Ambiente de simulación para la recuperación en un entorno con Bases de Datos Distribuidas

A.C. Ivana Miatón[1]
A.C. Sebastián Ruscuni[1]
Lic. Rodolfo Bertone [2]
Ing. A.De Giusti[3]

Laboratorio de Investigación y Desarrollo en Informática[4]
Facultad de Informática
UNLP

Resumen

La utilización de bases de datos distribuidas (BDD) o de bases de datos federales o federativas (DBF) representan una solución viable para los usuarios cuando deben optar para la generación de sus sistemas de información. La utilización de estas bases de datos para el mantenimiento de la información requiere el estudio de una gran cantidad de casos particulares, a fin de determinar las mejores condiciones de trabajo para cada problema real.

Este artículo es la continuidad del desarrollo de un modelo de simulación, donde estudiamos, modelizamos, implementamos y medimos situaciones en la que una BDD debe mantener la integridad de la información ante fallos producidos durante la ejecución de transacciones, utilizando el protocolo de commit de dos fases. Se optó, para implementar el entorno de simulación, Java, en lugar de un entorno PVM, debido a diferentes aspectos como facilidad de trabajo, portabilidad, etc.. El ambiente implementado se basa principalmente en la recuperación de fallas en un entorno con replicación estática de la información, pudiendo modificar para cada prueba realizada el porcentaje de replicación que presentan los datos la BDD.

[1] Analistas en Computación. Alumnos avanzados de la Licenciatura en Informática. Facultad de Informática, UNLP. Becarios L.I.D.I.
E-mail imiaton,sruscuni@lidi.info.unlp.edu.ar
[2] Prof. Adjunto con Dedicación Exclusiva, LIDI. Facultad de Informática, UNLP.
E-mail pbertone@lidi.info.unlp.edu.ar
[3] Inv. Principal CONICET. Profesor Titular Dedicación Exclusiva, Facultad de Informática, UNLP.
E-mail degiusti@lidi.info.unlp.edu.ar
[4] Calle 50 y 115 Primer Piso, (1900) La Plata, Argentina, Teléfono +54 221 422 7707
WEB: lidi.info.unlp.edu.ar

Introducción

Una Base de Datos Distribuidas (BDD) puede ser definida como una colección integrada de datos compartidos que están físicamente repartidos a lo largo de los nodos de una red de computadoras. Un DDBMS es el software necesario para manejar una BDD de manera que sea transparente para el usuario. [BURL 94]

Un DBMS centralizado es un sistema que maneja una BD simple, mientras que un DDBMS es un DBMS simple que maneja múltiples BD. El término global y local se utiliza, cuando se discute sobre DDBMS, para distinguir entre aspectos que se refieren al sitio simple (local) y aquello que se refiere al sistema como un todo (global). La BD local se refiere a la BD almacenada en un sitio de la red, mientras que la DB global se refiere a la integración lógica de todas las BD locales. [BELL 92]

El propósito de la BD es integrar y manejar los datos relevantes de una empresa o corporación. La motivación para definir esta BD es la de tener todos los datos relevantes de las operaciones de la empresa en un único almacenamiento, de manera que todos los problemas asociados con aplicaciones en una empresa puedan ser servidos de una manera uniforme. La dispersión geográfica de estas empresas hace a un modelo centralizado no aplicable, siendo un modelo distribuido la solución apropiada. [SCHU 94] [BHAS 92]

El modelo distribuido de datos hace posible la integración de BD heterogéneas proveyendo una independencia global del DBMS respecto de esquema conceptual. Además, es posible implementar una integración tal, que reúna varios modelos de datos, representado cada uno de ellos características propias de empresas diferentes, asociadas para un trabajo conjunto. Este modelo de distribución, genera las denominadas Bases de Datos Federativas. [LARS 95] [SHET 90]

Nuevamente está bajo discusión si las BDD son la solución para los problemas que presenta actualmente el mercado. Los conceptos de downsizing y rigthsizing están siendo nuevamente revisados. La globalización existente lleva a que grandes empresas fusiones sus actividades y esto conlleve la necesidad de integrar varios sistemas funcionando, y generando lo que denominamos BDF, las cuales tienen asociados los principales inconvenientes que aparecen en los entornos distribuidos.

Con los modelos distribuidos o federativos de datos, surgen una serie de consideraciones especiales y problemas potenciales que no estaban presente en los sistemas centralizados. La replicación y fragmentación de información hace a que los datos estén más “cerca” y “disponibles” para los usuarios. Estos dos conceptos necesitan nuevos mecanismos para la preservación de la integridad y consistencia de la información que contemplen la ubicación de los datos y las copias existentes de los mismos. [DATE 94] [THOM 90]

La utilización de BDD tiene asociado una serie de beneficios y costos que deben evaluarse en el momento de tomar decisiones respecto de su utilización.[ÖZSU 91]. Algunas de las ventajas mencionables son:

- Autonomía local: cada sitio de la red en un entorno distribuido debe ser independiente del resto. Cada BD local es procesada por su propio DBMS.
- Mejoras en la performance.

- Mejoras en la disponibilidad de la información.
- Economía: el equipamiento necesario para montar una red para distribuir la información, es comparativamente mucho menor que tener un sitio central con gran capacidad de procesamiento. Está asociado con el concepto de downsizing. [UMAR 93]
- Sistemas fácilmente expansibles.
- Compartir información.

Entre los inconvenientes y desventajas que surgen están:

- Poca experiencia en el desarrollo de sistemas y BD distribuidas
- Complejidad (sin inherentemente más complejos)
- Costo: se requiere hardware adicional, software más complejo, y el costo del canal de transmisión, para conectar los nodos distribuidos de la red.
- Distribución de control: se contradice con una de las ventajas planteadas. Se hace hincapié, en este punto, en los problemas creados por la sincronización y coordinación de tareas.
- Seguridad: aparecen nuevos problemas, y más complejos relacionados con la seguridad de la información manipulada.

Reglas para los DDBMS

En 1987 Date, propuso un conjunto de reglas que un DBMS debe cumplir [BERS 92]:

1. Autonomía local
2. No es necesario que exista un sitio central.
3. Operación continua
4. Independencia
   - Respecto de la fragmentación de los datos
   - Respecto de la replicación
   - Respecto de la ubicación de los datos
   - Respecto del hardware utilizado, sistema operativo, redes de datos, del DBMS.
5. Procesamiento de consultas distribuidas
6. Manejo de transacciones distribuidas

Seguridad e Integridad de la información

Una transacción es una colección de operaciones que realiza una única función lógica. Cada transacción es una unidad de atomicidad (debe ocurrir completa o no ocurrir). Los algoritmos para asegurar la consistencia de la BD, utilizando las transacciones, incluyen: [SILB 98]

- Acciones tomadas durante el procesamiento normal de las transacciones que aseguran la existencia de información suficiente para asegurar la recuperación de fallos.
- Acciones tomadas a continuación de un fallo, para asegurar la consistencia de la BD.

Las transacciones cumplen con dos propiedades:

- Conservar la consistencia de la BD,
- Ejecutarse como un todo o no llevarse a cabo (atomicidad).

Existen dos mecanismos utilizados para asegurar la atomicidad de las transacciones:

- Basado en bitácora
- Doble paginación

Protocolos de concurrencia

Se continúan aplicando los conceptos definidos para un entorno de BD centralizadas a fin de asegurar el acceso concurrente a la información. Sin embargo, deben tenerse en cuenta una serie de nuevas características que aparecen en los entornos distribuidos. Esto lleva a la definición de los siguientes protocolos:

- Protocolo de un único coordinador de bloqueos: el gestor de bloqueos del sistema es único y está ubicado en una localidad L de la red. Cada transacción que necesite un dato debe realizar el pedido al único coordinador. Características: fácil de implementar, fácil manipulación de bloqueos, todas los pedidos de bloqueo y liberación de datos se centran en una localidad con la consiguiente pérdida de performance, el sistema se vuelve vulnerable, se contradice uno de los principios de Date.
- Protocolo de mayoría: cada localidad posee su propio gestor de bloqueos que administra el acceso a sus datos locales. Cuando una transacción necesita bloquear un dato que se encuentra replicado, se envía un pedido de bloqueo a cada localidad, si la mitad contesta afirmativamente se obtiene el acceso al dato. Características: evitar el cuello de botella, más difícil de implementar, aumenta el tráfico en la red tanto para solicitar el bloqueo como para liberar el dato, posibilidad de deadlock.
- Protocolo preferencial: similar al anterior pero diferenciando bloqueos compartidos de exclusivos. La gestión de los bloqueos continúa descentralizada, esto es, cada localidad administra su propia información. Una transacción que necesita un bloqueo compartido del dato solo debe obtener el permiso desde una réplica. Para un bloqueo exclusivo es necesario obtener el acceso al dato en todas las réplicas. Características: menor tráfico en la red ante operaciones de consulta de información, mayor complejidad del algoritmo para el tratamiento de bloqueos compartidos. Por las características anteriores, sistemas con más operaciones de consulta que ABM se ven beneficiados con la utilización del protocolo.

Integridad en Bases de Datos Distribuidas. Protocolo de dos fases

A fin de mantener la integridad de los datos, en un entorno distribuido, surgen algunos mecanismos que permiten administrar las transacciones que acceden globalmente a la información. Se describe a continuación el entorno de trabajo y las principales características del protocolo de dos fases, que posteriormente se implementa en la simulación realizada.

Cada transacción se genera en una localidad, el coordinador local de transacciones es el encargado de controlar la ejecución de la misma. En caso de ser una transacción local, se procede igual a los sistemas centralizados. Ahora, si la transacción accede a datos globales, el coordinador debe encargarse de subdividir la transacción en subtransacciones que se ejecutarán en diferentes localidades de la red. Cada localidad

participante recibe la subtransacción que puede tratar y la procesa como si ésta fuera una local. [WEB1] [WEB2]

El coordinador debe, una vez que el procesamiento finaliza, decidir en que estado termina la transacción correspondiente. [WEB3] Para ello activa el protocolo de dos fases:

- *Fase 1*: el coordinador envía un mensaje a cada localidad involucrada preguntando si se pudo finalizar la transacción. Ante una respuesta negativa, o eventualmente, falta de respuesta, la transacción debe ser abortada. Si todos contestan afirmativamente la transacción está en condiciones de ser cometida.
- *Fase 2*: se envía un mensaje de abortar (si hay al menos una respuesta negativa o se cumple un tiempo de espera predeterminado), o un mensaje de finalizar (si todas incluyendo al coordinador pudieron finalizar la transacción. Cada localidad comete la transacción y envía un mensaje de reconocimiento al coordinador.

El protocolo que se utiliza incorpora un *timeout*, es decir, se tiene en cuenta el tiempo de espera máximo entre nodos. Si una subtransacción no recibe un mensaje del coordinador le pide a todas las participantes un mensaje de ayuda. Cuando alguna recibe este mensaje puede ayudar a su compañera, de acuerdo a su propio estado; es decir, si ha recibido del coordinador un mensaje commit o abort, envía el mismo. La transacción en espera puede ahora realizar el commit o abortar sin ningún lugar a dudas, pues su compañera ha replicado el mensaje seguro del coordinador. Por el contrario, si la subtransacción no recibe ningún mensaje, continuará en espera. [ZANC96] [BERN84]

Arquitectura utilizada

Las computadoras sobre Internet están conectadas a través del protocolo TCP/IP. En la década del 80, ARPA (Advanced Research Projects Agency) del gobierno de EEUU, desarrolló una implementación bajo UNIX del protocolo TCP/IP. Allí fue creada una interface socket. En la actualidad, la interface socket constituye el método más usado para el acceso a una red TCP/IP. [WEB4]

Un socket es una abstracción que representa un enlace punto a punto entre dos programas ejecutándose sobre una red TCP/IP. Utiliza el modelo Cliente/Servidor. Cuando dos computadoras desean conversar, cada una usa un socket. Una de ellas es el "Server" que abre el socket y espera por alguna conexión. La otra es el "Cliente" que llamará al socket server para iniciarla. Además, para establecer la conexión, solo será necesaria la dirección del Server y el número de puerto que se utilizará para la comunicación. Aquí se utiliza principalmente el package java.net. Cuando los dos sockets están comunicados, el intercambio de datos se realiza mediante InputStreams y OutputStreams. [WEB5]

La ventaja de este modelo sobre otros tipos de comunicación se ve reflejada en que el Server no necesita tener ningún conocimiento sobre el sitio en donde reside el cliente. Por otra parte, plataformas como UNIX, DOS, Macintosh o Windows no ofrecen ninguna restricción para la realización de la comunicación. Cualquier tipo de computadora que soporte el protocolo TCP/IP podrá comunicarse con otra que también lo soporte a través del modelo de sockets. [WEB6]

Presentación del trabajo

Se realizaron simulaciones sobre la ejecución de transacciones en una base de datos distribuida, con replicación total o parcial de información, utilizando el protocolo de 2 fases para la recuperación ante distintos casos de falla que se presentan a continuación, manteniendo en todo momento la consistencia de la base de datos. Dicho ambiente se implementó en Java.

Las distintas *situaciones de fallo* que pueden ocurrir durante la ejecución de una transacción distribuida son:

- *Fallo de una localidad participante.* Para recuperarse, debe examinar su bitácora y a partir de allí, decidir el destino de la transacción.
- *Fallo del coordinador*. En este caso, las localidades participantes son quienes deben decidir el destino de la transacción. Si no poseen suficiente información, tendrán que esperar a que se recupere el coordinador.

Para la implementación de una solución al problema, hay que tener en cuenta que se optó en la simulación por utilizar el mecanismo de preservación de integridad de datos basado en bitácora disponiendo, por lo tanto, de archivo *Log* que contiene la información que indica el tipo de operación realizada y los datos involucrados (comienzo, cometido, abortado, reconocimiento). Además, ante una instrucción de modificación de datos se opera con la técnica de modificación inmediata de los datos, debiendo almacenar el valor viejo y el valor nuevo.

Se analizaron diferentes porcentajes de replicación de la información con el objetivo de simular el comportamiento de la información ante diferentes configuraciones de la BDD.

La implementación del ambiente se realizó en Java. Se utilizó el JDK 1.2, conjuntamente con el VisualAge 2.0 para construir la interface con el usuario (GUI). De esta forma se logra contar con interoperabilidad relacionada al hardware, permitiendo que cualquier computadora que posea JVM (Java Virtual Machine) pueda ejecutar el ambiente sin ningún inconveniente.

La comunicación entre procesos se realiza en forma asincrónica, por lo tanto será bloqueante para el que recibe pero no para el que envía.

Se dispone de cuatro clases diferentes de tareas:

- Manager
- Server
- Coordinator
- Locale

Los procesos que simulan al coordinador y a las localidades participantes de la transacción serán “Threads” que son creados por los procesos “Servers” que residen en cada localidad.

Inicialmente, se ejecuta el proceso Manager, el cual provee la interface para que el usuario pueda realizar cualquier tipo de operación sobre la base de datos. Esto significa la realización de un Query, el cual puede ser tanto una consulta como una

operación de inserción, borrado o actualización. Para realizar estas operaciones, el usuario no debe ingresar directamente código SQL, sino que, a través de una interface, se define la información que será utilizada por el ambiente para generar la operación que posteriormente será ejecutada en la base de datos. Esta definición incluye principalmente las tablas sobre las que se va a operar, el tipo de operación a realizar, sitio en donde se desea iniciar la ejecución de la transacción, etc. Esta última elección permite al usuario de la simulación, que una misma operación sobre la BDD se inicie en diferentes lugares y entonces, luego, poder obtener y comparar los resultados obtenidos teniendo en cuenta la distribución inicial de los datos en cada localidad. Cabe destacar que el sitio elegido en donde se iniciará la transacción será el coordinador de la misma.

Cada localidad habilitada para la intervención en la ejecución de una transacción dentro del ambiente de simulación posee un archivo de configuración (aún estático) que permite conocer la ubicación y el uso de cada tabla de la BDD. A partir de esta información, el coordinador determinará cuáles serán las localidades que participarán en la ejecución de las subtransacciones que se generen a partir de la transacción a simular.

Como los casos de estudios elegidos para esta etapa del entorno de simulación contemplan un estudio del comportamiento ante fallos, se deberá indicar cual será la localidad (o el coordinador) que sufrirá el fallo durante la ejecución, produciéndose así la simulación del protocolo de recuperación de la base de datos. En el caso de estudio se contempla la simulación de una caída en la línea de comunicaciones produciendo solamente el fallo en la localidad conectada a través de esa línea.

La simulación necesita definir para cada fallo el momento en que el mismo se producirá.

Casos de fallo simulados

Los casos de fallos contemplados en la actual simulación son: fallos de una localidad, fallo del coordinador, fallo de una línea de comunicación (equivalente a el fallo de una localidad).

Fallo de una Localidad

Manejamos los fallos que pueden ocurrir en cuatro lugares claves durante el procesamiento. En todos los casos, se examina el archivo Log para decidir que acción llevar a cabo:

- *Antes de recibir el "Preparar*". En este caso se asume que el coordinador, como no va a recibir respuesta de ella, decidió abortar, por lo tanto, esta localidad decide abortar.
- *Luego de recibir el "Preparar" y antes de enviar el "Lista"*. Aquí sucede algo muy similar al caso anterior, por lo que también se decide abortar.
- *Luego de haber enviado el "Lista" y antes de recibir alguna respuesta del coordinador*. En este caso la localidad tendrá que comunicarse con el coordinador (o eventualmente con alguna de las localidades hermanas por si el coordinador también falle) para averiguar cual fue el destino de la transacción.

- *Luego de hacer "Commit" o "Abort"*. Aquí se sabe el destino de la transacción y lo que se hace solamente es enviar el "reconocimiento" al coordinador.

Fallo del coordinador

Si el coordinador falla en la mitad de la ejecución de la transacción entonces las localidades participantes deben decidir el destino de la misma. Existirán casos en donde las localidades no podrán determinar si ejecutar y abortar la transacción, y, por lo tanto, será necesario que esperen a que se recupere el coordinador que falló.

En todos los casos, luego que la localidad se recupera, le envía el *"reconocimiento"* al coordinador, estableciendo así el fin de la transacción. En el momento en que el coordinador recibe el reconocimiento de todas las localidades participantes le comunica al *Manager* el resultado de la ejecución de la transacción culminando así la simulación.

Comparación con experiencias anteriores

Los casos de estudio realizados anteriormente, utilizaron para la simulación del protocolo de recuperación una red Linux con soporte de Parallel Virtual Machine (PVM). La migración del soporte a JAVA redundaron en ventajas y mejoras. Estas mejoras están principalmente relacionadas con el ambiente de simulación, el cual presenta una interface mucho más amigable. Además, Java permite manipular conexiones vía JDBC (Java DataBase Connectivity) con la base de datos; con PVM la solución obtenida era no solo mucho más difícil de llevar a cabo, sino que además resultaba muy poco transparente en los pasos de su ejecución.

Entorno de trabajo

Las primeras experiencias realizadas con el entorno de simulación definido utilizaron una red local, es nuestra intensión llevar esta simulación hacia una red distribuida geográficamente, debiendo contar solamente con estaciones de trabajo para lograr este objetivo sin necesidad de hacer cambios en el entorno definido. Esto se debe a que crear los sockets basta con indicar la dirección física de cada localidad y posteriormente ejecutar en cada una de ellas procesos Server.

Simulaciones efectuadas. Estudio de casos.

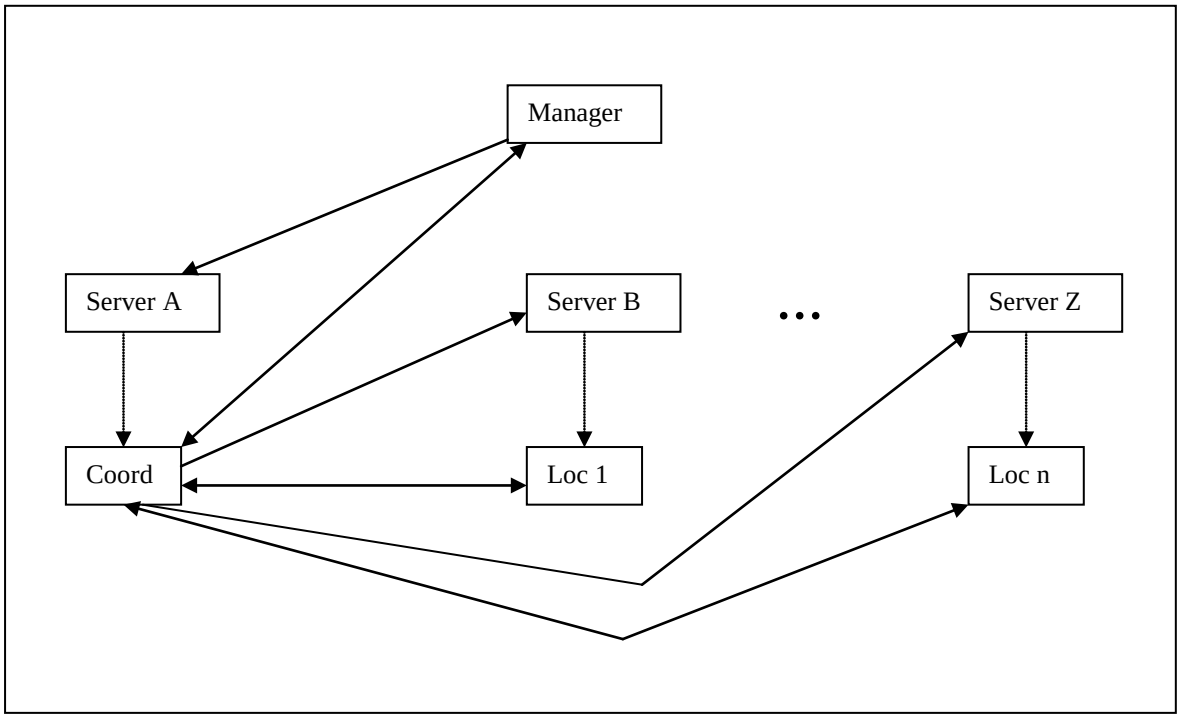
Las transacciones ejecutadas pueden ser definidas como locales o globales. En este último caso se define para dicha transacción el número de subtransacciones que la componen, y que involucran, a priori, diferentes localidades donde se ejecutarán. Es necesario que cada localidad simulada esté ejecutando un proceso denominado *Server*. Además, desde el sitio donde se comienza la ejecución de la transacción se debe ejecutar un proceso *Manager*.

El proceso *Manager*, a partir de las especificaciones de las transacciones (las cuales pueden estar en un archivo de trazas o definirse on line), determina qué proceso *Server* residente en alguna localidad del entorno de simulación, será el Coordinador de la transacción. Luego, este thread determina, consultando la tabla de ubicaciones,

quiénes serán las localidades intervinientes y se lo comunica a los *Servers* correspondientes.

Cuando un *Server* recibe un mensaje de un coordinador indicándole la participación como localidad en la ejecución de la transacción, crea un thread *Locale*.

El esquema de comunicación resultante es el siguiente:



La flechas continuas presentan los canales de comunicación entre procesos, y las punteadas indican la relación *Crea a,* con la que se instancia un nuevo thread.

A partir del establecimiento de los canales de comunicación entre la localidad generadora de la transacción (la cual actúa como coordinadora) y cada uno de las localidades intervinientes se comienza la ejecución de la transacción distribuida. Cada loc_n ejecuta su parte de la transacción y al finalizar indica al coordinador dicho estado. Este coordinador implementa, como se dijo anteriormente, el protocolo de dos fases para mantener íntegra la BD.

Una transacción puede o no fallar. Si no se produce un fallo la integridad de los datos se mantiene, en caso contrario se simulan cualquiera de los fallos anteriormente indicados. Se pudo observar que el ambiente generado mantiene la información de la BD consistente para todas las pruebas realizadas. Estas pruebas incluyeron situaciones donde la información no se encontraba replicada o se encontraba parcialmente replicada (dependiendo de la información contenida en el archivo de configuración).

A partir del ambiente generado es posible medir el comportamiento de las transacciones con o sin fallos en diferentes circunstancias:

- Sin replica de información

- Información replicada parcialmente en algunas localidades
- Información replicada parcialmente en todas las localidades
- Información replicada totalmente en algunas localidades
- Información replicada totalmente en todas las localidades

Conclusiones. Líneas de trabajo.

El objetivo perseguido es definir, modelar e implementar un ambiente de simulación para el mantenimiento y recuperación de datos, en un sistema de BDD, donde se pueda estudiar, monitorear y posteriormente comparar los resultados obtenidos, de la ejecución de transacciones distribuidas a partir de las características de replicación y fragmentación de datos en entornos distribuidos. El objetivo final es que el entorno permita emular problemas donde se utilicen BDD contemplando la mayoría de la situaciones que se puedan plantear, permitiendo de esta forma definir cual es la mejor distribución de datos, el porcentaje óptimo de replicación, fragmentación, etc. [MIAT 98].

Este trabajo es un avance más en el objetivo de desarrollar un ambiente de simulación completo de BDD. El ambiente generado utiliza una herramienta de base más flexible y amigable para realizar la simulación, en el que se pueden configurar los principales aspectos de simulación, poniendo énfasis en la replicación de la información, permitiendo así poder realizar diferentes análisis, modificando el porcentaje de la misma. Una misma transacción puede ser ejecutada desde distintos sitios y así conseguir evaluar la incidencia que existe, sobre la performance de la transacción, de la distribución de la información y de cómo se realizó la replicación de los datos.

Las trazas de ejecución probadas hasta el momento están dando resultados previsibles de acuerdo al contexto definido y se siguen definiendo nuevos casos de prueba para afianzar el modelo desarrollado.

Trabajos futuros

Las líneas de trabajo futuras son varias, con el objetivo final de llegar a completar el ambiente propuesto. Entre los problemas cercarnos a resolver están:

- La generación automática de trazas de prueba. Este punto está parcialmente desarrollado en otros trabajos ya presentados y experimentados. [BERT 99] [DIPA 99]
- Contemplar el manejo de transacciones concurrentes. Esto conlleva al uso locks y protocolos de seguridad e integridad de datos que no se habían tenido en cuenta hasta este momento.
- Contemplar la replicación dinámica de información, midiendo los resultados de la ejecución de transacciones en cada uno de los casos planteados.

Bibliografía

[BELL 92] *Distributed Database Systems*, Bell, David; Grimson, Jane. Addison Wesley. 1992

[BERN 84] *An algorithm for concurrency Control and Recovery in Replicated Distributed Databases* Bernstein, Goodman. ACM Trans. Database Systems, vol 9 no. 4, pp. 596-615, Dec 1984.

[BERS 92] *Client Server Architecture.* Berson, Alex. Mc Graw Hill Series. 1992

[BERT 99] *Ambiente de experimentación para Bases de Datos Distribuidas*. Bertone, Rodolfo; De Giusti, Armando; Ardenghi, Jorge. Anales WICC. Mayo 1999. San Juan Argentina.

[BHAS 92] *The architecture of a heterogeneous distributed database management system: the distributed access view integrated database (DAVID).* Bharat Bhasker; Csaba J. Egyhazy; Konstantinos P. Triantis. CSC '92. Proceedings of the 1992 ACM Computer Science 20th annual conference on Communications, pages 173-179

[BURL 94] *Managin Distributed Databases. Building Bridges between Database Island.* Burleson, Donal. 1994

[DATE 94] *Introducción a los sistemas de Bases deDatos*. Date, C.J. Addison Wesley 1994.

[DIPA 99] *Un ambiente experimental para evaluación de Bases de Datos Distribuidas.* Di Paolo, Mónica; Bertone, Rodolfo; De Giusti, Armando. Paper presentado (aún no evaluado) en la ICIE 99. Fac. Ingeniería. UBA. Buenos Aires. Argentina

[LARS 95] *Database Directions. From relational to distributed, multimedia, and OO database Systems.* Larson, James. Prentice Hall. 1995

[MIAT 98] *Experiencias en el análisis de fallas en BDD.* Miaton, Ivana; Ruscuni, Sebastián; Bertone, Rodolfo; De Giusti, Armando. Anales CACIC 98. Neuquén Argentina.

[ÖZSU 91] *Principles of Distributed Database Systems.* Özsu, M. Tamer; Valduriez, Patric. Prentice Hall 1991.

[SCHU 94] *The Database Factory. Active databse for enterprise computing.* Schur, Stephen. 1994.

[SHET 90] *Federated database systems for managing distributed, heterogeneous, and autonomous databases.* Amit P. Sheth; James A. Larson. ACM Computing Surveys.Vol. 22, No. 3 (Sept. 1990), Pages 183-236

[SILB 98] *Fundamentos de las Bases de Datos.* Silbershatz; Folk. Mc Graw Hill. 1998.

[THOM 90] *Heterogeneous distributed database systems for production use.* Thomas, Charles; Glenn R. Thompson; Chin-Wan Chung; Edward Barkmeyer; Fred Carter; Marjorie Templeton; Stephen Fox; Berl Hartman. ACM Computing Surveys. Vol. 22, No. 3 (Sept. 1990), Pages 237-266

[UMAR 93] *Distributed Computing and Client Server Systems.* Umar, Amjad. Prentice Hall. 1993.

[WEB1] *www.seas.gwu.edu/faculty/shmuel/cs267/textbook/tpcp.html*

[WEB2] *www.sei.cmu.edu/str/descriptions/dtpc_body.html*

[WEB3] *www.datanetbbs.com.br/dclobato/textos/bddcs/parte2.html*

[WEB4] *www.itlibrary.com/library/1575211971/ch45.htm*

[WEB5] *www.itlibrary.com/library/1575211971/ch26.htm*

[WEB6] *java.sun.com/docs/books/tutorial/networking/sockets/definition.html*

[ZANC96] *Análisis de Replicación en Bases de Datos Distribuidas*, Marcelo Zanconi, Tesis de Magister en Ciencias de la Computación, Univ. Nac. Del Sur, Bahía Blanca, 1996