proceso de actualización de la información.

Para ello nos basaremos en un modelo del dominio, ya que nos permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes y desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Primeramente se identifican todos los conceptos asociados con el problema.

Conceptos del Modelo del Dominio

Concepto	Descripción
Entidad	Representa a las dependencias de la Institución, tales como regiones, sectores, unidades militares, sectores militares, etc.
Sistema	Aplicación de software en funcionamiento, accesible al personal de las dependencias de la entidad.
Usuario	Representa a personas o sistemas de la entidad que interactúan con la aplicación.
BD	Base de datos asociada al sistema.
Administrador BD	Persona encargada del mantenimiento y gestión de las bases de datos en la entidad.
Despacho	Fichero que se envía de una entidad a otra con el objetivo de actualizar las bases de datos de dichas entidades.

Tabla 2.1 Conceptos asociados al dominio del problema

Los conceptos de más relevancia durante el desarrollo de la aplicación, se relacionan entre sí para dar a conocer el dominio de la información del sistema. Esto se representa a continuación en la Fig. 2.7:

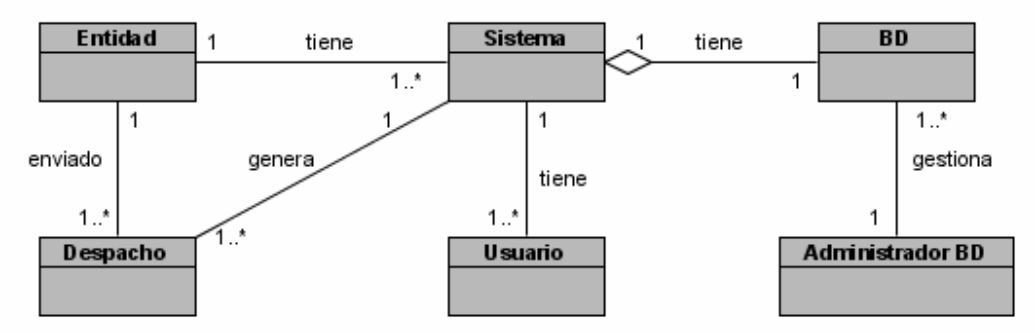


Figura 2.7 Diagrama del Modelo de Dominio

Descripción del Modelo de Dominio

Cada entidad de las FAR, se auxilia con el fin de agilizar sus procesos de sistemas informáticos, los cuales tienen su base de datos correspondiente. Estas bases de datos son gestionadas por un administrador que vela por su correcto funcionamiento.

Por su parte los sistemas tienen varios usuarios, uno de ellos tiene la responsabilidad de interactuar con el sistema para generar un despacho cada cierto tiempo y enviarlo a una entidad de un nivel superior o inferior dentro del árbol jerárquico.

2.3 Especificación de Requisitos

2.3.1 Requerimientos funcionales

R1 Conectar servidor BD
1.1 Especificar datos del servidor.
1.2 Seleccionar BD maestra.
1.3 Seleccionar BD esclava.

R2 Configurar la réplica de espejo
2.1 Conectar BD origen (maestra) y destino (esclava) (Ref. R1).
2.2 Seleccionar entidades a replicar.
2.3 Generar ficheros de configuración de Slony.

R3 Configurar la réplica de fragmentos
3.1 Conectar BD origen (maestra) y destino (esclava) (Ref. R1).
3.2 Seleccionar entidades a replicar.
3.3 Crear el nodo de replicación.
3.4 Especificar reglas para efectuar la réplica para cada una de las entidades seleccionadas (pudieran haber reglas de subconjuntos de datos, etc.).
3.4.1 Agregar reglas.
3.4.2 Modificar reglas.
3.4.3 Eliminar reglas.
3.5 Generar triggers.
3.6 Generar ficheros de configuración de Slony.

R4 Reporte de réplica
4.1 Conectar BD origen (maestra) (Ref. R1).
4.2 Mostrar Cluster.
4.3 Mostar usuarios de réplica.
4.4 Mostrar entidades replicadas en cada cluster.
4.5 Mostrar reglas creadas en cada entidad

R5 Realizar copias de seguridad de las BD.
5.1 Seleccionar BD (Ref. R1).
5.2 Permitir copias por Backup.
5.3 Permitir copias por Metadato.

R6 Realizar mantenimiento de las BD.
6.1 Seleccionar BD. (Ref. R1).
6.2 Realizar mantenimiento por Vacuum.
6.3 Realizar mantenimiento por ReIndex.

2.3.2 Requerimientos no funcionales

- Interfaz externa

Diseño sencillo e intituivo que permita a usuarios no familiarizados un uso extensivo de la aplicación.

- Usabilidad

La aplicación está concebida para un número limitado de usuarios, sólo podrá ser utilizado por aquellas personas que de una u otra forma deban velar por la integridad de la información en los servidores de datos de las FAR.

- Rendimiento

La aplicación debe tener un alto nivel de respuesta, tanto para los accesos a la BD, como para el proceso de réplica de datos.

- Portabilidad

La aplicación está diseñada para el sistema operativo GNU/Linux, por ser este el sistema operativo usado para los servidores de datos de las FAR.

- Seguridad

La aplicación no cuenta con el de autentificación y control de usuarios, pues es una aplicación que utiliza la autenticación del sistema operativo para poder acceder a ella. Concebida para servidores, solo podrá tener acceso físico un administrador autorizado.

- Software

Como sistema operativo *GNU/Linux Gentoo* (intregra el Python), para la interfaz gráfica la *librería GTK* y la *libglade*, como gestor de base de datos *PostgreSQL 8.1* o superior.

- Confiabilidad

Tendrá un alto grado de confiabilidad, dado su fin de gestionar la información.

- Ayuda y documentación en línea

Contará con un manual de usuario, que permita la aclaración de cualquier duda al interactuar con la aplicación, así como la opción de ayuda en línea a través de las facilidades que brinda la interfaz de usuario.

2.4 Descripción del sistema propuesto

2.4.1 Concepción general del sistema

La aplicación a desarrollar constituirá una herramienta para los administradores de las bases de datos de las FAR. Es una aplicación de escritorio que cumpla con las necesidades del usuario y además brinde otras mejoras.

Aunque los administradores de bases de datos son las únicas personas con acceso a su utilización, brinda beneficios a los especialistas del MINFAR, viabilizando su trabajo en la toma de decisiones.

Es importante tener en cuenta su generalidad, pues permitirá de manera flexible la configuración de las políticas de réplica de datos entre los servidores de las FAR.

Los actores del sistema serían entonces:

Actores del Sistema	Justificación
Administrador BD	Tiene acceso a todas las facilidades que brinda el sistema. Es el encargado de gestionar las BD y configurar la réplica de datos.

Tabla 2.2 Actores del Sistema

2.4.2 Modelo de casos de uso del sistema

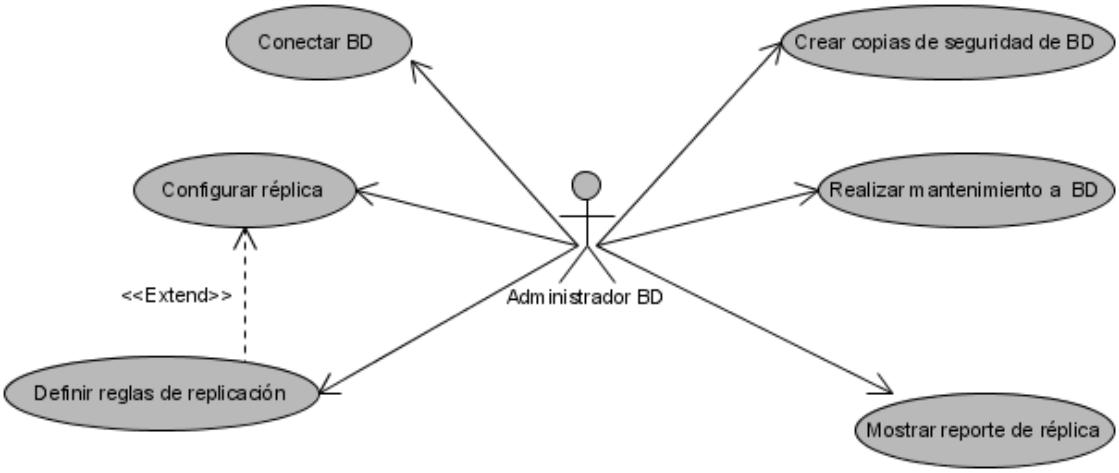


Figura 2.8 Diagrama del Casos de Uso del Sistema

2.4.2.1 Casos de uso por ciclo

Código	Nombre de caso de uso	Justificación de la selección
CU-1	Conectar BD	Crítico
CU-2	Configurar réplica	Crítico
CU-3	Definir reglas de replicación	Crítico
CU-5	Mostrar reporte de réplica	Secundario
CU-6	Crear copias de seguridad de BD	Secundario
CU-7	Realizar mantenimiento a BD	Secundario

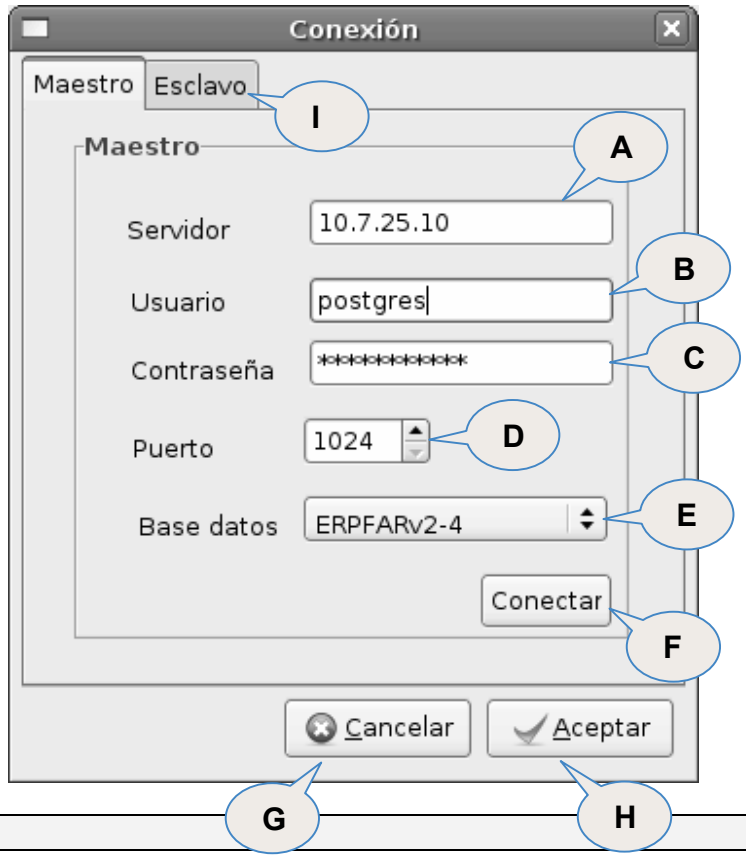
Tabla 2.3 Casos de Uso por Ciclos

2.4.3 Expansión de los casos de uso

A continuación se muestra la descripción expandida de los casos de uso del sistema:

Caso de Uso	Nombre del caso de uso
CU-1	Conectar BD
Actores	Administrador del servidor de BD
Propósito	Habilitar conexión a la base de datos para su posterior gestión.

Resumen	El caso de uso se inicia cuando el actor selecciona la opción Conectar. El sistema muestra la interfaz de conexión, que incluye la conexión al servidor maestro y al esclavo, para luego conectarse a las bases de datos, el actor realiza las operaciones necesarias, quedando conectado el sistema a la(s) BD seleccionada.
Referencias	R1, 1.1; R2, 2.1; R3, 3.1; R4, 4.1; R5, 5.1; R6, 6.1.
Precondiciones	
Interfaz I	



Descripción

A: Entry para especificar el número IP del servidor de datos.

B: Entry para especificar el usuario para conectarse a la BD.

C: Entry para especificar la contraseña del usuario.

D: SpinButton para especificar el número del puerto para la conexión.

E: Combobox para seleccionar la base de datos requerida.

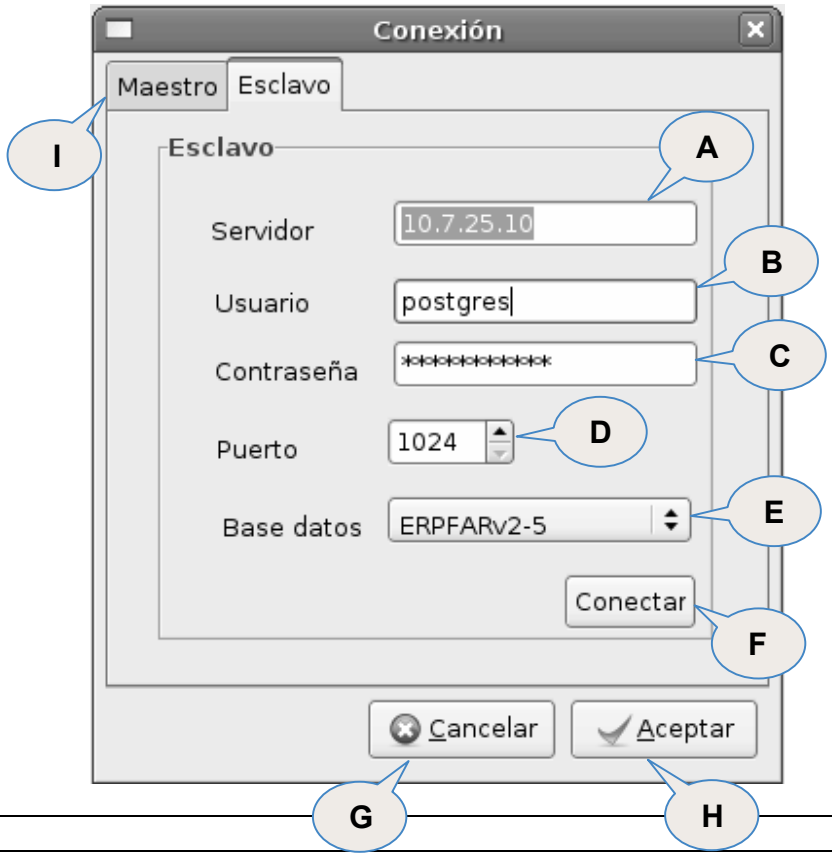
F: Button para conectarse al servidor de datos

G: Button para cancelar la acción de conectarse.

H: Button para aceptar la acción de la conexión.

I: Pestaña para los datos del servidor esclavo.

Interfaz II



Descripción

A: Entry para especificar el número IP del servidor de datos.

B: Entry para especificar el usuario para conectarse a la BD.

C: Entry para especificar la contraseña del usuario.

D: SpinButton para especificar el número del puerto para la conexión.

E: Combobox para seleccionar la base de datos requerida.

F: Button para conectarse al servidor de datos

G: Button para cancelar la acción de conectarse.

H: Button para aceptar la acción de la conexión.

I: Pestaña para los datos del servidor maestro.

<table>
<tr><th colspan="2">Flujo Normal de Eventos</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. Selecciona Conectar.</td><td>2. El sistema muestra la Interfaz I.</td></tr>
<tr><td>3. El actor introduce los datos necesarios para conectarse al servidor de datos primario (Servidor, usuario, contraseña y puerto).</td><td></td></tr>
<tr><td>4. Verifica que los datos estén correctos y presiona el botón Conectar.</td><td>5. El sistema muestra el control con las bases de datos existentes en el servidor.</td></tr>
<tr><td></td><td>6. El sistema indica la conexión con la BD especificada.</td></tr>
<tr><td>7. El actor selecciona la base de datos requerida.</td><td></td></tr>
<tr><td>8. El actor presiona la pestaña “Esclavo”.</td><td>9. El sistema muestra la Interfaz II.</td></tr>
<tr><td>10. Se repite desde la línea 3 del curso normal hasta la línea 6.</td><td></td></tr>
<tr><td>11. El actor presiona el botón Aceptar.</td><td>12. El sistema muestra la ventana principal.</td></tr>
<tr><td colspan="2">Cursos Alternos</td></tr>
<tr><td colspan="2">Curso Normal
Línea 4
– Si el actor omite un dato requerido y oprime el botón Conectar, el sistema muestra el mensaje “Se necesita especificar la contraseña”.
– Si los datos son erróneos y oprime el botón Conectar, el sistema muestra el mensaje “Falló la conexión, datos incorrectos”.
Línea 9
– Si el actor no especifica un servidor de datos maestro y uno esclavo, el sistema no tendrá habilitadas las funcionalidades de la Réplica.
Línea 12
– Si el actor oprime el botón Cancelar, el sistema muestra un mensaje de confirmación “Se cerrará el sistema. ¿Desea continuar?”, si es aceptado, el sistema termina su ejecución.</td></tr>
<tr><td colspan="2">Postcondiciones</td></tr>
<tr><td colspan="2">Queda habilitada la conexión a la BD.</td></tr>
</table>

Tabla 2.4 Descripción del CU Conectar BD

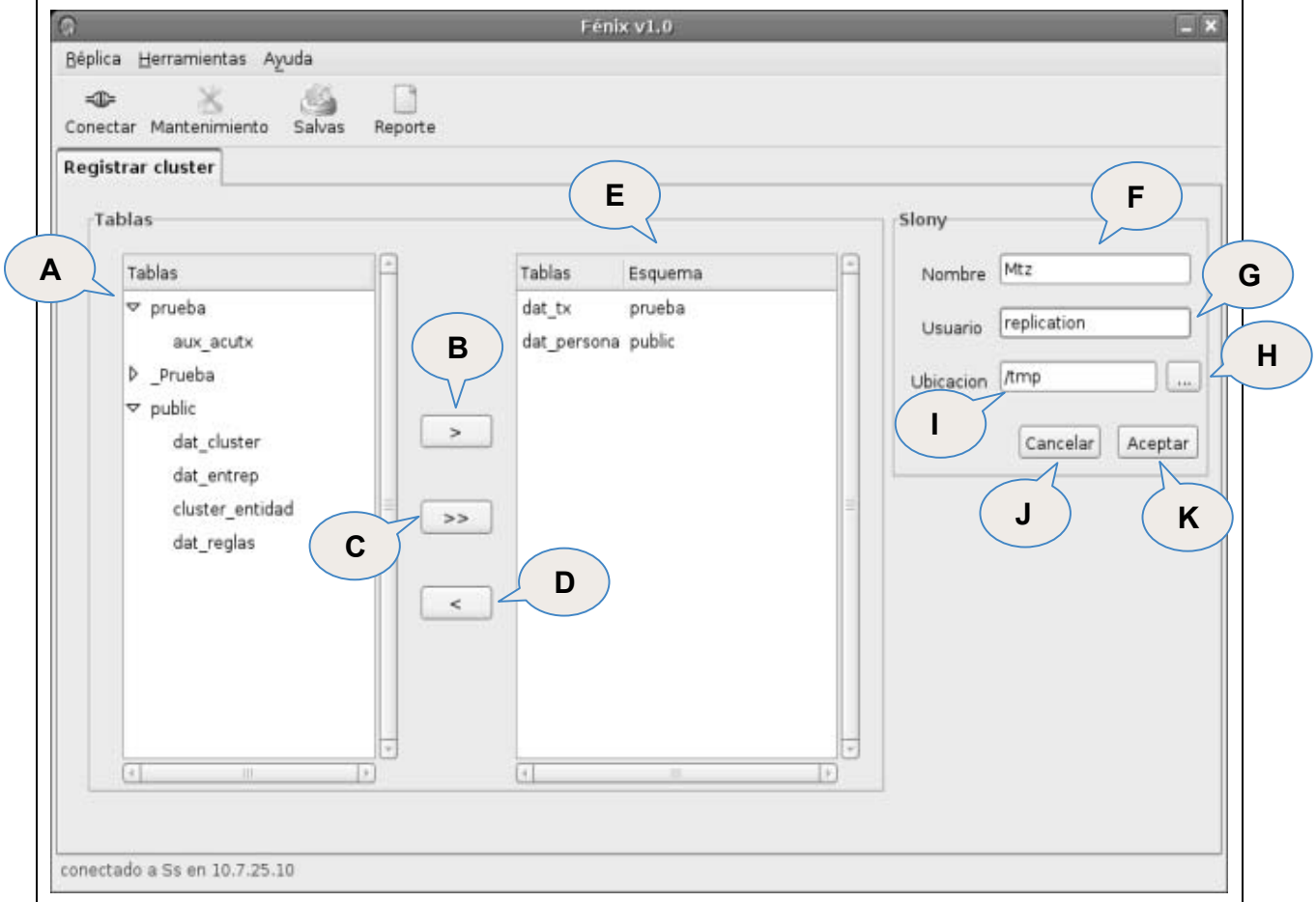
Caso de Uso	Nombre del caso de uso
CU-2	Configurar réplica
Actores	Administrador del servidor de BD.
Propósito	Configurar la réplica para la BD seleccionada.

Resumen

El caso de uso se inicia cuando el actor selecciona Espejo en el menú Réplica. El sistema muestra la interfaz de configuración de la réplica, el actor selecciona las BD y realiza las operaciones necesarias, configurando la réplica para la BD seleccionada.

Referencias	R1, 1.1, R2, 2.2, R2, 2.3, R3, 3.1, R3, 3.2, R3, 3.3, R3, 3.6
Precondiciones	El usuario debe haberse conectado a las BD (master y esclava).

Interfaz I



Descripción

A: TreeView para mostrar las tablas de la BD por esquemas.

B: Button para seleccionar una tabla e incluirla a la configuración de la réplica.

C: Button para seleccionar todas las tablas e incluirlas a la configuración de la réplica.

D: Button para no incluir una tabla seleccionada de la configuración de la réplica.

E: TreeView para listar las tablas seleccionadas de la BD con su esquema correspondiente.

F: Entry para especificar el nombre del cluster de Slony a crearse.

G: Entry para especificar el usuario de réplica para conectarse a la BD.

H: Button para especificar el lugar del disco duro a guardar los ficheros de configuración de Slony.

I: Entry para mostrar la ubicación de los ficheros de la configuración de Slony.

J: Button para cancelar la acción de configurar la réplica de espejos.

K: Button para aceptar la acción de configurar la réplica de espejos.

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1. Selecciona Espejo en el menú Réplica.	2. El sistema muestra la Interfaz I.
3. El actor selecciona las tablas de la BD a replicar y oprime el botón **>**.	
4. El actor introduce los datos necesarios para completar la configuración (nombre del cluster, usuario de réplica, escoge la ubicación de los ficheros de configuración) y oprime el botón Aceptar.	5. El sistema genera los ficheros de Slony a partir de los datos introducidos.
6. El actor oprime el botón Cancelar.	7. El sistema muestra la ventana principal.

Cursos Alternos

Curso Normal

Línea 1

- Si el actor desea configurar una réplica de fragmentos **ver Sección Réplica de Fragmentos**.

Línea 3

- Si el actor desea seleccionar todas las tablas y oprime el botón **>>**, el sistema incluirá todas las tablas en la configuración de réplica.
- Si el actor desea deseleccionar una tabla, la selecciona y oprime el botón **<**, el sistema no incluirá la tabla en la configuración de réplica.

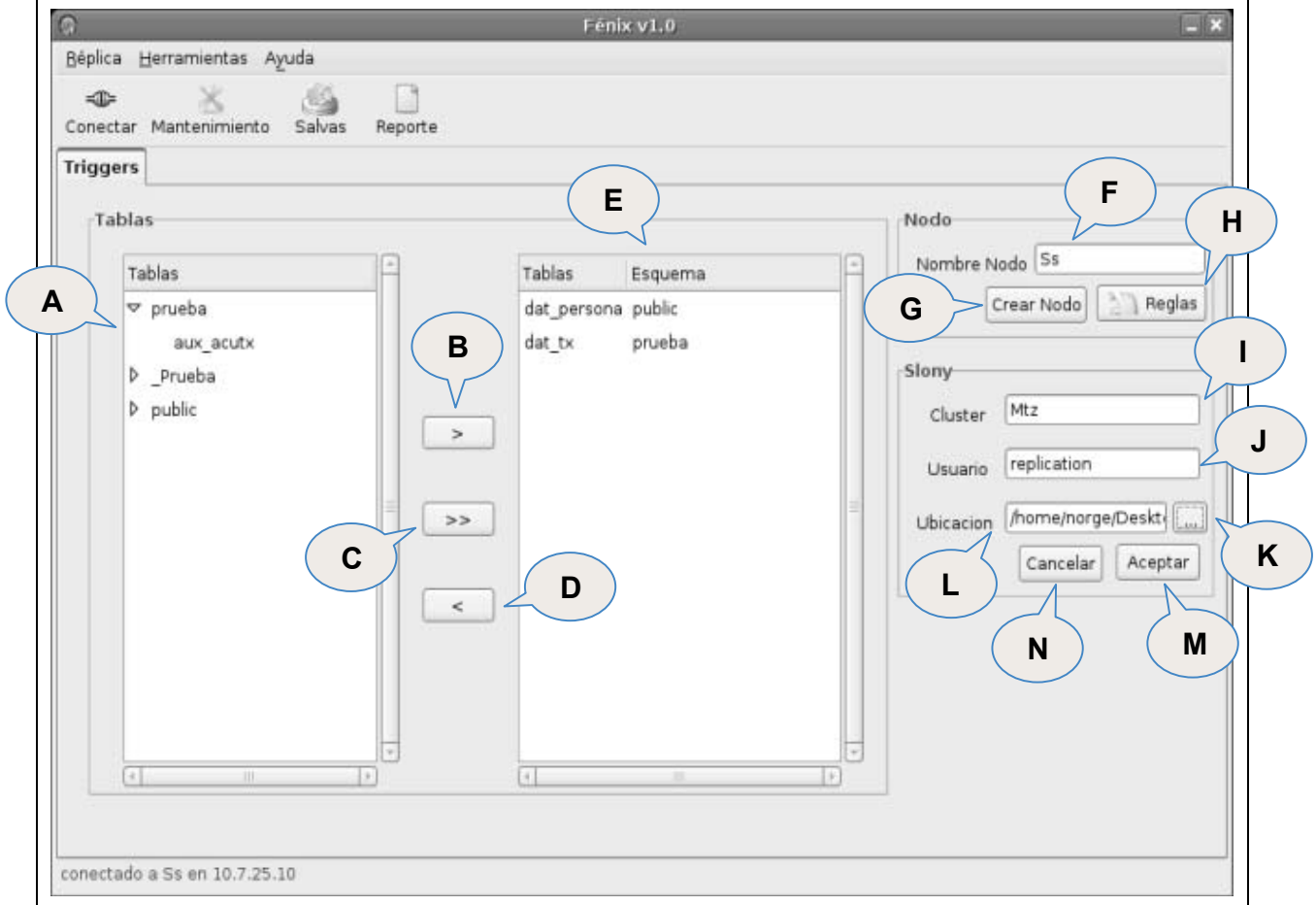
Línea 4

Si el actor omite un dato requerido o no selecciona alguna tabla a incluir en la configuración y oprime el botón Aceptar, el sistema muestra un mensaje “Existen campos vacíos. Debe

seleccionar alguna tabla".

Sección Réplica de Fragmentos

Interfaz II



Descripción

A: TreeView para mostrar las tablas de la BD por esquemas.

B: Button para seleccionar una tabla e incluirla a la configuración de la réplica.

C: Button para seleccionar todas las tablas e incluirlas a la configuración de la réplica.

D: Button para no incluir una tabla seleccionada de la configuración de la réplica.

E: TreeView para listar las tablas seleccionadas de la BD con su esquema correspondiente.

F: Entry para especificar el nombre del nodo a crearse.

G: Button para crear el nodo con los datos especificados anteriormente.

H: Button para especificar reglas al nodo creado.

I: .Entry para especificar el nombre del cluster de Slony a crearse.

J: Entry para especificar el usuario con acceso a la réplica.

K: Button para especificar el lugar del disco duro a guardar los ficheros de configuración de Slony.

L: Entry para mostrar la ubicación de los ficheros de la configuración de Slony.

M: Button para aceptar la acción de configurar la réplica de espejos.

N: Button para cancelar la acción de configurar la réplica de espejos.

Acción del Actor	Respuesta del Sistema
1. El actor selecciona Fragmentos en el menú de Réplica.	2. El sistema muestra la Interfaz II.
3. Se repite la línea 3 del flujo normal de los eventos.	
4. El actor especifica el nombre del nodo a crear y oprime el botón Aceptar.	5. El sistema emite el mensaje “Se creará el nodo. ¿Desea continuar?”.
6. El actor acepta la acción de registrar un nuevo nodo.	7. El sistema crea el nodo en la BD para configurar la réplica.
3. Se repite desde la línea 4 hasta la línea 7del flujo normal de los eventos.	

Cursos Alternos

Curso Normal

Línea 4

- Si el actor omite el nombre del nodo y oprime el botón Aceptar, el sistema muestra el mensaje “Especifique el nombre el nombre del nodo”.
- Si el actor introduce un nodo que ya existe y oprime el botón Aceptar, el sistema muestra el mensaje “Ya existe un nodo con ese nombre”.
- Si el actor al crear el nodo no ha seleccionado ninguna tabla para incluir en la configuración de la réplica, el sistema muestra el mensaje “Debe seleccionar alguna tabla”.

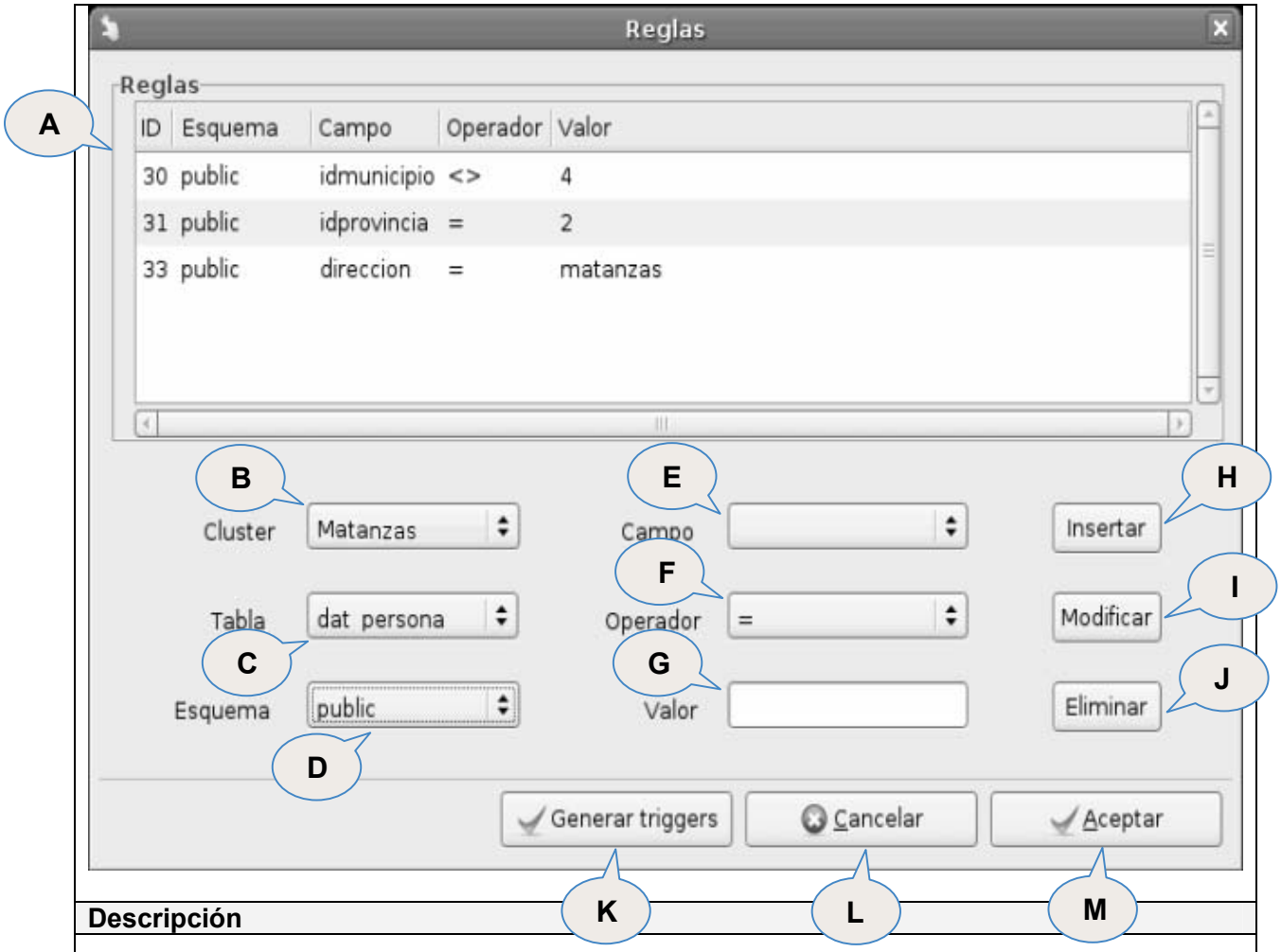
Línea 6

- Si el actor no acepta la creación del nodo, el sistema continúa mostrando la Interfaz I.
- Si el actor desea especificar reglas de replicación para las tablas seleccionadas ver

CU Definir reglas de replicación.
Postcondiciones
− Queda configurada la réplica de datos. − El administrador pone en funcionamiento la réplica.
Puntos de Extensión
− Línea 6 de la **Sección Réplica de Fragmentos** el actor define reglas. Ver CU Definir reglas de replicación.

Tabla 2.5 Descripción del CU Configurar réplica

Caso de Uso	**Nombre del caso de uso**
CU-3	Definir reglas de replicación
Actores	Administrador del servidor de BD
Propósito	Configurar las reglas para filtrar los datos a replicar.
Resumen El caso de uso se inicia para dar respuesta a la solicitud de otro CU. El sistema muestra la interfaz para la gestión de las reglas, el actor realiza las operaciones necesarias, dejando las reglas definidas según las necesidades. El CU termina cuando el actor genera los triggers a partir de las reglas de replicación.	
Referencias	R1, 1.1, R3, 3.4, R3, 3.4.1, R3, 3.4.2, R3, 3.4.3, R3, 3.5
CU relacionados	El **CU** es una extensión del **CU Configurar réplica**
Precondiciones	El usuario debe haberse conectado a las BD (master y esclava).
Interfaz I	



Descripción

A: TreeView para mostrar las reglas de la entidad seleccionada del cluster y el esquema especificado.

B: Combobox para seleccionar el cluster deseado.

C: Combobox para seleccionar la entidad seleccionada del cluster especificado.

D: Combobox para seleccionar el esquema de la entidad.

E: Combobox para seleccionar el campo de la entidad a efectuarle la regla.

F: Combobox para seleccionar el operador de la regla.

G: Entry para especificar el valor del campo al crear la regla.

H: Button para insertar una nueva regla a la entidad.

I: Button para modificar una regla existente en la entidad.

J: Button para eliminar una regla de la entidad.

K: Button para aplicar los cambios realizados.

L: Button para cancelar la acción de la creación de reglas para la réplica.

M: Button para aceptar la acción de configurar las reglas para las entidades seleccionadas.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra la Interfaz I.
2. El actor selecciona el cluster que desea.	3. El sistema le da la lista de tablas pertenecientes a ese cluster.
4. El actor selecciona la tabla que desea.	5. El sistema le da la lista de los esquemas donde se encuentra la tabla.
6. El actor selecciona el esquema al que pertenece la tabla deseada.	7. El sistema muestra las reglas que estén creadas para esa tabla.
8. El actor decide la acción a realizar: 1. Insertar 2. Modificar **(ver Sección Modificar reglas)** 3. Eliminar **(ver Sección Eliminar reglas)**	
9. El actor introduce los datos para definir la regla (campo, operador y valor) y presiona Insertar.	10. El sistema registra la regla y la muestra en el control A.
11. El actor presiona Generar triggers.	12. El sistema crea los triggers en las BD seleccionadas.
13. El actor presiona Aceptar.	14. Abandona el CU regresando al CU Configurar réplica.

Cursos Alternos

Curso Normal

Línea 11

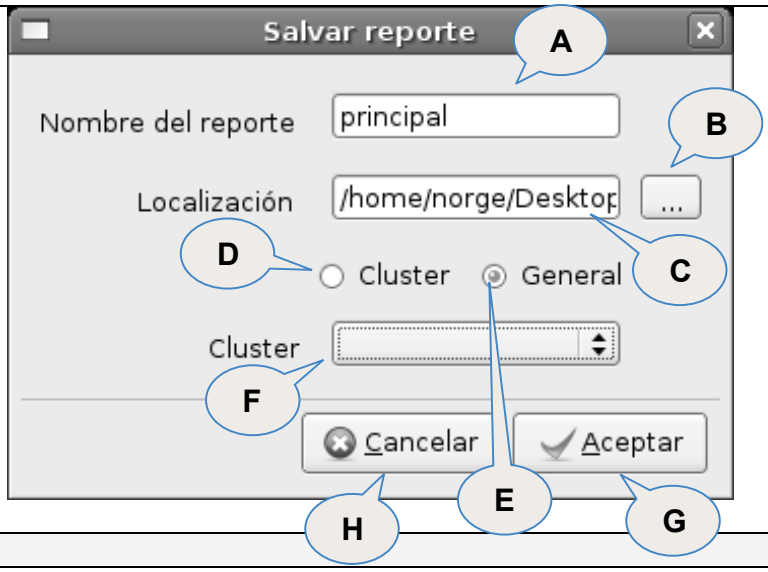
- Si el actor no desea generar los triggers en ese momento, ir a la línea 13 del curso normal de los eventos.

Sección Modificar reglas	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la regla a modificar presionando con doble clic sobre ella.	2. El sistema muestra los datos en los controles E, F y G.
3. El actor modifica el valor actual de la regla	4. El sistema registra el cambio y lo muestra

y presiona Modificar.	en el control A.
5. Se repite desde la línea 11 del curso normal hasta la 14.	
Cursos Alternos	
Curso Normal **Línea 5** − Si el actor no desea generar los triggers en ese momento, ir a la línea 13 del curso normal de los eventos.	
Sección Eliminar reglas	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la regla a modificar presionando con un clic sobre ella.	2. El sistema muestra la regla seleccionada en el control A.
3. El actor presiona Eliminar.	4. El sistema elimina la regla asociada a la tabla seleccionada y no la muestra en el control A.
5. Se repite desde la línea 11 del curso normal hasta la 14.	
Cursos Alternos	
Curso Normal **Línea 5** − Si el actor no desea generar los triggers en ese momento, ir a la línea 13 del curso normal de los eventos.	
Postcondiciones	
− Las reglas quedan definidas − Los triggers están creados para la configuración de réplica.	

Tabla 2.6 Descripción del CU Definir reglas de replicación

Caso de Uso	**Nombre del caso de uso**
CU-4	Mostrar reporte de réplica
Actores	Administrador del servidor de BD
Propósito	Reporte del estado de la configuración de réplica.
Resumen El caso de uso se inicia cuando el actor selecciona Reporte. El sistema permite generar dos tipos de reporte, por cluster o general. En el primero describe la configuración del cluster especificado y en el otro la situación de todos los clusters.	
Referencias	R1, 1.1, R4, 4.1, R4, 4.2, R4, 4.3, R4, 4.4, R4, 4.5
Precondiciones	El usuario debe haberse conectado a las BD maestra.
Interfaz I	



Descripción

A: Entry para especificar el nombre del reporte para visualizar la configuración de la réplica.

B: Button para especificar el lugar del disco duro a guardar el reporte.

C: Entry para especificar la ubicación del reporte.

D: RadioButton para especificar si el reporte es por cluster.

E: RadioButton para especificar si el reporte es general.

F: Combobox para seleccionar el cluster del cual se creará el reporte.

G: Button para cancelar la acción de generar el reporte.

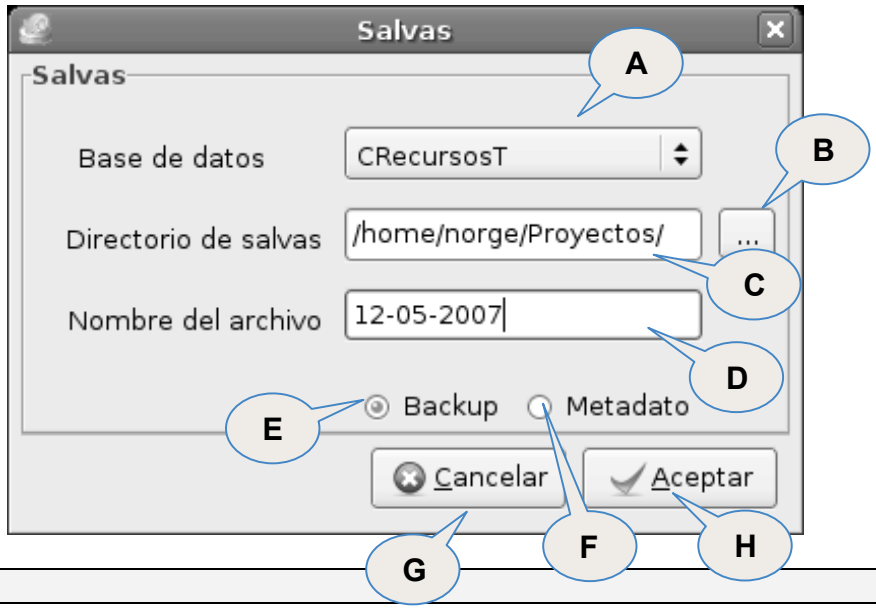
H: Button para aceptar la acción de generar el reporte.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona Reporte.	2. El sistema muestra la Interfaz I.
3. El actor introduce los datos necesarios para generar el reporte (nombre del reporte, la ubicación del reporte).	
4. El actor mantiene el reporte por la opción de Cluster y presiona el botón Aceptar.	5. El sistema crea el reporte como una página Web y lo muestra a continuación.
	6. El sistema muestra la ventana principal.
Cursos Alternos	
Curso Normal **Línea 4**	

– Si el actor escoge el reporte por la opción General se deshabilita el control F para escoger el cluster. – Si el actor escoge el reporte por la opción General y presiona el botón Aceptar continúan las acciones a partir de la línea 5 del flujo normal. – Si el actor oprime el botón Cancelar, el sistema muestra la ventana principal.
Postcondiciones
– El reporte está listo para ser analizado.

Tabla 2.6 Descripción del CU Mostrar reporte de réplica

Caso de Uso	**Nombre del caso de uso**
CU-6	Crear copias de seguridad de BD
Actores	Administrador del servidor de BD
Propósito	Crear copias de seguridad de la BD.
Resumen El caso de uso se inicia cuando el actor selecciona Salvas. El sistema permite realizar copias de seguridad de dos maneras, por Backup o Metadato según se necesite.	
Referencias	R1, 1.1, R5, 5.1, R5, 5.2, R5, 5.3
Precondiciones	El usuario debe haberse conectado a las BD.
Interfaz I	



Descripción

A: Combobox para seleccionar la base de datos requerida.

B: Button para especificar el lugar del disco duro a guardar el fichero de salva de la BD.

C: Entry para mostrar la ubicación del fichero de salva de la BD.

D: Entry para especificar el nombre del fichero de salva de la BD.

E: RadioButton para especificar si el tipo de salva es Backup.

F: RadioButton para especificar si el tipo de salva es Metadato.

G: Button para cancelar la acción de salvar la BD seleccionada.

H: Button para aceptar la acción de salvar la BD seleccionada.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona Salvas.	2. El sistema muestra la Interfaz I.
3. El actor introduce los datos necesarios (Selecciona BD, directorio de salvas y nombre del archivo).	
4. El actor mantiene la salva por la opción de Backup y presiona el botón Aceptar.	5. El sistema realiza la salva de seguridad de la BD seleccionada en un fichero.
	6. El sistema muestra la ventana principal.

Cursos Alternos

Curso Normal

Línea 4

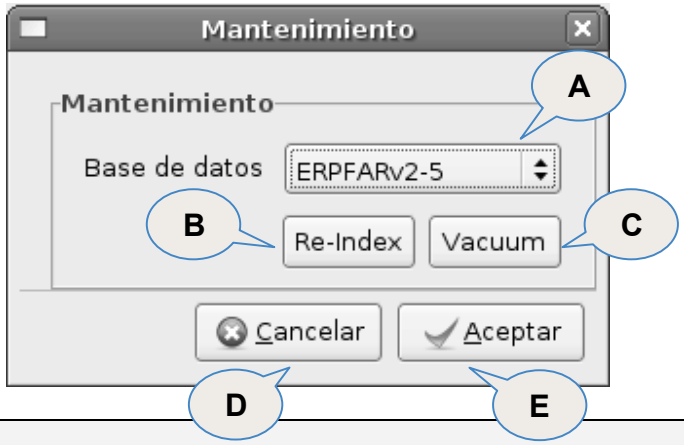
- Si el actor escoge la salva por la opción de Metadato y presiona el botón Aceptar continúan las acciones a partir de la línea 5 del flujo normal.
- Si el actor oprime el botón Cancelar, el sistema muestra la ventana principal.

Postcondiciones

- Es generado el archivo de salvas de la BD seleccionada.

Tabla 2.6 Descripción del CU Crear copias de seguridad de BD

Caso de Uso	**Nombre del caso de uso**
CU-7	Realizar mantenimiento a la BD
Actores	Administrador del servidor de BD
Propósito	Realizar mantenimiento a las BD
Resumen El caso de uso se inicia cuando el actor selecciona Mantenimiento. El sistema garantiza una mayor eficiencia y rendimiento de la BD seleccionada luego del mantenimiento por cualquiera de los criterios Re-Index, Vacumm.	
Referencias	R1, 1.1, R6, 6.1, R6, 6.2, R6, 6.3
Precondiciones	El usuario debe haberse conectado a las BD.
Interfaz I	



Descripción

A: Combobox para seleccionar la base de datos requerida.

B: Button para realizar el mantenimiento por Re-Index.

C: Button para realizar el mantenimiento por Vacuum.

D: Button para cancelar la acción de realizar el mantenimiento a la BD.

E Button para aceptar la acción de realizar el mantenimiento a la BD.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona Mantenimiento.	2. El sistema muestra la Interfaz I.
3. El actor selecciona la BD a darle mantenimiento.	
4. El actor presiona el botón Re-Index como tipo de mantenimiento a realizar.	5. El sistema realiza el mantenimiento solicitado.
6. El actor presiona el botón Aceptar.	7. El sistema muestra la ventana principal.

Cursos Alternos

Curso Normal

Línea 4

- Si el actor omite la selección de la BD y oprime el botón Re-Index, el sistema muestra un mensaje “Debe seleccionar la BD”.
- Si el actor presiona el botón Vacuum como tipo de mantenimiento a realizar, continúan las acciones a partir de la línea 5 del flujo normal.
- Si el actor omite la selección de la BD y oprime el botón Vacuum, el sistema muestra un mensaje “Debe seleccionar la BD”.

Línea 6 – Si el actor omite la selección de la BD y oprime el botón Aceptar, el sistema muestra un mensaje “Debe seleccionar la BD”. – Si el actor oprime el botón Cancelar, el sistema muestra la ventana principal.
Postcondiciones
– La BD seleccionada es más eficiente en su desempeño.

Tabla 2.7 Descripción del CU Realizar mantenimiento a las BD

2.5 Conclusiones

En este capítulo se comenzó a desarrollar la propuesta de solución a partir del estudio realizado de los procesos de negocio. Se definieron los requisitos que debe cumplir el sistema en toda su totalidad, así como los actores del mismo, incluyendo la descripción de cada una de sus funcionalidades. Partiendo de todas las características que se han expuesto, se está en condiciones de construir el sistema siguiendo sus especificidades.

Capítulo 3 Análisis y diseño del sistema

3.1 Introducción

En el presente capítulo se realiza detalladamente el análisis y diseño del sistema desde el punto de vista del proceso de desarrollo de software utilizado, lo que será de gran ayuda en la compresión de la realización del sistema. Se presentan igualmente el modelo de datos, los estándares de interfaz, los principios de diseño, concepción general de la ayuda, el tratamiento de errores, entre otros.

3.2 Análisis

3.2.1 Diagrama de Clases de Análisis

Las clases de análisis están centradas en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Dichas clases tienen atributos y existen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos entre ellas. Estas clases se clasifican en:

- *Entidad*: Modelan información que posee larga vida y que es a menudo persistente.
- *Interfaz*: Modelan la interacción entre el sistema y sus actores.
- *Control*: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

El diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema, como parte del mundo real, no desde el punto de vista de la construcción del sistema.

A continuación se relacionan los diagramas de clases de análisis, asociados al problema en cuestión.

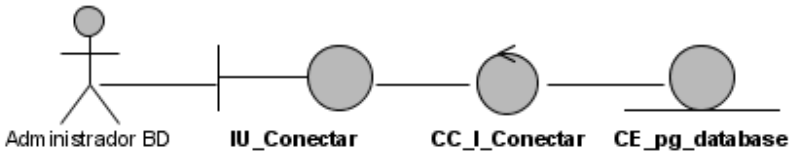


Figura 3.1 DCA. CU Conectar BD

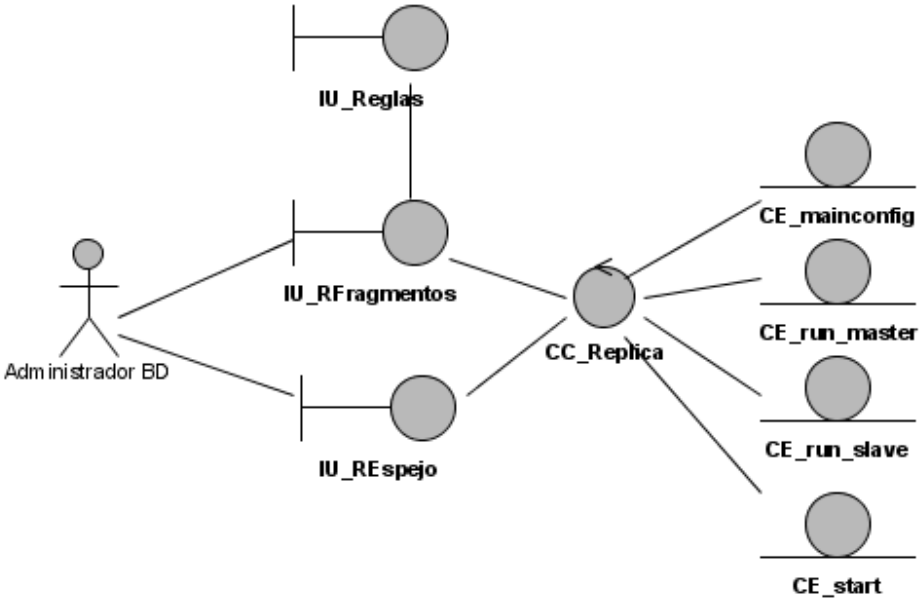


Figura 3.2 DCA. CU Configurar réplica

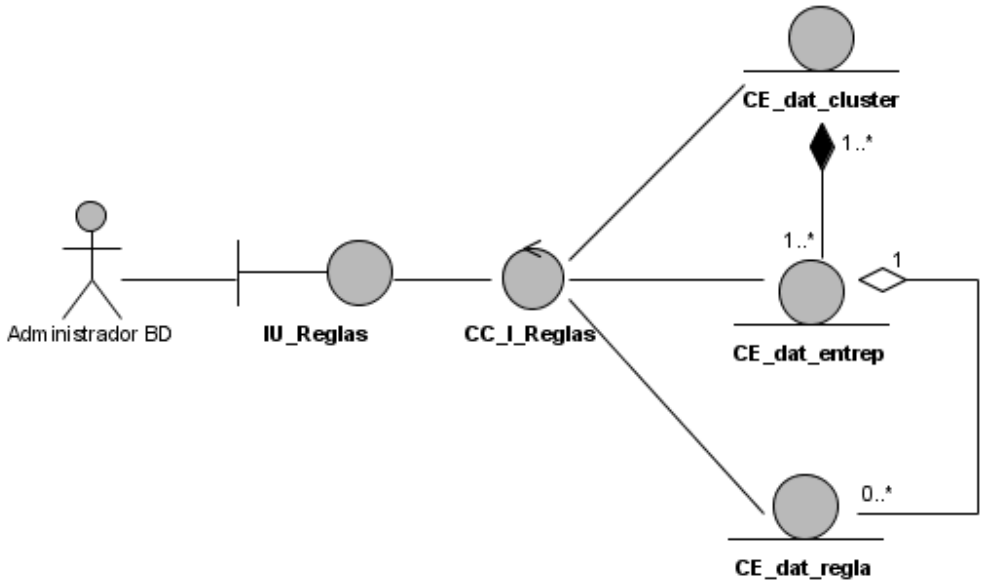


Figura 3.3 DCA. CU Configurar reglas de replicación

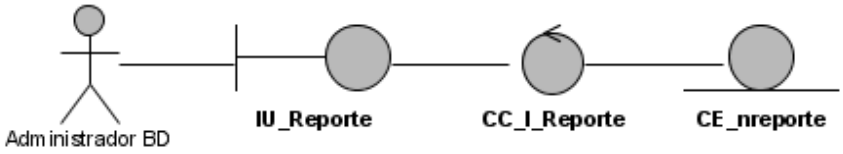


Figura 3.4 DCA. CU Mostrar reporte de réplica

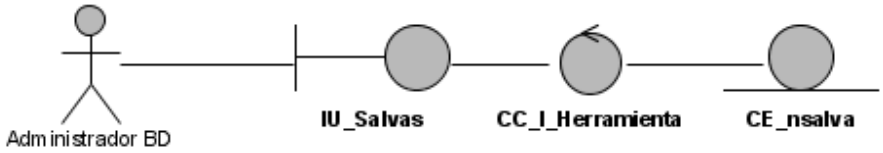


Figura 3.5 DCA. CU Crear copias de seguridad de BD

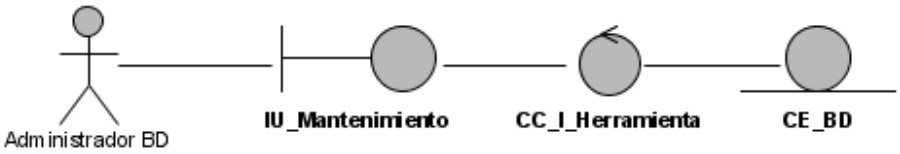


Figura 3.6 DCA. CU Realizar mantenimiento a BD

3.3 Diseño

3.3.1 Diagramas de Interacción

Los *diagramas de interacción* (diagramas de secuencia o diagramas de colaboración) explican gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas.

Para representar gráficamente ésta interacción se escogió el diagrama de colaboración para una mayor claridad.

Diagramas de Colaboración

El Diagrama de Colaboración presenta una alternativa al diagrama de secuencia para modelar interacciones entre objetos en el sistema. Mientras que el diagrama de secuencia se centra en la secuencia cronológica del escenario que estamos modelando, el diagrama de colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario.

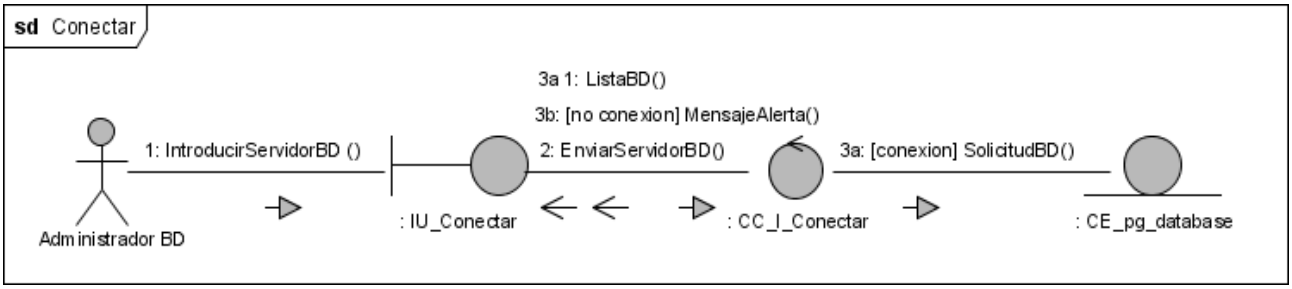


Figura 3.7 CU Conectar BD

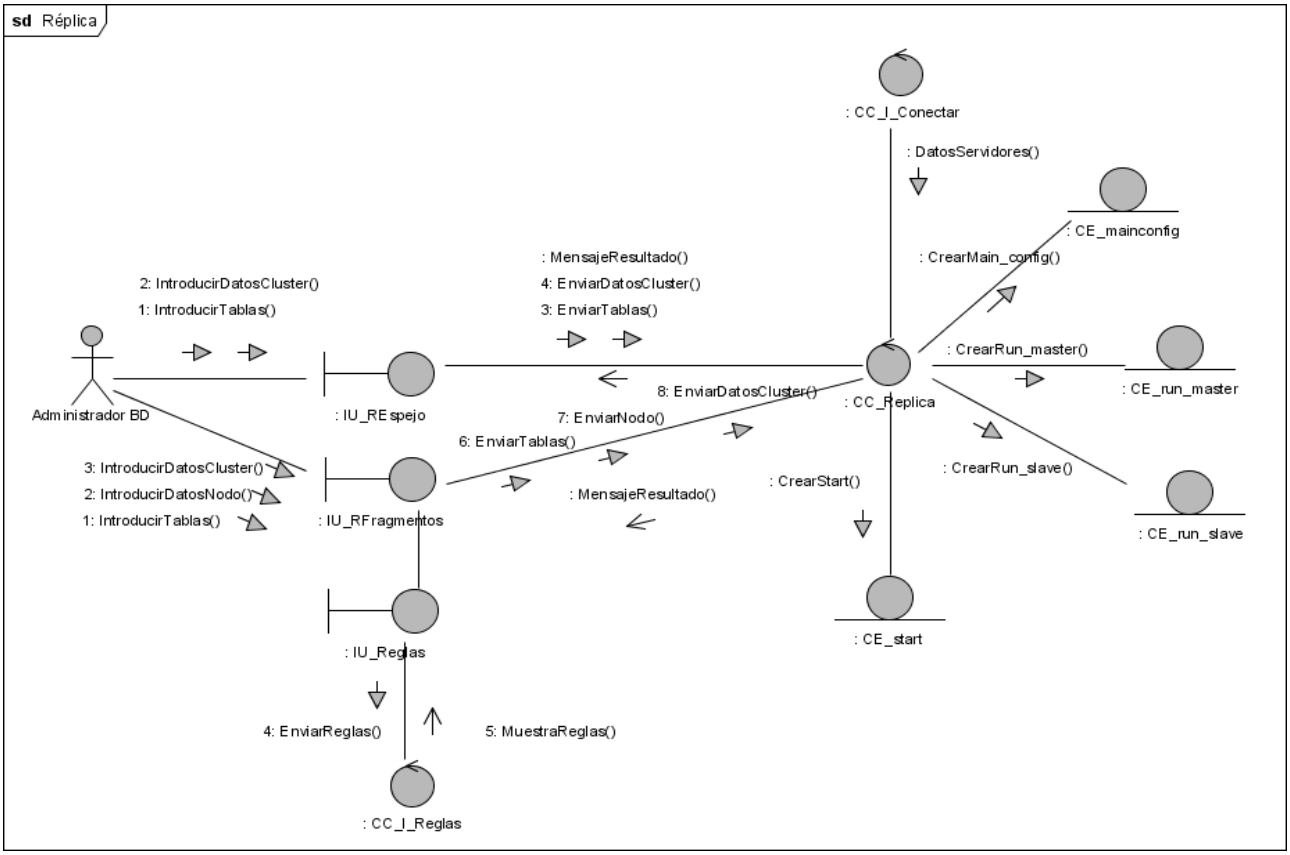


Figura 3.8 CU Réplica de Espejo

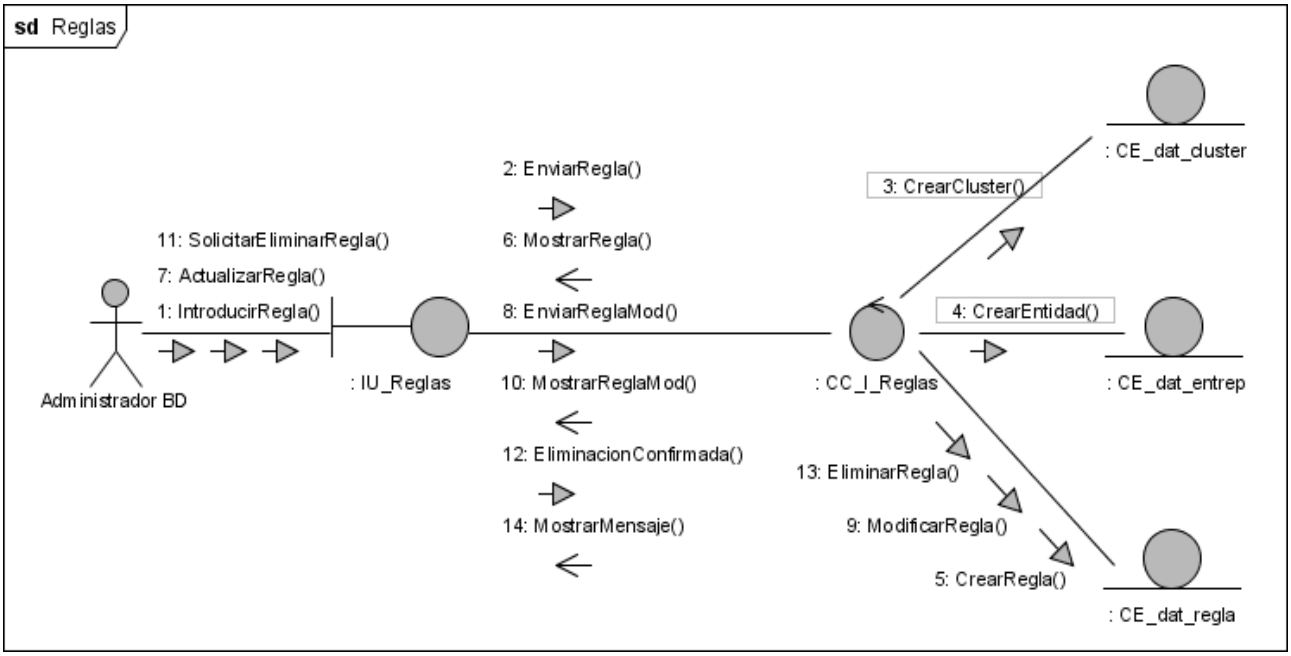


Figura 3.9 CU Reglas de replicación

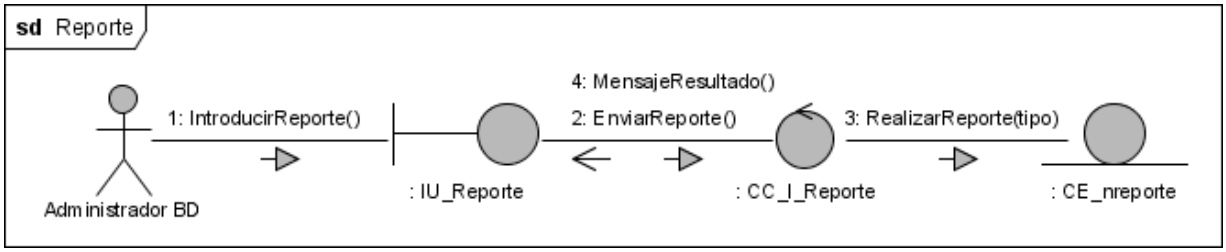


Figura 3.10 CU Mostrar estado de réplica

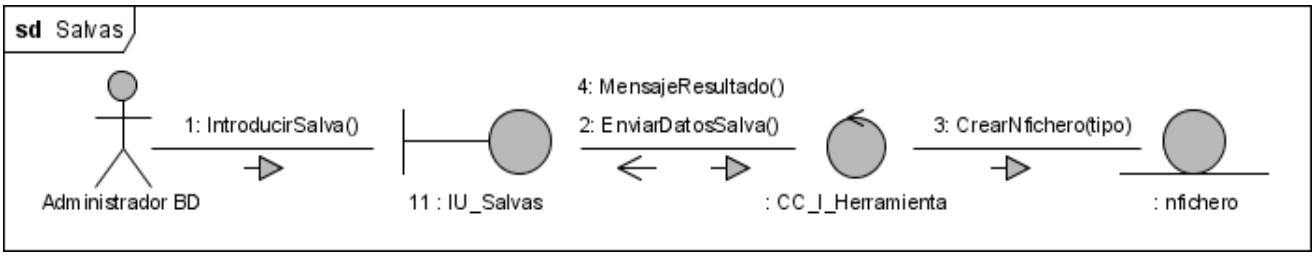


Figura 3.11 CU Crear copias de seguridad de BD

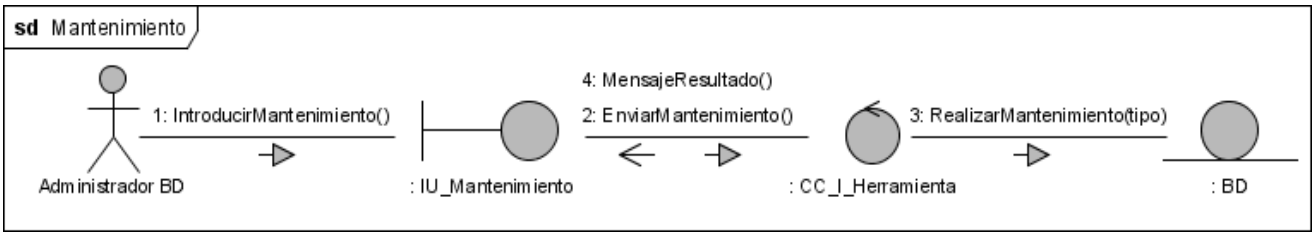


Figura 3.12 CU Realizar mantenimiento a las BD

3.3.2 Diagrama de Clases

Un Diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. La definición de clase incluye definiciones para atributos y operaciones.

Los Diagramas de Clases por definición son estáticos, esto es, representan que partes interactúan entre sí.

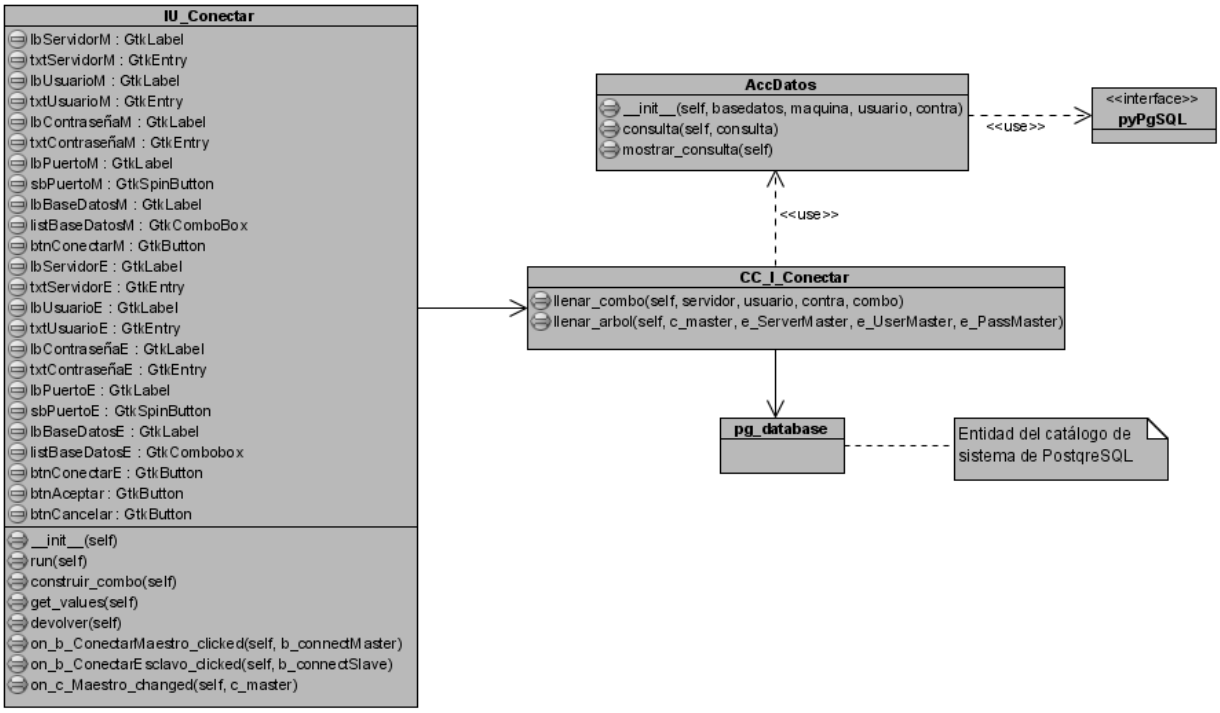


Figura 3.13 DC Conectar BD

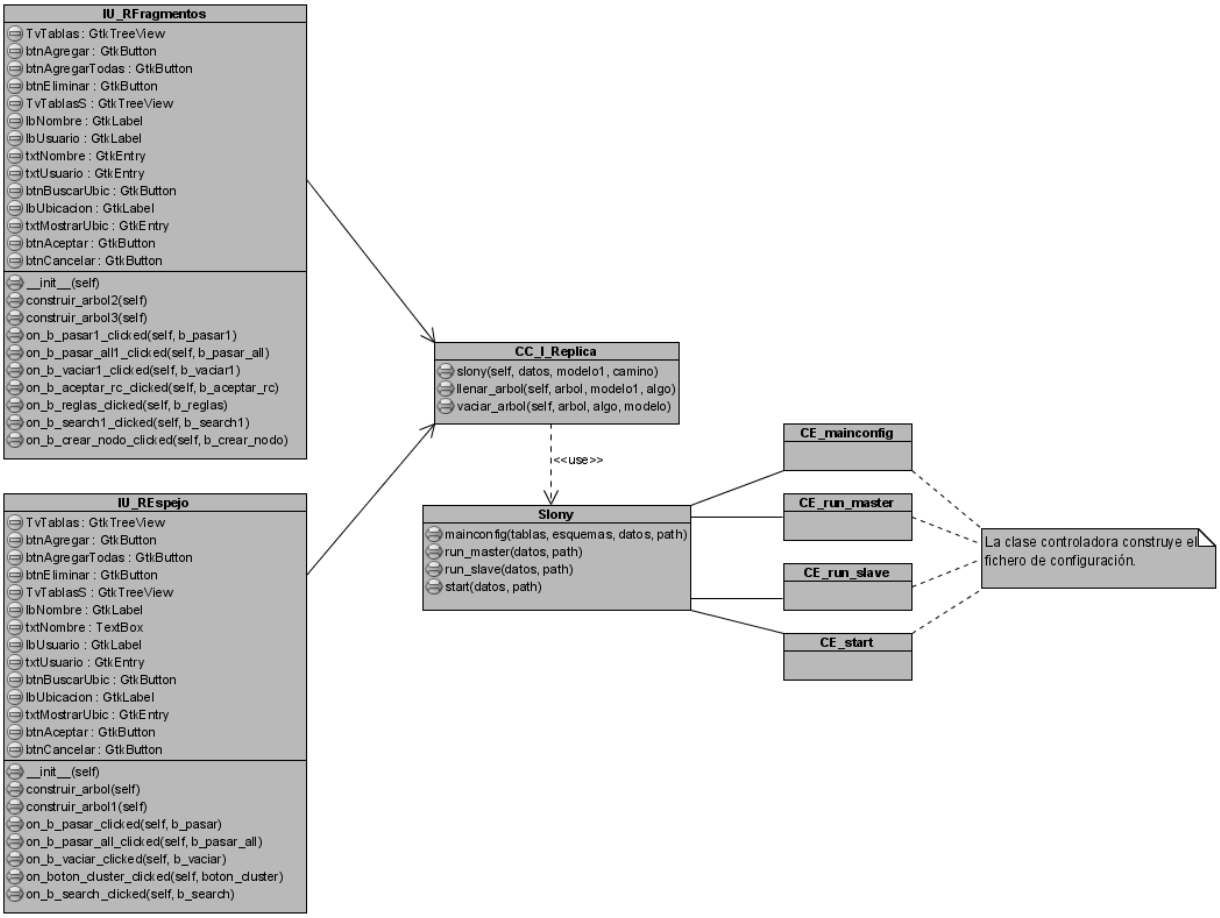


Figura 3.14 DC Configurar réplica

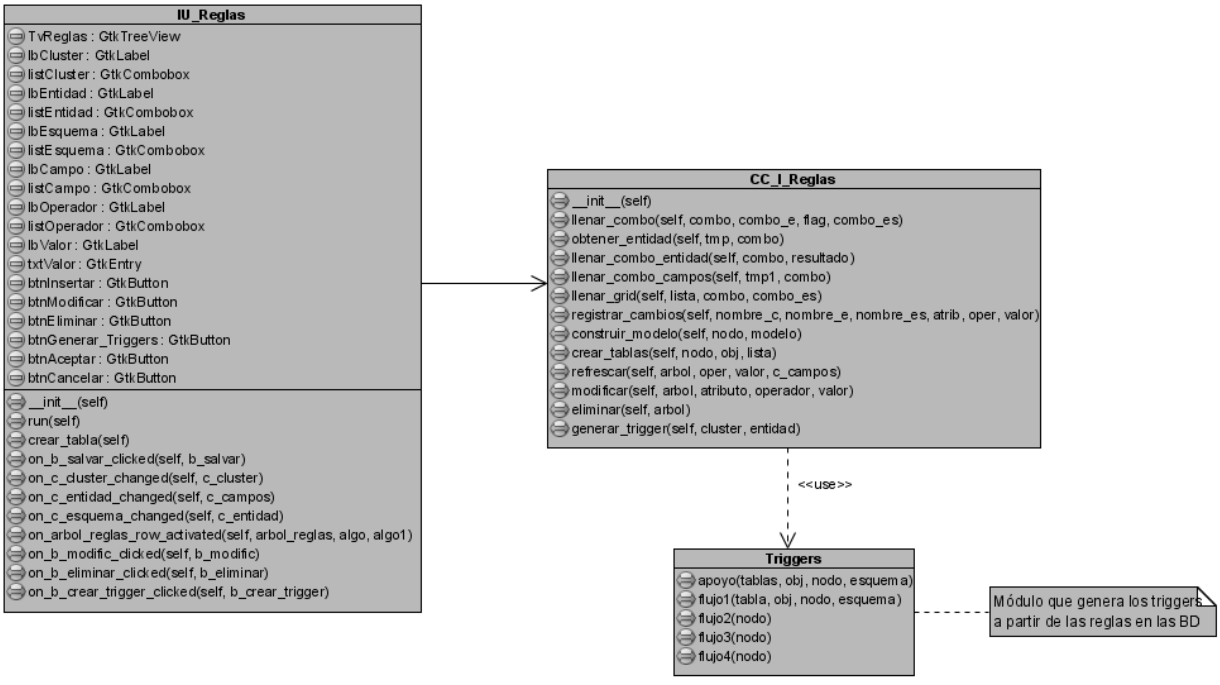


Figura 3.15 DC Configurar reglas de replicación

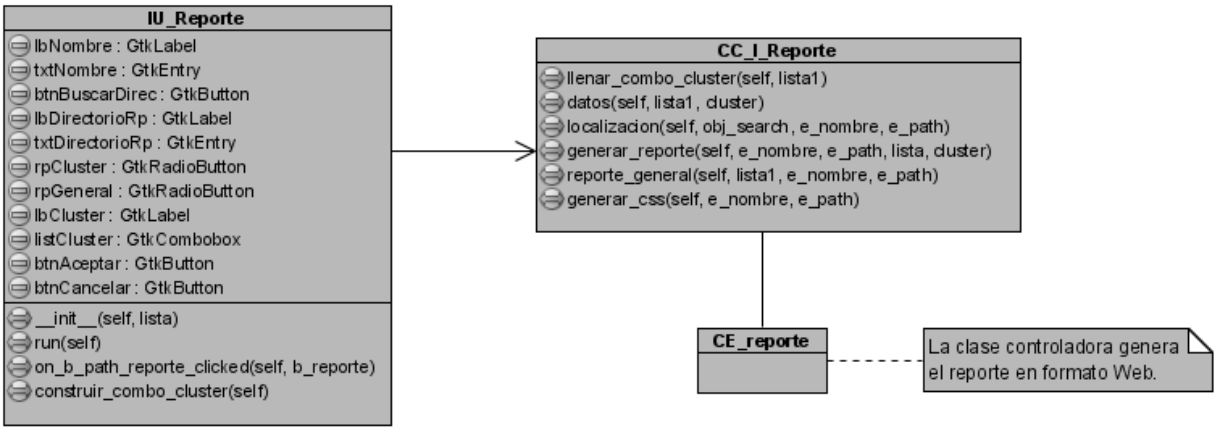


Figura 3.16 DC Mostrar reporte de réplica

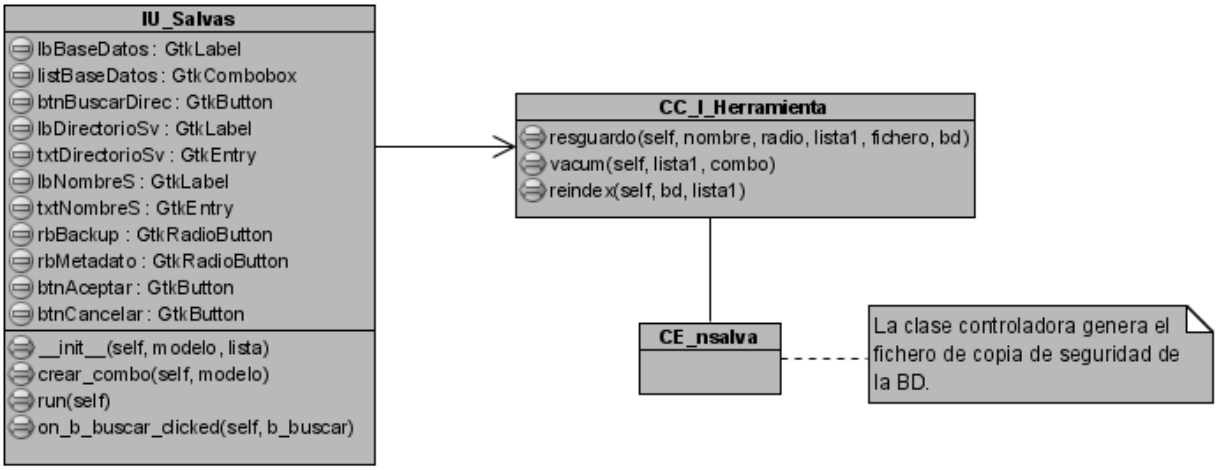


Figura 3.17 DC Crear copias de seguridad de BD

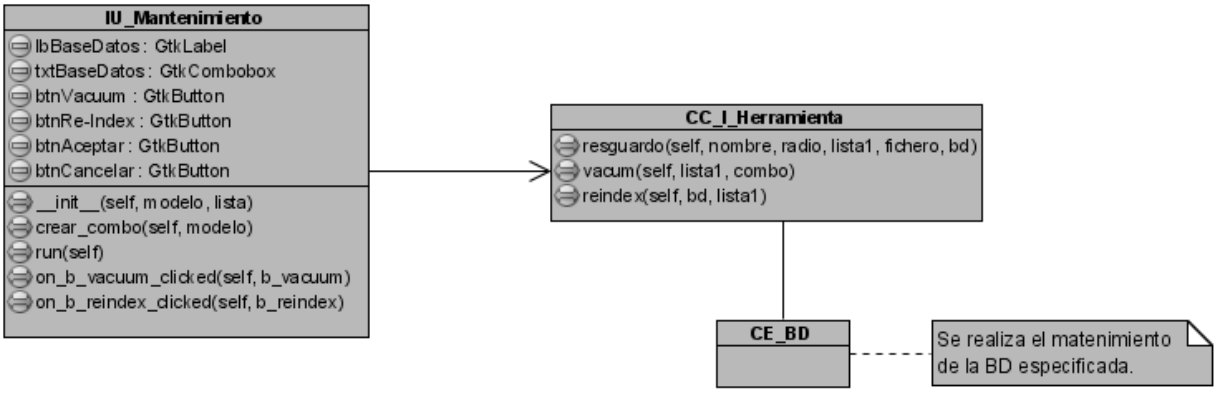


Figura 3.18 DC Realizar mantenimiento a las BD

3.3.2.1 Descripción de las clases

A continuación se describen las clases que conforman el sistema:

Nombre: IU_Conectar	
Tipo de clase: Interfaz	
Atributo	**Tipo**
lbServidorM	GtkLabel
txtServidorM	GtkEntry
lbUsuarioM	GtkLabel
txtUsuarioM	GtkEntry
lbContraseñaM	GtkLabel
txtContraseñaM	GtkEntry
lbPuertoM	GtkLabel
sbPuertoM	GtkSpinButton
lbBaseDatosM	GtkLabel
listBaseDatosM	GtkComboBox
btnConectarM	GtkButton
lbServidorE	GtkLabel
txtServidorE	GtkEntry
lbUsuarioE	GtkLabel
txtUsuarioE	GtkEntry
lbContraseñaE	GtkLabel
txtContraseñaE	GtkEntry
lbPuertoE	GtkLabel
sbPuertoE	GtkSpinButton
lbBaseDatosE	GtkLabel
listBaseDatosE	GtkCombobox
btnConectarE	GtkButton
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	__init__(self)
Descripción:	Contructor de la clase.
Nombre:	run(self)
Descripción:	Dibuja la interfaz.
Nombre:	construir_combo(self)
Descripción:	Contruye el control para la lista de BD.
Nombre:	get_values(self)
Descripción:	Devuelve los valores de los atributos de la clase.
Nombre:	devolver(self)
Descripción:	Permite acceder a las variables de una clase a otra.
Nombre:	on_b_ConectarMaestro_clicked(self, b_connectMaster)
Descripción:	Conectar servidor maestro.
Nombre:	on_b_ConectarEsclavo_clicked(self, b_connectSlave)
Descripción:	Conectar servidor esclavo.
Nombre:	on_c_Maestro_changed(self, c_master)
Descripción:	Refrescar selección del combobox.

Tabla 3.1 Descripción clase IU_Conectar

Nombre: CC_I_Conectar	
Tipo de clase: Controladora	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	llenar_combo(self, servidor, usuario, contra, combo)
Descripción:	Método para mostrar las entidades de la BD.
Nombre:	llenar_arbol(self, c_master, e_ServerMaster, e_UserMaster, e_PassMaster)
Descripción:	Muestra las entidades de la BD seleccionadas por esquema.

Tabla 3.2 Descripción clase CC_l_Conectar

Nombre: AccDatos	
Tipo: Módulo para el acceso a los datos	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	__init__(self, basedatos, maquina, usuario, contra)
Descripción:	Establece la conexión a la BD seleccionada.
Nombre:	consulta(self, consulta)
Descripción:	Define consulta a realizar.
Nombre:	mostrar_consulta(self)
Descripción:	Muestra resultado de la consulta.

Tabla 3.3 Descripción clase AccDatos

Nombre: IU_REspejo	
Tipo de clase: Interfaz	
Atributo	**Tipo**
TvTablas	GtkTreeView
btnAgregar	GtkButton
btnAgregarTodas	GtkButton
btnEliminar	GtkButton
TvTablasS	GtkTreeView
lbNombre	GtkLabel
txtNombre	TextBox
lbUsuario	GtkLabel
txtUsuario	GtkEntry
btnBuscarUbic	GtkButton
lbUbicacion	GtkLabel
txtMostrarUbic	GtkEntry
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	construir_arbol(self)
Descripción:	Crea el árbol de donde se seleccionarán las entidades a replicar.
Nombre:	construir_arbol1(self)
Descripción:	Crea el árbol hacia donde irán las entidades seleccionadas para la réplica.
Nombre:	on_b_pasar_clicked(self,b_pasar)
Descripción:	Pasa la tabla seleccionada del árbol hacia el árbol1.
Nombre:	on_b_pasar_clicked(self,b_pasar)
Descripción:	Pasa la tabla seleccionada del árbol hacia el árbol1.
Nombre:	on_b_vaciar_clicked(self,b_vaciar)
Descripción:	Regresa las entidades desde el árbol1 hacia el árbol.
Nombre:	on_boton_cluster_clicked(self,boton_cluster)
Descripción:	Llama a la función que genera los ficheros de configuración de slony.
Nombre:	on_b_cancelar_espejo_clicked(self,b_cancelar_espejo)

Descripción:	Oculta la interfaz de la réplica en espejo.
Nombre:	on_b_pasar_all_clicked(self,b_pasar_all)
Descripción:	Pasa todas las entidades desde árbol hacia árbol1.

Tabla 3.4 Descripción clase IU_REspejo

Nombre: IU_RFragmento	
Tipo de clase: Interfaz	
Atributo	**Tipo**
TvTablas	GtkTreeView
BtnAgregar	GtkButton
BtnAgregarTodas	GtkButton
BtnEliminar	GtkButton
TvTablasS	GtkTreeView
LbNombre	GtkLabel
LbUsuario	GtkLabel
TxtNombre	GtkEntry
TxtUsuario	GtkEntry
BtnBuscarUbic	GtkButton
LbUbicacion	GtkLabel
TxtMostrarUbic	GtkEntry
BtnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	construir_arbol2(self)
Descripción:	Construye el árbol desde donde se seleccionan las entidades para replicar.
Nombre:	construir_arbol3(self)
Descripción:	Construye el árbol hacia donde se pasan las tablas que se replicarán.
Nombre:	on_b_pasar1_clicked(self, b_pasar1)
Descripción:	Pasa la entidad seleccionada en árbol2 hacia árbol3.
Nombre:	on_b_pasar_all1_clicked(self, b_pasar_all)
Descripción:	Pasa todas las entidades desde árbol2 hacia árbol3.
Nombre:	on_b_vaciar1_clicked(self, b_vaciar1)
Descripción:	Pasa las entidades seleccionadas en árbol3 hacia árbol2.
Nombre:	on_b_aceptar_rc_clicked(self, b_aceptar_rc)
Descripción:	Llama a la función que genera los ficheros de configuración de slony.
Nombre:	on_b_reglas_clicked(self, b_reglas)
Descripción:	Muestra la ventana de configuración de las reglas de replcación.
Nombre:	on_b_search1_clicked(self, b_search1)
Descripción:	Muestra la ventana para seleccionar la ubicación de los ficheros de configuración de slony.
Nombre:	on_b_crear_nodo_clicked(self, b_crear_nodo)
Descripción:	Llama a la función que crea el nodo de replicación.

Tabla 3.5 Descripción clase IU_RFrafmento

Nombre: CC_I_Replica	
Tipo de clase: Controladora	
Atributo	**Tipo**
(2)	(3)
Para cada responsabilidad:	
Nombre:	llenar_arbol1(self,arbol,arbol1)
Descripción:	Función que llena el árbol con las entidades de la base de datos maestra.
Nombre:	vaciar_arbol(self,arbol,modelo_arbol,modelo)
Descripción:	Función que pasa las entidades desde el árbol1 hacia el árbol original.

Tabla 3.6 Descripción clase CC_I_Replica

Nombre: IU_Reglas	
Tipo de clase: Interfaz	
Atributo	**Tipo**
TvReglas	GtkTreeView
lbCluster	GtkLabel
listCluster	GtkCombobox
bEntidad	GtkLabel
listEntidad	GtkCombobox
lbEsquema	GtkLabel
listEsquema	GtkCombobox
lbCampo	GtkLabel
listCampo	GtkCombobox
lbOperador	GtkLabel
listOperador	GtkCombobox
lbValor	GtkLabel
txtValor	GtkEntry
btnInsertar	GtkButton
btnModificar	GtkButton
btnEliminar	GtkButton
btnGenerar_Triggers	GtkButton
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	run(self)
Descripción:	Muestra la interfaz.
Nombre:	crear_tabla(self)
Descripción:	Crea el grid donde se mostrarán las reglas.
Nombre:	on_b_salvar_clicked(self, b_salvar)
Descripción:	Llama al método que guarda las reglas dentro de la base de datos.
Nombre:	on_c_cluster_changed(self, c_cluster)
Descripción:	Refresca la lista de las entidades que se están resplicando en el cluster seleccionado.
Nombre:	on_c_entidad_changed(self, c_campos)
Descripción:	Refresca la lista de los esquemas a los cuales pertenece la entidad seleccionada.
Nombre:	on_c_esquema_changed(self, c_entidad)
Descripción:	Refesca la lista de las reglas que contiene la entidad seleccionada.
Nombre:	on_arbol_reglas_row_activated(self, arbol_reglas, algo, algo1)
Descripción:	Llama al método que captura los datos de la fila seleccionada.
Nombre:	on_b_modific_clicked(self, b_modific)
Descripción:	Llama al metodo que actualiiza la regla seleccionada.
Nombre:	on_b_eliminar_clicked(self, b_eliminar)
Descripción:	Elimina la regla seleccionada.
Nombre:	on_b_crear_trigger_clicked(self, b_crear_trigger)
Descripción:	Llama a la función que genera los triggers en dependencia de los datos seleccionados en la interfaz.

Tabla 3.9 Descripción clase IU_Reglas

Nombre: CC_I_Reglas	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	apoyo(tablas, obj, nodo, esquema)
Descripción:	Función que va ejecuta los métodos que generarán los ficheros de configuración

	de slony.
Nombre:	flujo1(tabla, obj, nodo, esquema)
Descripción:	Método que crea el trigger que se encarga del flujo #1 del modelo de réplica.
Nombre:	flujo2(nodo)
Descripción:	Método que crea el trigger que se encarga del flujo #2 del modelo de réplica.
Nombre:	flujo3(nodo)
Descripción:	Método que crea el trigger que se encarga del flujo #3 del modelo de réplica.
Nombre:	flujo4(nodo)
Descripción:	Método que crea el trigger que se encarga del flujo #4 del modelo de réplica.
Nombre:	flujo5(nodo)
Descripción:	Método que crea el trigger que se encarga del flujo #5 del modelo de réplica.

Tabla 3.10 Descripción clase CC_I_Reglas

Nombre: IU_Salvas	
Tipo de clase: Interfaz	
Atributo	**Tipo**
lbBaseDatos	GtkLabel
listBaseDatos	GtkCombobox
btnBuscarDirec	GtkButton
lbDirectorioSv	GtkLabel
txtDirectorioSv	GtkEntry
lbNombreS	GtkLabel
txtNombreS	GtkEntry
rbBackup	GtkRadioButton
rbMetadato	GtkRadioButton
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	__init__(self, modelo, lista)
Descripción:	Costructor de la clase.
Nombre:	crear_combo(self, modelo)
Descripción:	Método para mostrar las entidades de la BD.
Nombre:	run(self)
Descripción:	Dibuja la interfaz.
Nombre:	on_b_buscar_clicked(self, b_buscar)
Descripción:	Método para definir la ubicación de la salva.

Tabla 3.14 Descripción clase IU_Salvas

Nombre: CC_I_Herramienta	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	resguardo(self, nombre, radio, lista1, fichero, bd)
Descripción:	Método para realizar la copia de seguridad de la BD.
Nombre:	vacum(self, lista1, combo)
Descripción:	Método para realizar el mantenimiento por Vacuum.
Nombre:	reindex(self, bd, lista1)
Descripción:	Método para realizar el mantenimiento por ReIndex.

Tabla 3.15 Descripción clase CC_I_Herramienta

Nombre: IU_Mantenimeinto	
Tipo de clase: Interfaz	
Atributo	**Tipo**

lbBaseDatos	GtkLabel
listBaseDatos	GtkCombobox
btnVacuum	GtkButton
btnReIndex	GtkButton
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	__init__(self, modelo, lista)
Descripción:	Costructor de la clase.
Nombre:	crear_combo(self, modelo)
Descripción:	Método para mostrar las entidades de la BD.
Nombre:	run(self)
Descripción:	Dibuja la interfaz.
Nombre:	on_b_vacuum_clicked(self, b_vacuum)
Descripción:	Método que llama al que realiza el Vacuum.
Nombre:	on_b_reindex_clicked(self, b_reindex)
Descripción:	Método que llama al que realiza el ReIndex.

Tabla 3.16 Descripción clase IU_Mantenimiento

Nombre: IU_Reporte	
Tipo de clase: Interfaz	
Atributo	**Tipo**
lbNombre	GtkLabel
txtNombre	GtkEntry
btnBuscarDirec	GtkButton
lbDirectorioRp	GtkLabel
txtDirectorioRp	GtkEntry
rpCluster	GtkRadioButton
rpGeneral	GtkRadioButton
lbCluster	GtkLabel
listCluster	GtkCombobox
btnAceptar	GtkButton
btnCancelar	GtkButton
Para cada responsabilidad:	
Nombre:	__init__(self, lista)
Descripción:	Costructor de la clase.
Nombre:	run(self)
Descripción:	Dibuja la interfaz.
Nombre:	on_b_path_reporte_clicked(self, b_reporte)
Descripción:	Método que llama al que define la ubicación del reporte.
Nombre:	construir_combo_cluster(self)
Descripción:	Método que llama para mostrar los cluster de la BD.

Tabla 3.17 Descripción clase IU_Reporte

Nombre: CC_I_Reporte	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	llenar_combo_cluster(self, lista1)
Descripción:	Método para mostrar los clusters de la BD.
Nombre:	datos(self, lista1, cluster)
Descripción:	Método que aporta los datos a la plantilla del reporte.
Nombre:	localizacion(self, obj_search, e_nombre, e_path)

Descripción:	Método para definir la ubicación del reporte.
Nombre:	generar_reporte(self, e_nombre, e_path, lista, cluster)
Descripción:	Genera el reporte por cluster en estilo Web.
Nombre:	reporte_general(self, lista1, e_nombre, e_path)
Descripción:	Genera el reporte general en estilo Web.
Nombre:	generar_css(self, e_nombre, e_path)
Descripción:	Genera el estilo para el reporte.

Tabla 3.18 Descripción clase CC_I_Reporte

3.4 Diseño de la BD

3.4.1 Diagrama Entidad Relación de la BD

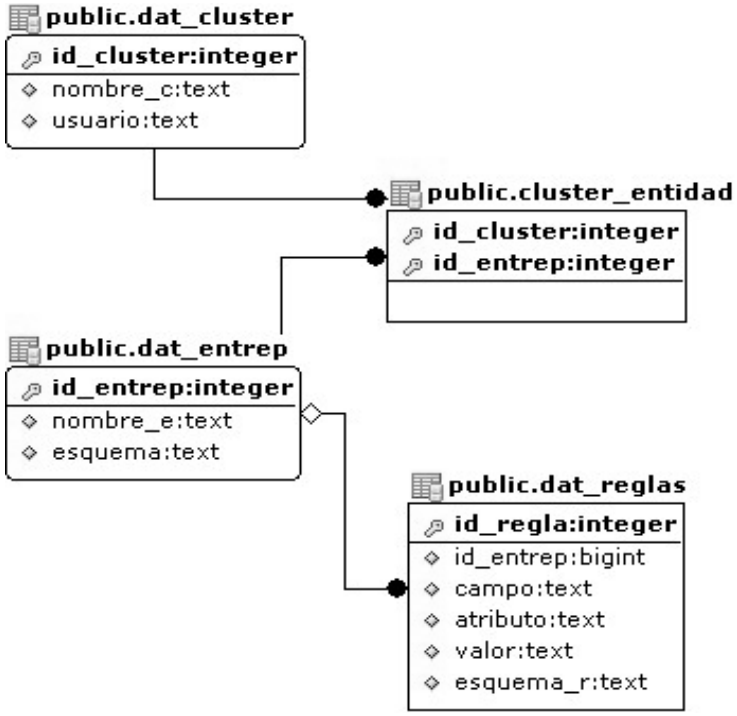


Figura 3.19 Diagrama Entidad Relación de la BD

El diagrama visto anteriormente resuelve la gestión de las reglas de replicación. Las tablas generadas por la propuesta de réplica para bases de datos fragmentadas (dat_tx, aux_acutx) son creadas en dependencia del nodo que se implique en la réplica.

Los triggers son creados en cada flujo de la propuesta como se explicó anteriormente con el objetivo de registrar los cambios ocurridos en las tablas que se seleccionen para la réplica, en caso de que sean definidas reglas, estás pasan a forman parte de él, ayudando en el filtrado de los datos a replicar.

A continuación se muestra un ejemplo de trigger creado dinámicamente según las condiciones del usuario.

```
--Trigger que registra los cambios ocurridos sobre la tabla que se va a replicar
CREATE OR REPLACE FUNCTION flujo1()
  RETURNS trigger AS $$
DECLARE
sql1 varchar;
BEGIN

if (TG_OP = 'INSERT')then
sql1 := 'insert into dat_persona
values('''||NEW.idpers||''','''||NEW.numid||''','''||NEW.nombre||''','''||NEW.papell||''','''||NEW.sapell||''','''||NEW.n
ompa||''','''||NEW.nomma||''','''||NEW.direccion||''','''||NEW.idprovincia||''','''||NEW.idmunicipio||''','''||NEW.rmo
dific||''')';
elsif (TG_OP = 'UPDATE')then
sql1 := 'update dat_persona set
idpers='''||NEW.idpers||''',numid='''||NEW.numid||''',nombre='''||NEW.nombre||''',papell='''||NEW.papell||''',sa
pell='''||NEW.sapell||''',nompa='''||NEW.nompa||''',nomma='''||NEW.nomma||''',direccion='''||NEW.direccion||'
'',idprovincia='''||NEW.idprovincia||''',idmunicipio='''||NEW.idmunicipio||''',rmodific='''||NEW.rmodific||''' where
idpers='''||OLD.idpers||'';
else
sql1 := 'delete from dat_persona where idpers='''||OLD.idpers||'';
end if;
if (TG_OP = 'INSERT')then
          if (NEW.idmunicipio <> 4 and NEW.idprovincia = 2 and NEW.direccion = matanzas)then
   insert into prueba.dat_tx (sql) values(sql1);
  end if;
elsif (TG_OP = 'UPDATE')then
 if (NEW.idmunicipio <> 4 and NEW.idprovincia = 2 and NEW.direccion = matanzas)then
   insert into prueba.dat_tx (sql) values(sql1);
 end if;
elsif (TG_OP = 'DELETE')then
 if (OLD.idmunicipio <> 4 and OLD.idprovincia = 2 and OLD.direccion = matanzas)then
   insert into prueba.dat_tx (sql) values(sql1);
 end if;
end if
;
return New;
END;
$$ language 'plpgsql';
```

3.4.2 Descripción de las tablas

Nombre: dat_cluster		
Descripción: Información sobre el cluster		
Atributo	**Tipo**	**Descripción**
id_cluster	integer	Identificador del cluster
nombre_c	text	Nombre del cluster
usuario	text	Supersuario con permiso para la réplica

Tabla 3.19 Descripción de la tabla dat_cluster

Nombre: dat_entrep		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_entrep	int	Identificador de la entidad
nombre_e	text	Nombre de la entidad
esquema	text	Nombre del esquema

Tabla 3.20 Descripción de la tabla dat_entrep

Nombre: dat_regla		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_regla	integer	Identificador de la regla
id_entrep	bigint	Identificador de la entidad
campo	text	Campo al que se le aplicará la regla
atributo	text	Operador que participa en la regla
valor	text	Valor que del campo para cumplir la regla
esquema_r	text	Esquema de réplica.

Tabla 3.21 Descripción de la tabla dat_regla

Nombre: cluster_entidad		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_cluster	integer	Identificador del cluster
id_entidad	integer	Identificador de la entidad

Tabla 3.22 Descripción de la tabla cluster_entidad

3.5 Principios de diseño

3.5.1 Tratamiento de errores

El tratamiento de excepciones o errores es un detalle considerado en cada acción ejecutada por el sistema. Se evitan de esta manera operaciones innecesarias y aumenta el tiempo útil disponible por el usuario.

En caso de que no se pueda continuar ejecutando la opción deseada porque una secuencia no se pueda realizar se emite un mensaje de alerta al usuario.

Figura 3.20 Mensaje de alerta

Se controla la manipulación de la información en la base de datos, informando al usuario en cada momento lo que ha ocurrido como resultado de su interacción con esta.

3.5.2 Estándares en la interfaz de la aplicación

La aplicación, a pesar de ser del tipo de escritorio y no tener tanta posibilidad como una Web para mostrar una interfaz colorida, presenta una interfaz intuitiva e interactiva que le guiará al usuario en las diferentes funcionalidades del sistema.

El sistema de ventanas que caracteriza este tipo de aplicaciones brinda ventajas en el diseño.

3.5.3 Concepción general de la ayuda

Para brindarle al usuario ayuda sobre el trabajo con el sistema en caso de dudas, el manual de usuario muestra una descripción de las acciones realizadas. Para acceder a la misma, puede hacerlo con la tecla caliente F1. Además se puede obtener una ayuda de los controles en la barra de estado del sistema y los mensajes en línea.

3.6 Conclusiones

En el presente capítulo se ha descrito la solución propuesta a través de los distintos artefactos que define RUP para la construcción de aplicaciones. Han quedado detalladas las clases y sus relaciones. Se definió el diagrama entidad-relación y las responsabilidades de cada uno de sus entidades, determinando además los estándares a utilizar.

Capítulo 4. Implementación y prueba

4.1 Introducción

En este capítulo se define como se va a desarrollar la aplicación, tanto la física mediante sus componentes en el desarrollo, como la tecnología a aplicar, así como la interrelación entre ambos, los casos de prueba establecidos dará una mayor seguridad de la eficiencia de la solución.

4.2 Implementación

4.2.1 Diagrama de despliegue

Muestra las relaciones entre el hardware y el software en el sistema final. Se representa como un grafo de nodos unidos por conexiones de comunicación.

Es la manera mediante la cual el sistema interactúa con el usuario. La representación se realizó teniendo en cuenta las dos maneras en la que puede interactuar el usuario, desde una estación de trabajo cliente, teniendo en cuenta que el servidor de datos no posee interfaz gráfica, lo cual se representa en los niveles ministerio y territorio. En el nivel provincia por el contrario, no se presenta una estación de trabajo cliente, pues el servidor si posee interfaz gráfica y el sistema residirá en el mismo servidor de datos. Este segundo escenario es el que se tiene hoy en todo el país, pero se debe llegar paulatinamente al primer escenario por lo que el diagrama no se limita en este sentido.

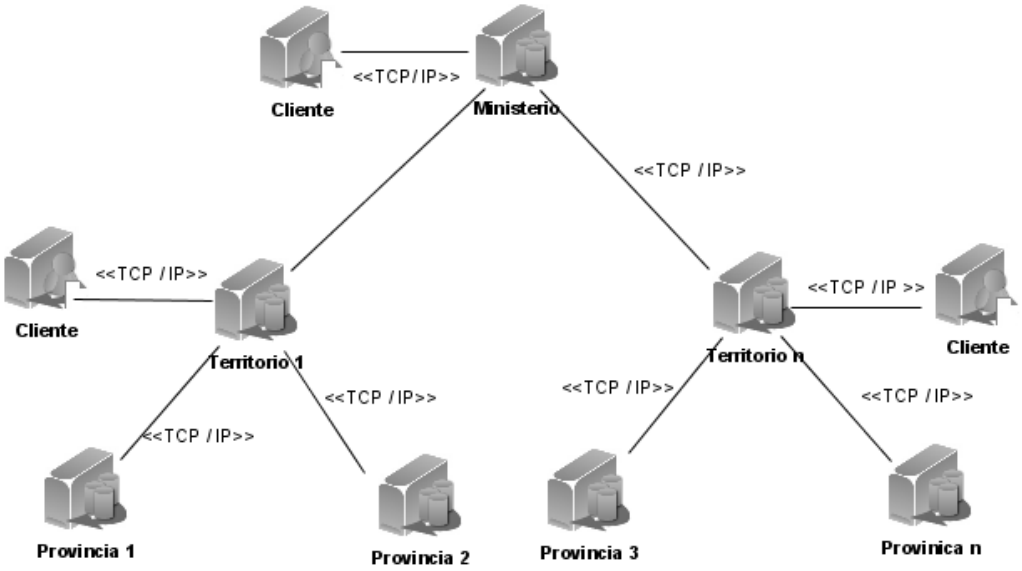


Figura 4.1 Diagrama de despliegue del sistema

4.2.2 Diagrama de componentes

Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia, pudiendo mostrarse las interfaces que estos soporten.

Es el conjunto de ficheros interrelacionados entre sí para lograr la completa funcionalidad del sistema.

A continuación se describen los componentes que conforman el diagrama de componentes del sistema:

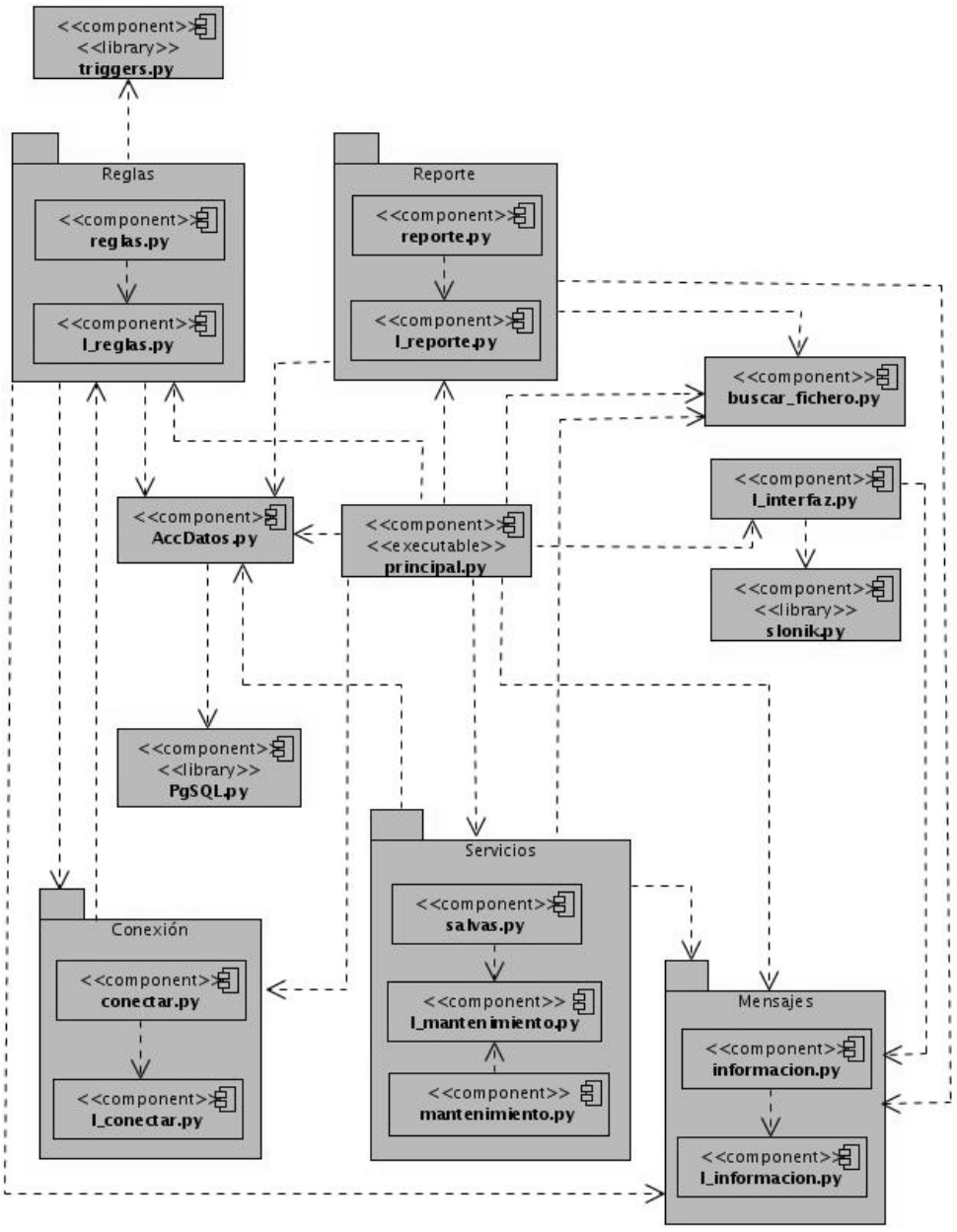


Figura 4.2 Diagrama de componentes del sistema

4.3 Modelo de prueba

La prueba de software es un proceso que corre en paralelo al proceso de desarrollo de software, y que se realiza por el convencimiento de que todo sistema debe ser “revisado” con el objetivo de establecer el nivel de calidad requerido. [21]

En ese proceso se ejecuta el sistema a probar bajo condiciones específicas y se le aplica un conjunto de estímulos diseñados ingenierilmente, con el objetivo de detectar insatisfacción de los requerimientos planteados; todas las actividades y sus resultados son documentados.

Nombre del caso de uso: Conectar BD

Entrada	**Resultados**	**Condiciones**
El actor introduce los datos (correctos) necesarios para conectarse al servidor de datos primario (Servidor, usuario, contraseña y puerto).	El sistema muestra la lista de las bases de datos existentes en el servidor.	Existe el servidor de BD.
El actor introduce los datos (incorrectos) necesarios para conectarse al servidor de datos primario (Servidor, usuario, contraseña y puerto) y oprime el botón Conectar.	El sistema muestra un mensaje “No se pudo realizar la operación, datos incorrectos”.	No existe el servidor de BD especificado.
El actor omite algunos de los datos necesarios para conectarse al servidor de datos primario (Servidor, usuario, contraseña y puerto) y oprime el botón Conectar.	El sistema muestra un mensaje “Se necesitan todos los datos”.	
El actor no especifica un servidor de datos esclavo.	El sistema no tendrá habilitadas las funcionalidades de la Réplica.	Dos servidores definidos, uno como maestro y el otro esclavo.
El actor no especifica un servidor maestro.	El sistema no tendrá habilitadas las funcionalidades de la	Dos servidores definidos, uno como

	Réplica.	maestro y el otro esclavo.
Si el actor oprime el botón Cancelar.	El sistema muestra un mensaje de confirmación "¿Está seguro de abandonar el sistema?"	

Tabla 4.1 Descripción de prueba para el CU Conectar BD

Nombre del caso de uso: Configurar réplica

Entrada	**Resultados**	**Condiciones**
El actor selecciona las tablas de la BD a replicar.	Se incluyen en la configuración de réplica de espejo.	
El actor selecciona todas las tablas y oprime el botón **>>**	El sistema incluirá todas las tablas en la configuración de réplica.	
El actor deseleccionar una tabla, la selecciona y oprime el botón **<**	El sistema no incluirá la tabla en la configuración de réplica.	
El actor introduce los datos necesarios para completar la configuración (nombre del cluster, usuario de réplica, escoge la ubicación de los ficheros de configuración) y oprime el botón Aceptar.	El sistema guarda los datos y genera los ficheros de réplica.	Seleccionar tablas para incluir en la configuración de réplica.
El actor omite un dato requerido y oprime el botón Aceptar.	El sistema muestra un mensaje "Se necesitan todos los datos".	

Tabla 4.2 Descripción de prueba para el CU Configurar réplica

Nombre del caso de uso: Definir reglas de replicación

Entrada	**Resultados**	**Condiciones**
El actor selecciona el cluster que desea.	El sistema le da la lista de entidades pertenecientes a ese cluster	Existen entidades registradas en el cluster.
El actor selecciona la entidad	El sistema le da la lista de los	Existen esquemas

que desea	esquemas donde se encuentra la entidad.	donde las entidades están creadas.
El actor selecciona el esquema al que pertenece la entidad deseada.	El sistema muestra las reglas que están creadas para esa entidad.	Existen reglas definidas para esa tabla.
El actor presiona Generar Triggers sin haber especificado las reglas de replicación.	El sistema muestra un mensaje “Debe especificar las reglas”.	Existen reglas definidas para esa tabla.

Tabla 4.3 Descripción de prueba para el CU Definir reglas de replicación

Nombre del caso de uso: Mostrar reporte de réplica

Entrada	**Resultados**	**Condiciones**
El actor introduce los datos necesarios para generar el reporte (nombre del reporte, la ubicación del reporte), mantiene el reporte por la opción de Cluster.	El sistema realiza el reporte configurado como una página Web.	Conexión habilitada a un servidor DB, específicamente a una BD.
El actor escoge el reporte por la opción General.	Se deshabilita el combobox para escoger el cluster.	Conexión habilitada a un servidor DB, específicamente a una BD.
El actor introduce los datos necesarios para generar el reporte (nombre del reporte, la ubicación del reporte), mantiene el reporte por la opción de General.	El sistema realiza el reporte configurado como una página Web.	Conexión habilitada a un servidor DB, específicamente a una BD.

Tabla 4.4 Descripción de prueba para el CU Mostrar reporte de réplica

Nombre del caso de uso: Crear copias de seguridad de la BD

Entrada	**Resultados**	**Condiciones**
El actor introduce los datos necesarios (Selecciona BD, directorio de salvas y nombre	El sistema realiza la salva de seguridad en un fichero.	Conexión habilitada a un servidor DB. Existe una BD

del archivo) y se mantiene por la opción de Backup y presiona el botón Aceptar.		seleccionada.
El actor escoge la salva por la opción de Metadato y presiona el botón Aceptar.	El sistema realiza la salva de seguridad en un fichero.	Conexión habilitada a un servidor DB. Existe una BD seleccionada.
El actor oprime el botón Cancelar.	El sistema muestra la ventana principal.	

Tabla 4.5 Descripción de prueba para el CU Crear copias de seguridad de la BD

Nombre del caso de uso: Realizar mantenimiento a BD

Entrada	**Resultados**	**Condiciones**
El actor selecciona la BD a darle mantenimiento y oprime el botón Re-Index como tipo de mantenimiento a realizar.	El sistema realiza el mantenimiento solicitado.	Conexión habilitada a un servidor DB.
El actor omite la selección de la BD y oprime cualquiera de los tipos de mantenimiento.	El sistema muestra un mensaje “Debe seleccionar la BD”.	Conexión habilitada a un servidor DB.
El actor presiona el botón Vacuum como tipo de mantenimiento a realizar.	El sistema realiza el mantenimiento solicitado.	Conexión habilitada a un servidor DB. Existe una BD seleccionada.

Tabla 4.6 Descripción de prueba para el CU Realizar mantenimiento a BD

4.4 Conclusiones

En el presente capítulo se ha quedado especificada la estructura física del sistema, así como la forma en que puede intercambiar la información con los usuarios, además se han definido los casos de prueba para garantizar la correcta funcionalidad del sistema.

Conclusiones

Para el desarrollo del presente trabajo de diploma se realizó un estudio de los procesos que intervienen en los flujos informativos de las FAR a través de varias entrevistas realizadas a especialistas relacionados con los mismos, conjuntamente con la búsqueda bibliográfica sobre el proceso de réplica en bases de datos distribuidas.

Se realizó una exhaustiva búsqueda bibliográfica acerca de las últimas tendencias y tecnologías que a nivel internacional se están utilizando en el mundo de la informática, para definir aquellas que mejor respuesta darían al problema al cual se quería dar solución, partiendo de la previa definición del gestor de base de datos PostgreSQL: el lenguaje Python y el PL/pgSQL para la escritura de los triggers.

Se seleccionó RUP como proceso de desarrollo de software a seguir para la construcción de la solución propuesta. Siguiendo este proceso, se diseñó e implementó la aplicación con la cual se propone aumentar la eficiencia y confiabilidad en el intercambio de información entre las estructuras de la institución.

Por tanto se considera que se han cumplido cada uno de los objetivos trazados al comenzar este trabajo, demostrándose además con el mismo que con la creación de un sistema que permita la configuración, gestión y mantenimiento de una política de réplica de datos basada en reglas dinámicamente configurables, será posible transmitir la información cumpliendo las normas establecidas y los plazos requeridos entre los servidores de datos de las FAR.

Recomendaciones

Este trabajo propone una solución para la gestión y configuración de réplica para las bases de datos distribuidas de las FAR, donde los servidores de datos utilizan el gestor de BD PostgreSQL sobre la plataforma GNU/Linux, por lo que se recomienda la implementación de una versión para servidores cuyo sistema operativo sea Windows, siempre manteniendo el SGBD.

Se recomienda además enriquecer el sistema con una funcionalidad que permita establecer las condiciones iniciales en el momento de la réplica de datos, lo que garantizará la integridad de la información.

Por otra parte, se considera que una aplicación de este tipo podría ser muy útil para cualquier institución interesada en respaldar sus datos o satisfacer otra necesidad a través de la réplica, por lo que se recomienda sea generalizada.

Bibliografía

Consultada

1. Olarte, C. “Bases de Datos Distribuidas”. 2005. [Consultado en enero del 2007]
Disponible en World Wide Web:
http://atlas.puj.edu.co/~caolarte/puj/cursos/cc100/files/clases/BDDistribuidas.pdf

2. “FAQ de PostgreSQL en castellano actualizado”. 2005.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.dbrunas.com.ar/article.php/7394.8400337425

3. Freedman, A. “Diccionario de Computación Bilingüe”. Editorial McGrawHill. Madrid, España. 1993.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.agapea.com/Diccionario-bilingue-de-Computacion-n9673i.htm

Citada

[1] Díaz, A. “Sistema de Bases de Datos Distribuidas”. [Consultado en enero del 2007]
Disponible en World Wide Web:
http://www.cs.cinvestav.mx/SC/prof_personal/adiaz/Disdb/Cap_1.html

[2] Date, C. J. “Introducción a los Sistemas de Bases de Datos”. 3ra parte. Ciudad de la Habana. Editorial Félix Varela, 2003.

[3] Aravena, S; Diaz, A; Machiavello, C; Nieto, A; Ortiz, E. “Estructura de Base de Datos Distribuidas”. DIICC Univ. De Concepción. Concepción, 2003. 56.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://asignaturas.inf.udec.cl/~basedato/trabajos/estructura_debasededatosdistribuidas.doc

[4] Malinowski, E. “Fragmentación vertical de clases en las bases de datos distribuidas orientadas a objetos”. 1999. [Consultado en enero del 2007]
Disponible en World Wide Web:
www.dlsi.ua.es/asignaturas/bdd/articulos%20recomendados/Malinowski1999.pdf

[5] Departamento de O.E.I. - U.P.M. “Diseño y Optimización de Bases de Datos: Bases de Datos Distribuidas”. [Consultado en enero del 2007]
Disponible en World Wide Web:
www.oei.eui.upm.es/Asignaturas/BD/DYOBD/DISTRIBUIDAS.pdf

[6] Hende, G. “Aplicación para resolución de conflictos en bases de datos que no ofrezcan características de distribución de datos”. Fundación Universitaria San Martin. Bogotá DC, 2005. 30.

[7] Case, K. “Replication Strategies: Data Migration, Distribution and Synchronization”.
Sysbase, white paper, 2003. 30.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.sybase.com/content/1028711/6143_whitepaper_v2.pdf

[8] González, G, Yorio, P, Alonzo, L, Barreto, C. “Técnicas Avanzadas para Gestión de Sistemas de Información”. Instituto de Computación. Facultad de Ingeniería. Universidad de la República Oriental del Uruguay. Uruguay, 2003. 21.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.fing.edu.uy/inco/cursos/tagsi/Trabajos/2003/TAGSI03-TareaTecnSI-grupo4.pdf

[9] Enciclopedia Wikipedia. “Software libre”. [Consultado en enero del 2007]
Disponible en World Wide Web:
http://es.wikipedia.org/wiki/Software_libre

[10] Stallman, R. Porqué "Software Libre" es mejor que software de "Código Fuente Abierto". [Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.gnu.org/philosophy/free-software-for-freedom.es.html

[11] PostgreSQL 8.2.4 Documentation. “What is PostgreSQL?” [Consultado en enero del 2007].
Disponible en World Wide Web:
http://www.postgresql.org/docs/8.2/interactive/intro-whatis.html

[12] Enciclopedia Wikipedia. “PL/PgSQL”. [Consultado en enero del 2007].
Disponible en World Wide Web:
http://es.wikipedia.org/wiki/PL/PgSQL

[13] Pachon, A. “PL/pgSQL - SQL Procedural Language”, 2000.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.sobl.org/traducciones/postgresql-develdoc/node3.html
[14] Sitio oficial de Python. “About Python”
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.python.org/about/

[15] Sitio oficial de GTK. “GTK+, the GIMP Toolkit”
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://www.gtk.org/

[16] Sitio oficial de Glade. “Glade - a User Interface Designer for GTK+ and GNOME”
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://glade.gnome.org/

[17] Jacobson, I; Booch,G; Rumbauhgh,J. “El proceso Unificado de desarrollo de software”. Editorial Addison Wesley, 2000.

[18] Soto,N; Saborit, Y. *Hubble: Propuesta para un sistema de catalogación y recuperación de recursos de información*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, junio 2004.

[19] Mustain, E “Introducing Slony”, 2004. [Consultado en enero del 2007]
Disponible en World Wide Web:
http://www.onlamp.com/pub/a/onlamp/2004/11/18/slony.html

[20] Sitio oficial PgCluster. “PGCluster is the synchronous replication system of the multi-master composition for PostgreSQL”.
[Consultado en enero del 2007]. Disponible en World Wide Web:
http://pgcluster.projects.postgresql.org/feature.html

[21] e-Quallity Corp. “Fundamentos”. [Consultado en enero del 2007].
Disponible en World Wide Web:
http://www.e-quallity.net/definiciones.php

Glosario de Términos

Aplicación: “A menudo se refiere al programa que se está ejecutando y a los archivos y bases de datos con los que se trabaja.”

Nomencladores: En términos de bases de datos, conjuntos de valores que se definen para un campo determinado. Se usan con el objetivo de evitar errores de los usuarios, pues no se incurre en la entrada de valores erróneos, además de ahorrar espacio en la base de datos, pues solo se almacena el identificador del valor.

Transacciones: son grupos de operaciones que deben dar la apariencia de ser ejecutadas secuencialmente como una unidad. La definición de 'correcta' en una transacción se refiere al cumplimiento de las propiedades ACID.

Atomicidad:”Las transacciones son atómicas cuando al ejecutarse una operación esta se ejecuta de comienzo a fin, de llegarse a presentar algún problema deshace toda la operación (todo o nada)”.

Consistencia: “…Una transacción transforma un estado consistente de la base de datos en otro igual sin necesidad de conservar la consistencia en todos los puntos intermedios”.

Aislamiento: Cuando dos transacciones se ejecutan concurrentemente, sus efectos deben ser aislados. Esto es, no debe haber efectos que no ocurrirían si ambas transacciones se hubiesen ejecutado secuencialmente.

Durabilidad: Una vez que la transacción ha terminado, su efecto no puede perderse en caso de fallas del sistema; ni siquiera si la falla ocurre inmediatamente después de terminada la transacción.

Nodo: En términos de la propuesta para replicar base de datos fragmentadas, son dos bases de datos PostgreSQL que intervienen en la réplica.

Cluster: En términos de Slony-I, un cluster son dos bases de datos PostgreSQL que intervienen en la réplica.

Backup: **copia de seguridad** o **copia de respaldo,** se refiere a la copia de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una perdida de información.

Metadato: En términos de base de datos, es un fichero texto en el cual se plasma toda la información para construir una base de datos, estructura y datos.

Índices: Uno de los mecanismos más importantes en las bases de datos relacionales son los índices, los cuales aceleran infinitamente las consultas de mayor relevancia sobre los datos almacenados. En PostgreSQL el usuario puede definir los índices que necesite.

ReIndex: Reconstruye los índices usando los datos almacenados en la tabla de los índices, remplazando la copia vieja del índice.

Vacuum: Recupera espacio de almacenamiento ocupado por tuplas que han sido borradas.