PERSEO:
AMBIENTE DE PROGRAMACION DE APLICACIONES TRANSACCIONALES DISTRIBUIDAS SOBRE BASES DE DATOS RELACIONALES

Consuelo FRANKY
José ABASOLO
Raúl CUCALON
Gloria ACOSTA
Santiago MAYA

Depto. de Ingeniería de Sistemas y Computación
Universidad de los Andes
Apartado Aéreo 4976
e-mail: cfranky@andescol.bitnet
Santafé de Bogotá- COLOMBIA

RESUMEN:

En este artículo se presenta la descripción de un proyecto en desarrollo conducente a un conjunto de facilidades para la programación de aplicaciones transaccionales distribuidas sobre bases de datos relacionales. Dichas bases de datos se suponen ubicadas en diversos computadores conectados por medio de una red de comunicación. Sobre la plataforma que constituye el conjunto disponible de sistemas manejadores de las bases de datos de una organización cualquiera, se construye un conjunto de servicios con una interfaz para el programador de aplicaciones distribuidas. El ambiente de programación resultante, "PERSEO", ofrece transparencia total a la fragmentación, a la distribución de las bases de datos y a la heterogeneidad de los sistemas manejadores relacionales presentando al programador la visión de un solo sistema centralizado. Para ello el ambiente asegura automáticamente las funciones relacionadas con la descomposición y optimización de consultas distribuidas. Además libera al programador del manejo complejo de transacciones distribuidas, incluyendo recuperación ante fallas y control de concurrencia, funciones todas que asegura el ambiente. Algunos de los aspectos ofrecidos por PERSEO no se encuentran aún disponibles en los sistemas manejadores de bases de datos distribuidas a nivel comercial, hecho que lo convierte en un proyecto novedoso y útil.

PALABRAS CLAVES: Bases de Datos Distribuidas. Sistemas transaccionales distribuidos. Programación de aplicaciones distribuidas sobre Bases de Datos.

1. INTRODUCCION

Indudablemente las redes de comunicación son en la actualidad un avance que se afianza cada vez más en los diferentes ámbitos de las organizaciones. El establecimiento de redes nacionales e internacionales impulsa el desarrollo de aplicaciones distribuidas, especialmente en aquellas organizaciones de carácter netamente distribuido que poseen departamentos o seccionales geográficamente dispersas y que requieren de la combinación de sus diferentes bases de datos para operar adecuadamente.

Sin embargo, la programación de aplicaciones distribuidas sobre el conjunto de bases de datos de una organización puede resultar una labor muy compleja si no se cuenta con ayudas adecuadas. Una primera alternativa para facilitar la labor del programador es contar con un ambiente basado en un Sistema Manejador de Bases de Datos Distribuidas que opere sobre el conjunto de computadores que componen la red de la organización. La adquisición de un ambiente de esta naturaleza, por parte de una organización, puede implicar tener que adquirir también sistemas manejadores locales y la conversión de bases de datos existentes para que puedan ser manejadas en el nuevo ambiente, resultando todo ello en un proceso muy costoso.

Una segunda alternativa menos onerosa es aprovechar la plataforma existente en la organización, referente al conjunto de sistemas manejadores de bases de datos locales, y desarrollar sobre esta plataforma un conjunto de servicios con una interfaz para el programador. Esta segunda alternativa es la que nosotros ofrecemos en el ambiente **PERSEO**: en este proyecto (que actualmente se encuentra en desarrollo) pretendemos elaborar librerías de funciones que el programador pueda incluir e invocar en sus aplicaciones con el fin de brindarle transparencia a la fragmentación y distribución de las bases de datos, con el resultado de que la programación resulta similar a la que se tiene en un solo sistema centralizado de bases de datos. Se trata entonces de aprovechar las facilidades que brindan los sistemas manejadores de bases de datos existentes en los diferentes computadores de la organización y por otro lado evitar la conversión de las bases de datos existentes.

El conjunto de funciones que constituyen el ambiente PERSEO facilita la programación de aplicaciones transaccionales distribuidas sobre un conjunto de bases de datos relacionales localizadas en computadores conectados a través de una red de comunicación. Las principales facilidades que ofrece PERSEO son las siguientes:

- **Transparencia a la fragmentación y a la distribución de las bases de datos relacionales**: la visión que tiene el programador es la de un conjunto de tablas globales relacionales ubicadas en un solo sistema centralizado.

- **Transparencia a la heterogeneidad de los sistemas manejadores relacionales**: el programador invoca dentro de su aplicación funciones tipo C que le proveen servicios de consulta y actualización sobre el conjunto de tablas globales que constituyen el sistema. Estas funciones son equivalentes en su expresividad al lenguaje SQL y constituyen una interfaz estándar a un conjunto de bases de datos relacionales administradas por diversos sistemas manejadores relacionales.

- **Descomposición y optimización de consultas y actualizaciones**: PERSEO provee funciones que transforman pedidos de consultas y actualizaciones expresadas por el programador en términos de tablas globales a conjuntos de operaciones sobre tablas locales en bases de datos específicas. En el caso de consultas se busca además minimizar el volumen de datos a transferir (mediante la selección óptima de los sitios encargados de realizar operaciones binarias como JOIN's entre tablas ubicadas en distintas bases de datos).

- **Ejecución de consultas y actualizaciones**: PERSEO se encarga de la transmisión de mensajes necesarios para la ejecución de una consulta o actualización que involucra múltiples bases de datos. Además sincroniza los procesos participantes y reúne los resultados que se le van a retornar a la aplicación. Para asegurar transparencia a la heterogeneidad de los diferentes sistemas manejadores relacionales, los mensajes llevan pedidos en términos de una interfaz estándar que luego debe transformarse en cada sitio al lenguaje SQL específico de la base de datos local.

Manejo de transacciones distribuidas: el programador puede invocar en su aplicación funciones especiales para delimitar el principio y el final de una transacción, la cual incluye en general varios pedidos de consultas y actualizaciones sobre las múltiples bases de datos que constituyen el sistema. PERSEO maneja automáticamente la ejecución simultánea de múltiples transacciones distribuidas solicitadas en diferentes sitios, liberando al programador de tener que prever funciones de recuperación ante fallas para asegurar la atomicidad de cada transacción y de control de concurrencia en el acceso a datos comunes (incluyendo el tratamiento de abrazos mortales distribuidos) : todas estas funciones las provee el ambiente.

La suma de facilidades que ofrecen PERSEO y los sistemas manejadores relacionales en los que se apoya son equivalentes a las facilidades que ofrecería un sistema manejador de bases de datos distribuidas. La novedad de este proyecto radica en el hecho de que los sistemas manejadores de bases de datos distribuidas disponibles actualmente a nivel comercial, no ofrecen todavía los aspectos relativos a fragmentación y al manejo completo de transacciones distribuidas. Desde el punto de vista del grado de evolución de los sistemas manejadores de bases de datos distribuidas presentado en [10] PERSEO corresponde al cuarto y máximo grado en razón de que maneja transacciones distribuidas compuestas de pedidos, en donde cada uno de ellos puede involucrar múltiples bases de datos de diferentes sitios.

El enfoque adoptado en PERSEO consistente en presentar al programador una interfaz de acceso a una Bases de Datos Distribuida a través de un conjunto de funciones C es también novedoso y atractivo. En efecto, este enfoque es más uniforme y más simple de usar que el enfoque tradicional de incluir claúsulas SQL mezcladas con instrucciones C en un programa de aplicación. Además permite independizar la aplicación de los diferentes sistemas manejadores relacionales participantes.

A continuación en la sección 2 de este artículo se presentan las facilidades de especificación de esquemas que ofrece PERSEO al administrador de un sistema de múltiples bases de datos. En la sección 3 se presenta la interfaz de servicios ofrecidos en PERSEO al programador de aplicaciones distribuidas sobre múltiples bases de datos. En la sección 4 se muestra la arquitectura del sistema PERSEO en términos de procesos Clientes - Servidores, con sus módulos asociados, encargados de ejecutar transacciones distribuidas solicitadas por las aplicaciones. En la sección 5 se describe el módulo Optimizador que provee las funciones de descomposición y optimización de consultas y actualizaciones distribuidas. En la sección 6 se describen los módulos encargados de coordinar y ejecutar transacciones distribuidas. Por último, en la sección 7 se plantean las proyecciones y futuras etapas del proyecto PERSEO.

2. FACILIDADES DE ESPECIFICACION DE ESQUEMAS QUE SE OFRECEN AL ADMINISTRADOR DE UN SISTEMA DE MULTIPLES BASES DE DATOS

El ambiente PERSEO provee un programa Utilitario que permite a un administrador de un sistema de múltiples bases de datos relacionales especificar los esquemas global, de fragmentación y de distribución. Siguiendo las definiciones de [3] el administrador especificará los esquemas bajo el modelo relacional con las siguientes opciones:

- Esquema Global: corresponde a especificar el esquema relacional de tablas globales que componen el conjunto de bases de datos. Este esquema corresponde a la visión centralizada del sistema que se le quiere dar al programador de aplicaciones (visión denominada base de datos global).

- Esquema de Fragmentación: el administrador podrá especificar una fragmentación horizontal (simple y/o derivada hasta un nivel de profundidad) de las tablas globales. En la primera versión de PERSEO no se soportará la fragmentación vertical ni la combinación de la fragmentación horizontal con la vertical.

- Esquema de Distribución: corresponde a especificar el sitio de localización de cada uno de los fragmentos de las tablas globales. En la primera versión de PERSEO no se soportará la replicación de fragmentos.

El programa Utilitario materializa las anteriores especificaciones en un Catálogo Global del sistema que debe replicarse en cada uno de los sitios para poder llevar a cabo las funciones de descomposición y optimización de consultas y de actualizaciones distribuidas.

Adicionalmente, el programa Utilitario permite actualizar el Catálogo Global cuando hay creación y destrucción de tablas e igualmente permite mantener y actualizar estadísticas sobre las tablas con miras a proveer información más precisa a las funciones de optimización.

3. INTERFAZ DE SERVICIOS OFRECIDA AL PROGRAMADOR DE APLICACIONES DISTRIBUIDAS SOBRE BASES DE DATOS

El programador de aplicaciones distribuidas sobre un conjunto de bases de datos relacionales cuenta en PERSEO con una librería de funciones tipo C que le ofrece transparencia a la fragmentación, distribución y heterogeneidad de los sistemas manejadores relacionales. Adicionalmente cuenta con funciones que le permiten delimitar transacciones distribuidas (que no deben estar anidadas). En total son 4 las funciones de PERSEO que el programador puede invocar dentro de su aplicación (la cual debe escribir en lenguaje ANSI C [11]):

- **begin_transaction**: con la invocación de esta función la aplicación señala el inicio de una transacción distribuida que contendrá solicitudes de consultas y actualizaciones sobre tablas globales intercaladas con operaciones locales (i.e. interacción con el usuario y manipulación de los datos resultantes de las consultas).

- **end_transaction**: con la invocación de esta función la aplicación señala el fin de una transacción distribuida. El sistema asegurará la atomicidad de la transacción completando sus efectos sobre los datos involucrados, o anulando esos efectos en caso de que haya problemas como caídas de sitios participantes o conflictos de concurrencia.

- **sql**: con la invocación de esta función la aplicación solicita una consulta o actualización sobre tablas globales. El único parámetro de esta función es una cadena con la sentencia SQL solicitada, la cual puede incluir las operaciones SELECT, INSERT o DELETE. La operación SELECT podrá usar funciones agregadas (i.e. AVG, SUM, COUNT, etc.) así como cláusulas de ordenamiento (GROUP BY, HAVING).

  En general la función sql permite a la aplicación solicitar cualquier operación del lenguaje estándar SQL [10] excepto la operación UDPDATE que será tratada en una siguiente versión del proyecto.

- **sqlv**: esta función permite también solicitar una consulta o actualización sobre tablas globales y adicionalmente permite usar en la sentencia SQL parámetros variables cuyos valores se conocen solo en tiempo de ejecución. Un parámetro de esta función sqlv podrá indicar opcionalmente una función que la aplicación quiere ejecutar sobre cada una de las tuplas resultantes de una consulta (por ejemplo una función para imprimir dichas tuplas o hacer cálculos con ellas) : esta función debe estar definida dentro de la aplicación.

En resumen, la aplicación escrita por el programador se ocupa de la interacción local con el usuario mientras que el manejo de todos los aspectos distribuidos del sistema corre por cuenta de las funciones provistas por el ambiente PERSEO. Es de esperar que todos los accesos a las bases de datos del sistema, incluyendo la local, se hagan a través de la interfaz de servicios descritos en esta sección. Esta interfaz usada por la aplicación se denomina "IABDD" (Interfaz de Acceso a Bases de Datos Distribuidas) y se diferenciará de la interfaz "IABD" (Interfaz de Acceso a una Base de Datos) usada por los procesos del sistema para accesar una base de datos particular con transparencia a la heterogeneidad de los diferentes sistemas manejadores relacionales que participan en el sistema.

4. ARQUITECTURA DEL SISTEMA PERSEO

Una aplicación distribuida sobre múltiples bases de datos que invoque los servicios de PERSEO a través de la interfaz IABDD, da lugar en ejecución a un proceso CLIENTE que va a interactuar con procesos SERVIDORES provistos por el sistema.

Conceptualmente un proceso CLIENTE está compuesto por 3 módulos:

- la **aplicación** propiamente dicha: Interactúa con el usuario e invoca los servicios distribuidos de PERSEO a través de la interfaz IABDD (Interfaz de Acceso a Bases de Datos Distribuidas).

- el módulo **Coordinador**: Sincroniza la ejecución de los servicios distribuidos invocados por la aplicación. Para ello invoca un módulo Optimizador que se encarga de elaborar el plan de ejecución de cada consulta o actualización distribuida, y luego se sincroniza con los procesos SERVIDORES del sistema para llevar a cabo dicho plan. Igualmente se sincroniza con los procesos SERVIDORES para asegurar la ejecución consistente de la transacción distribuida solicitada por la aplicación.

- el módulo **Optimizador**: Descompone cada solicitud de consulta o actualización de tablas globales en conjuntos de instrucciones que el Coordinador debe dirigir a bases de datos específicas. La descomposición corresponde a una transformación de una solicitud de servicio expresada en interfaz IABDD (Interfaz de Acceso a Bases de Datos Distribuidas) a un conjunto de solicitudes de servicios expresadas en interfaz IABD (Interfaz de Acceso a una Base de Datos) destinadas a sitios específicos. Para llevar a cabo esta función, el Optimizador se basa en la información sobre esquemas de la base de datos global contenida en el Catálogo Global y utiliza como criterio de optimización minimizar el volumen de datos a transferir (criterio adecuado para sistemas geográficamente distribuidos [3]).

En términos de implantación, el proceso CLIENTE proviene de la compilación del programa de aplicación junto con librerías provistas por el sistema, las cuales corresponden a los módulos Coordinador y Optimizador.

Cada proceso SERVIDOR interactúa con el sistema manejador relacional de una base de datos local para ejecutar las instrucciones provenientes de los procesos CLIENTES del sistema (cada CLIENTE está asociado a una transacción distribuida en curso de ejecución). Los pedidos de las instrucciones llegan expresados en interfaz IABD que es independiente de cualquier base de datos local. El SERVIDOR debe entonces transformar cada pedido en intrucciones SQL propias del sistema manejador relacional local y para ello utiliza una librería IABD particular para cada sistema manejador. Adicionalmente ejecuta instrucciones de control asociadas a la ejecución consistente de transacciones distribuidas.

La figura 1 muestra la arquitectura del sistema PERSEO en términos de procesos CLIENTES - SERVIDORES y de sus módulos asociados. También se ilustra la ubicación de las interfaces IABDD (aplicación - PERSEO) e IABD (SERVIDOR - sistema manejador local).

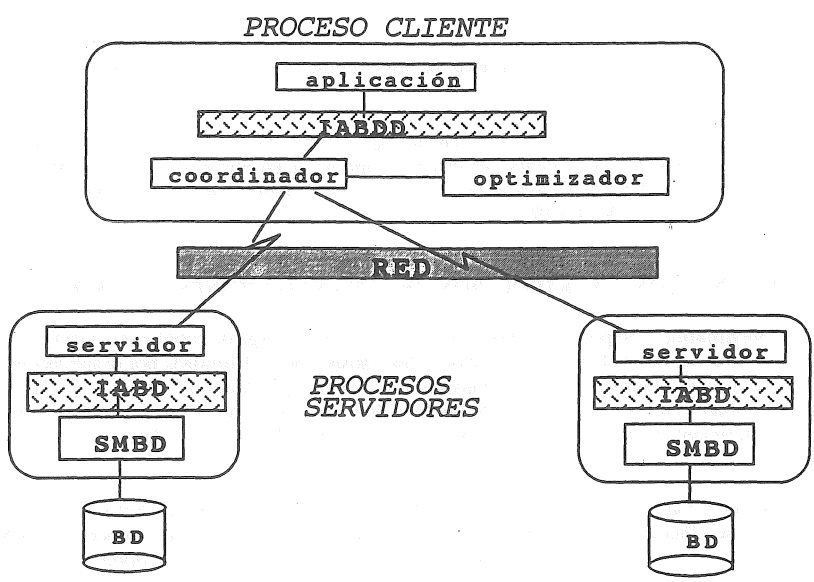

Figura 1. Arquitectura del sistema PERSEO.

Para múltiples transacciones simultáneas se tendría un proceso CLIENTE en el sitio origen de cada transacción. Por otro lado hay un proceso SERVIDOR en cada sitio que maneje una base de datos del sistema. Un proceso CLIENTE requiere comunicarse en general con procesos SERVIDORES remotos y también con el proceso SERVIDOR local.

5. DESCOMPOSICION Y OPTIMIZACION DE CONSULTAS Y ACTUALIZACIONES DISTRIBUIDAS

El módulo Optimizador, componente del proceso CLIENTE, se encarga de descomponer y optimizar las consultas y actualizaciones distribuidas que se quieran realizar sobre las múltiples bases de datos que componen el sistema.

En el ambiente PERSEO, cada pedido de consulta o actualización sobre tablas globales, hecho por una aplicación, genera una invocación al módulo Optimizador, a quien se le informa como parámetro una cadena con la sentencia SQL que se quiere ejecutar. En la primera versión de PERSEO la interfaz IABDD permite al programador hacer pedidos SQL que incluyen operaciones SELECT, INSERT y DELETE, como se explicó en la sección 3 (Interfaz IABDD de servicios ofrecidos al programador).

Con base en la información de esquemas (global, de fragmentación y de distribución) contenida en el catálogo global, el Optimizador genera el plan óptimo de ejecución de la sentencia SQL en el cual identifica los SERVIDORES que deben participar y las operaciones que cada uno de ellos debe ejecutar. Estas operaciones se expresan en términos de la interfaz IABD, independiente de cualquier sistema manejador relacional. El criterio de optimización, como ya se mencionó, es minimizar el volumen de datos a transferir por la red.

Para generar el plan óptimo de ejecución de una sentencia SQL, el módulo Optimizador cubre las siguientes etapas (basándose en la teoría de [3] y en algoritmos de tranformación SQL - algebra relacional [4])):

- transformación de la sentencia SQL en el árbol equivalente de operaciones del álgebra relacional sobre tablas globales (árbol global)

- transformación del árbol global en un árbol canónico expresado en términos de fragmentos

- optimización y materialización del árbol canónico mediante la eliminación de subpartes que conducen a resultados nulos y mediante la selección de sitios óptimos para realizar las operaciones binarias JOIN's

- transformación del árbol óptimo en sentencias SQL expresadas en interfaz IABD y destinadas a SERVIDORES específicos. Además, en este plan óptimo de ejecución se incluyen operaciones que deben realizar los SERVIDORES referentes a la creación de tablas temporales, envío y recepción de las mismas por otros SERVIDORES.

6. COORDINACION Y EJECUCION DE TRANSACCIONES DISTRIBUIDAS

El módulo **Coordinador**, componente del proceso CLIENTE, se encarga de coordinar la ejecución de cada transacción distribuida solicitada por la aplicación. Una transacción es delimitada por la aplicación a través de las funciones Begin_Transaction y End_Transaction y contiene solicitudes de consultas y actualizaciones sobre tablas globales (invocadas a través de las funciones sql y sqlv de la interfaz IABDD) junto con instrucciones puramente locales (interacción con el usuario, manipulación de los datos que resultan de una consulta, etc).

La ejecución de una transacción distribuida en el ambiente PERSEO se rige por el Protocolo de Validación en dos etapas ("Two Phase Commit Protocol") [6] y exige la sincronización entre el proceso CLIENTE del sitio donde se origina la transacción y los procesos SERVIDORES de los sitios en donde residen las bases de datos involucradas por la transacción.

Las siguientes son las principales funciones que realiza el módulo Coordinador en relación a cada transacción distribuida solicitada por la aplicación:

- **Generación de una estampilla** para identificar la transacción de forma única respecto a las demás transacciones distribuidas del sistema.

- **Distribución del plan óptimo de ejecución de una consulta o actualización**: Después de invocar al módulo Optimizador se obtiene el plan de ejecución de una consulta o actualización sobre tablas globales. El módulo Coordinador envía a cada uno de los SERVIDORES participantes el conjunto de operaciones del plan que debe ejecutar, junto con la estampilla de la transacción.

- **Aplicación de funciones a los resultados de las consultas globales**: en caso de que el programa de aplicación especifique una función en una solicitud de consulta global, el módulo Coordinador la aplicará a cada una de las tuplas que son resultado de la consulta antes de devolver el control a la aplicación propiamente dicha.

- **Atomicidad de la transacción**: Ante una solicitud de End_Transaction hecha por la aplicación, el módulo Coordinador realiza las etapas de validación de la transacción enviando los mensajes correspondientes a los SERVIDORES participantes. Bajo el Protocolo de Validación en dos etapas, el proceso CLIENTE origen de la transacción debe lograr que todos los SERVIDORES participantes completen los efectos de la transacción ("commit") o todos los anulen ("rollback") a fin de asegurar la atomicidad de la transacción (aún en caso de fallas).

- **Control de concurrencia y prevención de abrazos mortales entre transacciones** : Para asegurar el control de concurrencia entre transacciones que accesan datos comunes, cada transacción utiliza candados para reservar las tablas que requiere (cumpliendo la propiedad de reservación en "dos fases" [3]). Para resolver situaciones de abrazos mortales entre dos o más transacciones, en PERSEO se aplica una técnica de prevención de abrazos mortales.

En cada uno de los sitios del sistema se tiene un proceso SERVIDOR capacitado para ejecutar sobre la base de datos local diversas solicitudes provenientes de múltiples transacciones distribuidas.

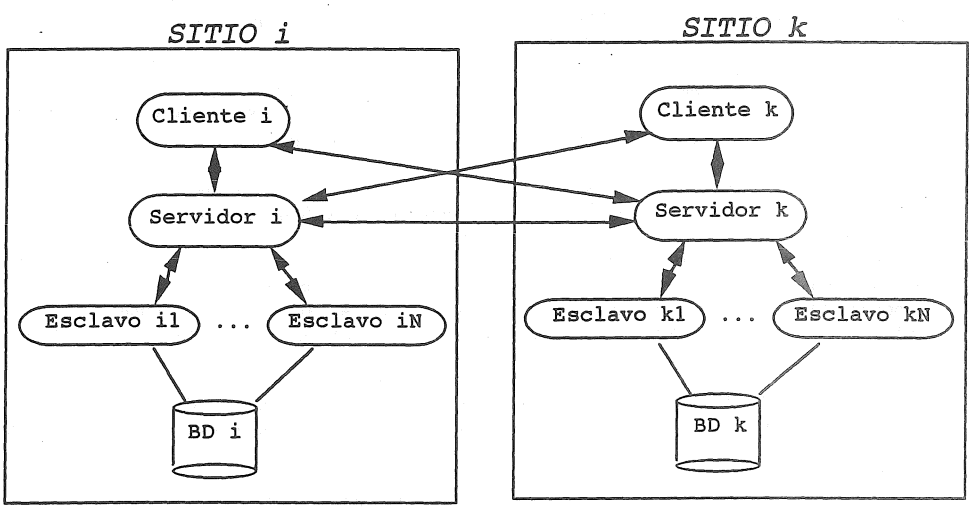

Figura 2. Relación de comunicación entre procesos CLIENTES, SERVIDORES y ESCLAVOS en el sistema PERSEO.

Para asegurar una buena eficiencia en la atención de las solicitudes de las transacciones, el proceso SERVIDOR activa un proceso ESCLAVO por cada transacción en curso que involucre su sitio. De esta

manera se pueden atender simultáneamente múltiples solicitudes de transacciones distintas. El proceso SERVIDOR centraliza, sin embargo, la función de control de concurrencia entre esas transacciones basándose en el manejo de candados. Al finalizar una transacción, finalizan también sus procesos ESCLAVOS activados en los sitios participantes. Son los procesos ESCLAVOS y no los SERVIDORES quienes utilizan una librería IABD (Interfaz de Acceso a una Base de Datos) específica para el sistema manejador relacional local.

La figura 2 ilustra la relación de comunicación entre procesos CLIENTES, SERVIDORES y ESCLAVOS correspondientes a diversas transacciones.

El plan global de una consulta o actualización invocada por una aplicación da lugar a solicitudes de operaciones sobre bases de datos específicas y a solicitudes de envíos o recepciones de tablas temporales que resultan de ejecutar dichas operaciones. Por esta razón, el proceso SERVIDOR debe estar en capacidad de enviar la tabla temporal resultante de una operación de consulta sobre la base de datos local hacia otro sitio que requiera dicha tabla. Igualmente el proceso SERVIDOR debe estar en capacidad de recibir una tabla temporal y avisarle al proceso ESCLAVO que la esté esperando.

En el Ambiente PERSEO se aprovechan las facilidades de los sistemas manejadores relacionales disponibles en los sitios con el fin de asegurar la ejecución consistente de transacciones distribuidas. Por ejemplo, para completar los efectos de una transacción en un sitio, el proceso ESCLAVO asociado invocará la operación "Commit" o "Rollback" que le ofrece el sistema manejador local. Sin embargo, ciertas funciones como el control de concurrencia de transacciones distribuidas, requieren replicar funciones de los sistemas manejadores: en este caso, es necesario que cada proceso SERVIDOR registre los candados que diversas transacciones ponen sobre las tablas de la base de datos local a fin de poder prevenir abrazos mortales distribuidos (el manejo de candados por parte de un sistema manejador permite prevenir o detectar abrazos mortales únicamente locales pero no distribuidos).

7. PROYECCIONES Y EXTENSIONES PREVISTAS

El proyecto PERSEO se desarrolla actualmente sobre una red local de computadores UNIX comunicados bajo protocolos TCP/IP [5]. La primera versión debe soportar aplicaciones distribuidas sobre bases de datos relacionales ORACLE y UNIFY ubicadas respectivamente en un computador Unysis U6000 y en un computador Digital MicroVax II. Para lograr el acceso transparente a la heterogeneidad de los sistemas manejadores relacionales se utilizan librerías que soportan la interfaz IABD para ORACLE y para UNIFY desarrolladas en proyectos paralelos ([2], [12]).

La comunicación remota entre procesos CLIENTES y SERVIDORES utiliza la interfaz de sockets sobre protocolos TCP/IP y la comunicación local CLIENTE - SERVIDOR y SERVIDOR - ESCLAVOS utiliza el mecanismo de Colas de Mensajes ("Messages Queues") propio de UNIX [16].

Para independizar el proyecto PERSEO de los mecanismos específicos que se utilicen para la comunicación entre procesos, su segunda versión se implantará en el lenguaje y ambiente JOYCE+ [8] : este lenguaje es una extensión de C que permite una programación de alto nivel de los procesos de un sistema distribuido, de manera independiente a la red de comunicación en donde van a operar. El ambiente provisto para el lenguaje JOYCE+ [1] ofrecerá todos los mecanismos necesarios para brindar transparencia al programador respecto a la red específica en donde se va a trabajar. Actualmente el lenguaje JOYCE+ se utiliza en el proyecto PERSEO como lenguaje de especificación de los procesos CLIENTES, SERVIDORES y ESCLAVOS que lo componen.

Las ideas principales del proyecto PERSEO provienen de un proyecto anterior "TESEO" ([7], [13], [14]) que fue implantado sobre una red de microcomputadores que presentaba facilidades muy restringidas de manejo de bases de datos.

Como versiones futuras del proyecto PERSEO se planea el tratamiento de la operación SQL "UPDATE", el manejo de tablas replicadas, el manejo dinámico del Catálogo Global y herramientas que faciliten el diseño de los esquemas de fragmentación y de distribución de la base de datos global. Igualmente se buscará implantar herramientas que faciliten la integración de bases de datos heterógeneas. Estos objetivos plantean retos novedosos teniendo en cuenta que casi ninguna de estas facilidades se ofrecen actualmente en

los sistemas manejadores de bases de datos distribuidas disponibles a nivel comercial: de hecho, se observa un gran vacío en lo que respecta a ayudas al diseño.

En conclusión, con el proyecto PERSEO pretendemos crear un Ambiente que facilite la programación de aplicaciones distribuidas sobre bases de datos, aprovechando facilidades de sistemas manejadores relacionales existentes en distintos sitios y sin requerir la inversión costosa de adquirir o construir todo un sistema manejador de bases de datos distribuidas.

En las versiones futuras del proyecto PERSEO buscaremos crear un Ambiente que apoye cada una de las etapas del desarrollo de una aplicación distribuida sobre múltiples bases de datos, tanto lo que respecta al diseño de la distribución de los datos como al diseño de la distribución del procesamiento.

BIBLIOGRAFIA

[1]: A. Arenas S., C. Franky; "Ambiente JOYCE+ de desarrollo de sistemas distribuidos para entorno UNIX - TCP/IP", Memo CIFI de la Facultad de Ingeniería de la Universidad de los Andes, Bogotá-Colombia, 1991.

[2]: R. Blanco; "Interface de acceso a un sistema manejador de base de datos relacional desde un programa C", tesis de pregrado en Ingeniería de Sistemas y Computación, Universidad de los Andes, Bogotá - Colombia, 1992.

[3]: S. Ceri, G. Pelagatti; "Distributed Databases: Principles and Systems"; McGraw-Hill 1984.

[4]: S. Ceri, G. Gotlob; "Translating SQL into relational algebra: optimizations, semantics and equivalence of SQL queries", IEEE Transactions on Software Engineering, Vol. SE-11 N° 4, Abril de 1984.

[5]: D. E. Comer; "Internetworking with TCP/IP", Prentice-Hall 1988.

[6]: G. F. Coulouris, J. Dollimore; "Distributed Systems", Addison-Wesley 1988.

[7]: C. Franky ; "TESEO: Ambiente de programación de sistemas transaccionales distribuidos", XIII Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Noviembre de 1987.

[8]: C. Franky.; "JOYCE+ : Model an Language for multi-site Distributed Systems", IEEE-ACM Second International Symposium on Databases in Parallel and Distributed Systems, Dublin Irlanda, Julio 1.990. También como Memo CIFI de la Facultad de Ingeniería de la Universidad de los Andes, 1990.

[9]: C. Franky; "Desarrollo de aplicaciones distribuidas sobre bases de datos", Revista ACIS de Sistemas, Bogotá - Colombia, Agosto de 1991.

[10] : J. R. Groff, P. N. Weinberg; "Aplique SQL", McGraw-Hill 1991.

[11]: B. Kernighan, D. Ritchie; "The C programming language", Prentice-Hall 1988.

[12] : C. Jiménez, R. Blanco; "IABD: Diseño e implantación de un API para el acceso a un sistema manejador de base de datos relacional desde un programa C", XVIII Conferencia Latinoamericana de Informática (CLEI), Las Palmas de Gran Canaria - España, Agosto de 1992.

[13]: P. Levy ; "Compilador de expresiones de lenguaje relacional para el sistema TESEO", XIII Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Noviembre de 1987.

[14]: H. Sin ; "Un sistema de consulta para bases de datos distribuidas", tesis de pregrado en Ingeniería de Sistemas y Computación, Universidad de los Andes, Bogotá - Colombia, 1986.

[15]: M. Sloman, J. Kramer.; "Distributed Systems and Computer Networks", Prentice-Hall, 1987.

[16]: R. Stevens; "Unix Network programming", Prentice-Hall 1990.