ARTÍCULO ORIGINAL

Bases de datos distribuidas para aplicaciones médicas en el Sistema Nacional de Salud

Distributed database for medical applications in the National Health System

MsC. Lic. María del Carmen Paderni López,[I] Ing. Idorys Aguilar León,[II] MsC. Ing. Mirna Cabrera Hernández,[III] MsC. Dr. Ariel Delgado Ramos[IV]

[I]SOFTEL. La Habana, Cuba. E-mail: carmenchu@softel.cu
[II]SOFTEL. La Habana, Cuba. E-mail: idorys@softel.cu
[III]SOFTEL. La Habana, Cuba. E-mail: mirna@softel.cu
[IV]MINSAP. La Habana, Cuba. E-mail: ariel.delgado@infomed.sld.cu

RESUMEN

Cuba enfrenta el reto de informatizar su sociedad optimizando el uso de nuevas tecnologías, para mejor gestión de la información médica. Lleva a cabo la informatización del Sistema Nacional de Salud, desarrollando el Sistema de Información para la Salud (SISalud), plataforma formada por componentes con un nivel de cohesión y acoplamiento que permite integrar las aplicaciones informáticas para la salud, interactuar entre ellas y reutilizar la información de manera eficiente; almacenada en bases de datos independientes, soportadas sobre un servidor MySQL y comunicándose entre sí mediante Servicios Web, basados en XML y arquitectura en 3 capas. Esto le ofrece al usuario final una visión integrada de los datos para ser usados por distintos niveles de dirección, docencia, investigación y gestión de salud, lo que garantiza la gestión, en tiempo real y con alcance nacional, de los datos generales de los ciudadanos cubanos y el capital humano en Unidades de Salud. El objetivo de este trabajo es demostrar las ventajas del uso de las bases de datos distribuidas como parte del desarrollo de esta solución informática para el sector de la salud.

Palabras clave: proceso, informatización, servicios web, integración.

ABSTRACT

Cuba faces the challenge of computerizing their society optimizing the use of new technologies to better management of medical information. Performs computerization of NHS, developing the Information System for Health (SISalud) platform consists of components with a level of cohesion and coupling that allows applications to integrate health, interact with them and reuse information efficiently; stored in separate databases, supported on a MySQL server and communicating with each other via Web Services, and XML-based 3-tier architecture. This gives the end user an integrated view of the data to be used by different levels of management, teaching, research and health management, ensuring management, and real-time national reach of the general data of citizens Cubans and human capital in health units. The aim of this work is to demonstrate the advantages of using the distributed database as part of the development of this software solution for the health sector.

Key words: process, informatization, web services, integration.

INTRODUCCIÓN

En el 2003 el Ministerio de Salud Pública como organismo rector del Sistema Nacional de Salud (SNS) define como una prioridad su informatización, con el propósito de elevar la eficiencia, seguridad, calidad y estética en todas las direcciones. Para ello se convoca a un grupo de instituciones propias del sector, del Ministerio de Informática y Comunicaciones y de otros organismos de la administración central del estado, para definir de conjunto la estrategia a llevar a cabo. En algunos casos se ha tomado como punto de partida sistemas ya elaborados en el país en el marco de aquella primera estrategia de desarrollo.[1]

El MINSAP ha definido un grupo de premisas y requisitos que incorporan los últimos adelantos en el área de las tecnologías de la información y las comunicaciones y que garantizan la plataforma de integración de las aplicaciones, la compatibilidad y sostenibilidad de los productos a elaborar, tales como: empleo de tecnologías basadas en Internet (XML, Web Services), software libre (PHP, MySql, Linux), documentación de todo el proceso productivo, requisitos de seguridad del software, independencia de la base de datos, desarrollo en multiplataforma y empleo de estándares internacionales para los productos relacionados con la salud. El soporte de infraestructura en todos los aspectos mencionados es la Red Telemática de la Salud.

Con esta estrategia se lograría que la información en las bases de datos sea única, confiable y en tiempo real, que se garantice la integridad de la información e interconectar entre sí las diferentes aplicaciones existentes.

El objetivo de este trabajo es mostrar cómo el Sistema Información para la Salud (SISalud), cuenta con una base de datos que da solución centralizada y disponible para el dominio de salud, lo que garantiza brindar información en forma estándar a todos los proyectos que la necesiten y evitar la duplicidad de la misma, ya sea por exportación de datos, o a través de otras tecnologías como Servicios Web.

METODOLOGÍA

Para llevar a cabo la solución propuesta por el MINSAP para la informatización de la Salud Pública, se crea una plataforma de aplicaciones, abierta, con una interfaz de programación que permite incorporar nuevos módulos compatibles entre sí. Es portable a diferentes sistemas operativos, tanto en los servidores como en los clientes, replicable en otros entornos. Es una plataforma en constante desarrollo, que crece en la medida en que se implementan nuevos módulos. En la primera etapa se desarrolló el Registro Informatizado de Salud (RIS), implementándose entre los módulos iniciales aquellos que serían los nomencladores nacionales de recursos, geográficos y servicios.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel seleccionando la arquitectura cliente servidor y en 3 capas.

Arquitectura cliente servidor. Arquitectura hardware y software adecuada para el proceso distribuido, en el que la comunicación se establece de uno a varios. Un proceso es un programa en ejecución. *Proceso cliente* es el que solicita un servicio. *Proceso servidor* es el capaz de proporcionar un servicio. Un proceso cliente se puede comunicar con varios procesos servidores y un servidor se puede comunicar con varios clientes.

El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.[2] El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

Arquitectura en tres capas. Es aquella donde la solución es segmentada desde el punto de vista lógico en Presentación, *Lógica de Negocio y Datos*. La Capa de Presentación es la que presenta el sistema al usuario, le comunica la información y captura la información del usuario, se comunica únicamente con la capa de negocio. La Capa de Negocio es donde residen los programas que se ejecutan y se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con *la capa de datos, para solicitar al gestor de base de datos*, almacenar o recuperar datos de él. La Capa de Datos es donde residen los mismos, está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.[3] (Fig. 1).

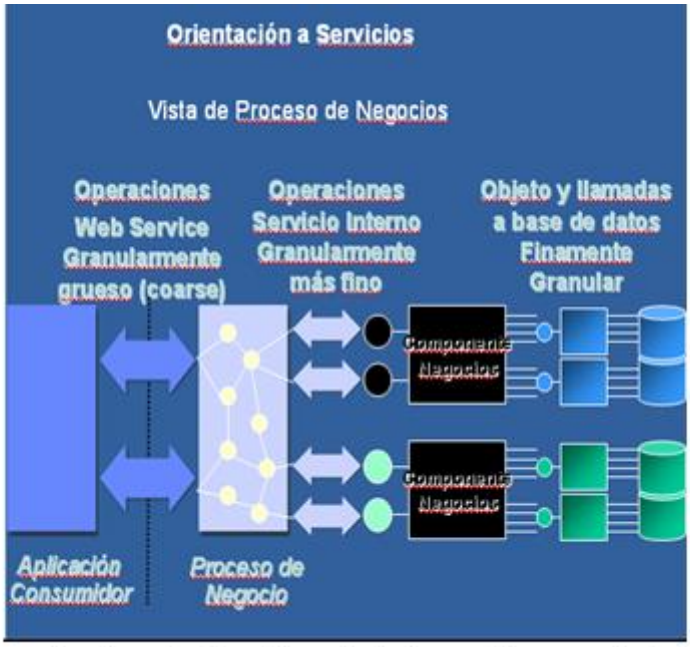


Fig. 1. Arquitectura del Registro de Información para la Salud (RIS)

Las bases de datos son utilizadas para el almacenamiento estructurado de datos y gestionar la labor, tanto de usuarios, como de los programadores que las desarrollaron.

El servidor de bases de datos elegido inicialmente fue MySQL 4.0.3., hoy ya con sus versiones más actuales. Es el sistema de gestión de bases de datos SQL Open Source (Open Source significa que es posible para cualquiera usar y modificar el software bajándolo desde Internet sin pagar nada.), más popular, lo desarrolla, distribuye y soporta MySQL AB.

Una base de datos es una colección estructurada de datos. Desde las grandes aplicaciones multiusuario, los hospitales inteligentes, pasando por las historias clínicas electrónicas, hasta las tiendas virtuales utilizan tecnología de bases de datos para gestionar los mismos y garantizar su integridad. MySQL es un sistema de gestión de bases de datos relacionales. Una base de datos relacional almace na datos en tablas separadas. Esto añade velocidad y flexibilidad.

RESULTADOS Y DISCUSIÓN

El servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años. Ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad lo convierten en un servidor bastante apropiado para acceder a bases de datos en Internet. Con la versión 5.1 se pueden crear procedimientos almacenados. MySQL soporta distintos tipos de tablas, tales como ISAM, MyISAM e InnoDB.

Al inicio del desarrollo de la aplicación (año 2003) fue elegida como *storage system* el MYSAM, tipo predeterminado, realizando el control de la Integridad Referencial a nivel de la capa de negocios. Esto fue hecho para no hacer la aplicación dependiente de este motor, relativamente nuevo, y con la idea de poder realizar en un futuro la migración para otro motor que pudiera existir.

Posteriormente en el año 2006, después de distintos estudios fue seleccionado InnoDB, la cual es el tipo de tabla más importante, siendo definidas con ésta todas las nuevas bases de datos de los diferentes componentes.

Las tablas del tipo InnoDB están estructuradas de forma distinta que MyISAM y soportan verificación para restricciones de claves foráneas. Hay que tener en cuenta que la sintaxis para FOREIGN KEY en las columnas en la tabla referenciada debe ser nombrada explícitamente. Soporta las acciones ON DELETE y ON UPDATE. No es algo nuevo, desde la versión 3.23 se podía hacer uso de tablas InnoDB.

Se comenzaron a realizar estudios para optimizar las consultas a las bases de datos:

- Uso de índices.
- Uso de tablas caches.
- Solución Balanceo carga con LVS sobre réplicas de MySQL.

Uso de índices

Se realizaron estudios de los índices los que son utilizados para encontrar de forma rápida los registros que tengan un determinado valor en alguna de sus columnas. Sin un índice, MySQL tiene que iniciar con el primer registro y leer a través de toda la tabla para encontrar los registros relevantes. Aún en tablas más pequeñas es más rápido leer los datos usando un índice, que haciendo una lectura secuencial. Para lograr la agilización de las consultas utilizando la sentencia *Explain*, veamos el ejemplo siguiente:

```
EXPLAIN
Select a.ID, a.NOMBRE, a. APELLIDO ,
    a.CARNET , b. TIPO
from Bd1.registro a
  inner join Bd2.t_medico b
  on b.ID =a.ID-M Where
  a.CARNET_M = '62050714234'
```

En cada fila del resultado, nos explica cómo ha utilizado los índices de cada tabla involucrada en la consulta. La columna ''*type*'' indica el tipo de ''*join*'' que ha podido hacer. En nuestro caso, ''*ref*'' se ha consultado una fila de esta tabla para cada combinación de filas de las otras. Se están utilizando los índices. (Fig. 2).

Fig. 2. Ejemplo de Consulta 1

No es así en el siguiente caso:

```
EXPLAIN
Select
  a. ID, a. NOMBRE, a. APELLIDO ,
  a.CARNET , b.CUBANO
FROM
  Bd1.registro a
 inner join bd2.t_medico b
  on b.ID = a.ID_M Where
  a. CUBANO = 1
```

Ahora en la tabla b el tipo es ''ALL'', esto indica que el gestor ha tenido que leer toda la tabla para comprobar las condiciones de la consulta. (Fig. 3).

ID	NOMBRE	APELLIDO	CARNET	CUBANO
1	Anibal	Marin	55030600264	1

	select_ty	table	type	po
1	SIMPLE	b	ALL	Pro
1	SIMPLE	a	eq_ref	PRI

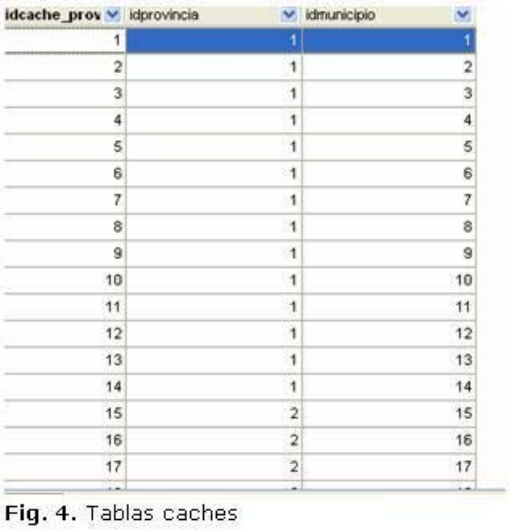
Fig. 3. Ejemplo de Consulta 2

Uso de tablas caches

Las búsquedas entre los distintos componentes, por ejemplo saber todos los médicos que pertenecen a un municipio, se podría resolver con **tablas caches** que agilicen estas búsquedas por ejemplo:

```
SELECT
  Tabla_ cache_p_m.idcache_prov_mun,
 Tabla_ cache_p_m.idprovincia,
  Tabla_ cache_p_m.idmunicipio
FROM
  Tabla_ cache_p_m.
```

Resultado en figura 4:



idcache_prov	idprovincia	idmunicipio
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	1	10
11	1	11
12	1	12
13	1	13
14	1	14
15	2	15
16	2	16
17	2	17

Fig. 4. Tablas caches

Que si bien traería cierta redundancia de la información podría evitar abortar las sesiones.

Solución con Transacciones en MySQL

También como otra mejora, el uso de las transacciones permite una fiabilidad superior, e integración de los datos. Las transacciones aportan para una serie de consultas SQL que deben ejecutarse en conjunto. Con el uso de las transacciones estaremos seguros que no quedaremos incompletos, produciendo como tal falta de integridad de los datos. Por ejemplo, en las aplicaciones de SISalud, si a un médico se le asignan sus datos que son hijos de la tabla principal y el método de negocio al insertar la tabla padre falla, no deben quedar 'colgados' los hijos. Es como adicionar la característica de "deshacer" a las aplicaciones sobre bases de datos.

Las tablas que soportan transacciones, como es el caso de InnoDB, son más seguras y fáciles de recuperar si se produce algún fallo en el servidor, ya que las consultas se ejecutan o no en su totalidad.

Un ejemplo típico lo podemos encontrar en una actualización del estado de un médico; si cambia su estado, debe actualizarse su ubicación física.

```
UPDATE `t_medico`
SET estado = 'P' WHERE id_medico = 30;
UPDATE t_ubicacion
SET baja = '2008-02-24'
WHERE id_ medico_u = 30
```

Estas dos consultas deben trabajar bien, ¿Pero qué sucede si ocurre algún imprevisto y "se cae" el sistema después de que se ejecuta la primera consulta, pero la segunda aún no se ha completado?. El médico quedaría pasivo, pero no se actualizaría su fecha de baja.

Ambas consultas deben ser ejecutadas de manera conjunta, o en su caso, que no se ejecute ninguna de ellas. Es aquí donde las transacciones toman un papel muy importante.

Los pasos para usar transacciones en MySQL son:

- Iniciar una transacción con el uso de la sentencia BEGIN.

- Actualizar, insertar o eliminar registros en la base de datos.

- Si se quieren los cambios a la base de datos, completar la transacción con el uso de la sentencia COMMIT. Solo cuando se procesa esta sentencia los cambios hechos por las consultas serán permanentes.

- Si sucede algún problema, podemos hacer uso de la sentencia ROLLBACK.

De esta forma se resolvía el problema a nivel de cada base de datos, pero qué podría suceder entre las diferentes relaciones que sí existen virtualmente entre las bases de datos, o la mal llamada integridad entre los módulos. ¿Podrían estas aplicaciones web resolver estos problemas?

Por ejemplo: a nivel nacional se manipula el Registro de Unidades de Salud, pero los médicos en el componente de Personal de Salud están asignados a una de esas Unidades de Salud.

Qué sucedería si decidiera eliminar una unidad de salud, esto se envía a un Work Flow, o sea está encaminado a nivel de un Work Flow (en una base de datos independiente), que permite notificar qué debe ser eliminado, pero esta información no llega al otro componente el Registro Personal de Salud (RPS), pues estos médicos deben ser cambiados a otra unidad de salud, antes de que esto ocurra, pues ningún componente debe realizar responsabilidades que no son suyas, ya que se cumplen los siguientes patrones:[4]

Patrón Alta Cohesión. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se simplifica el mantenimiento. Soporta mayor capacidad de reutilización.

Patrón Bajo Acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Con el empleo de este patrón no se afectan por los cambios de otros componentes, estos serían fáciles de entender por separado y de reutilizar.

Actualmente solo está propuesto crear un sistema de aviso para notificar cualquier cambio, ejemplo si un médico es declarado pasivo, todos los módulos que lo tiene asignados debían saberlo, etc.

En el año 2010, se desplegó la versión 2.0 de RPS, que fue implementado por el grupo de mantenimiento del RIS de la Factoría de Software de Softel, la cual genera los siguientes cambios:[5]

- Creación de nuevas tablas y nuevas relaciones entre ellas, lo que implica un rediseño de la base de datos.

- Cambios en el diseño de las páginas y en los códigos.

- Migración de los datos actuales a las nuevas estructuras.

CONCLUSIONES

Para poder lograr la integración de todas las Componentes y con ello, de las Bases de Datos que formen parte de SISalud, en un primer momento fue creado un Grupo de Arquitectura en cada entidad, además del Grupo de Arquitectura MINSAP - MIC, del cual forman parte los líderes de los grupos de arquitectura de cada entidad y algunos miembros del MINSAP. En el consejo de este grupo se analizaba cada proyecto y sus resultados, realizándose el estudio de la posible integración entre los mismos. Hoy esto está organizado en estrecha coordinación entre la Dirección de Informática del MINSAP y la empresa SOFTEL.

Si bien es cierto que la conectividad pudiera estar afectada producto de los desastres naturales (como ciclones, inundaciones, etc.) ocurridos o que pudieran ocurrir, las características que tiene el RIS, permiten usar la aplicación por aquellas entidades que si tengan las condiciones técnicas adecuadas. Para los otros casos

habrá que crear un Plan de Contingencia. También se hace necesario que a nivel nacional del MINSAP existan un grupo de administradores encargados de mantener actualizada la información de cada componente.

REFERENCIAS BIBLIOGRÁFICAS

1. Delgado, A., Vidal, MV. Informática en la salud pública cubana. Revista Cubana Salud Pública. Volumen 32 no. 3, 2006. [Consultado en febrero 2014]]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-46620060003000015&lng=es&nrm=iso&tlng=es

2. Guerra, J. Sistemas Distribuidos. Agosto 2011. [Consultado en enero 2013]. Disponible en: http://es.slideshare.net/jguerra42/caracteristicas-de-los-sistemas-distribuidos

3. Softel. Documento sobre la Arquitectura de Software para los componentes a emplear por el Sistema de Información para la Salud. La Habana, 2006.

4. Paderni, MC. Gestión centralizada del Personal de la Salud para dar servicio a los distintos componentes del Sistema de Información para la Salud. Tesis de Maestría Universidad de Ciencias Informáticas, 2011.

5. Gutiérrez, S., García, D., Infante, E. Registro de Trabajadores de la Salud. VIII Congreso Internacional de Informática en la Salud, La Habana 2011.

Recibido: 15 de octubre de 2014.
Aprobado: 21 de noviembre de 2014.