Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Tecnologías Avanzadas

upiita-ipn

Unidad de Aprendizaje: Bases de Datos Distribuidas.

Profesor: Carlos de la Cruz Soza.

Reporte de lectura del tema arquitecturas de BDD.

Integrantes equipo 5:

Arciniega Noriega Carlos Josué

González Plata José Enrique

Morales Rodríguez Diego Emilio

Grupo: 3TM3.

Índice

Introducción

[1]"La arquitectura de un sistema refleja la estructura del sistema subyacente. Define los diferentes componentes del sistema, las funciones de estos componentes y las interacciones y relaciones generales entre ellos. Este concepto es válido tanto para los sistemas informáticos generales como para los sistemas de software."

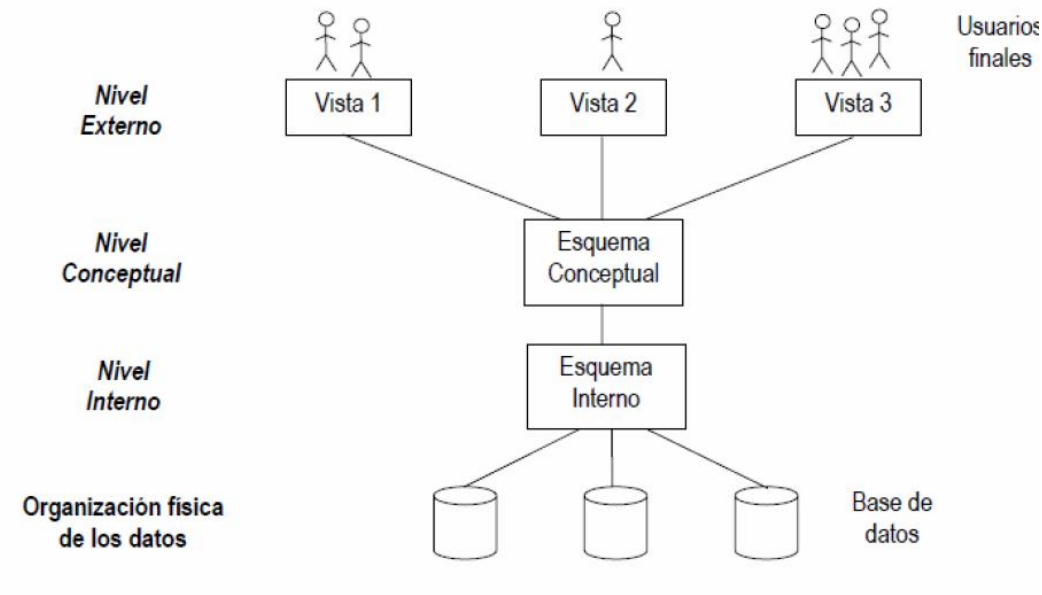
Arquitectura ANSI/SPARC

En la década de los 70s el comité de computación y procesamiento de información (X3) del ANSI estableció un grupo de estudio en sistemas de gestión de bases de datos, auspiciado por el SPARC, con el propósito de establecer estándares en esa área. El framework arquitectónico propuesto es conocido como "arquitectura ANSI/SPARC".

[2]"El framework de ANSI/SPARC DBMS describe un sistema gestor de bases de datos en términos de interfaces, roles de personal, funciones de procesamiento, y flujos de información dentro del sistema."

Niveles de Representación y Descripción de datos

El framework ANSI/SPARC propone una arquitectura con "coexistencia" de tres niveles para DBMSs. Una versión simplificada se muestra en la siguiente imagen.



Hay tres vistas de datos: la *vista externa*, la cual es del usuario final, quien puede ser un programador; la *vista interna*, esa del sistema o maquina; y la *vista conceptual*, la de la empresa. Por cada vista se requiere definir un esquema.

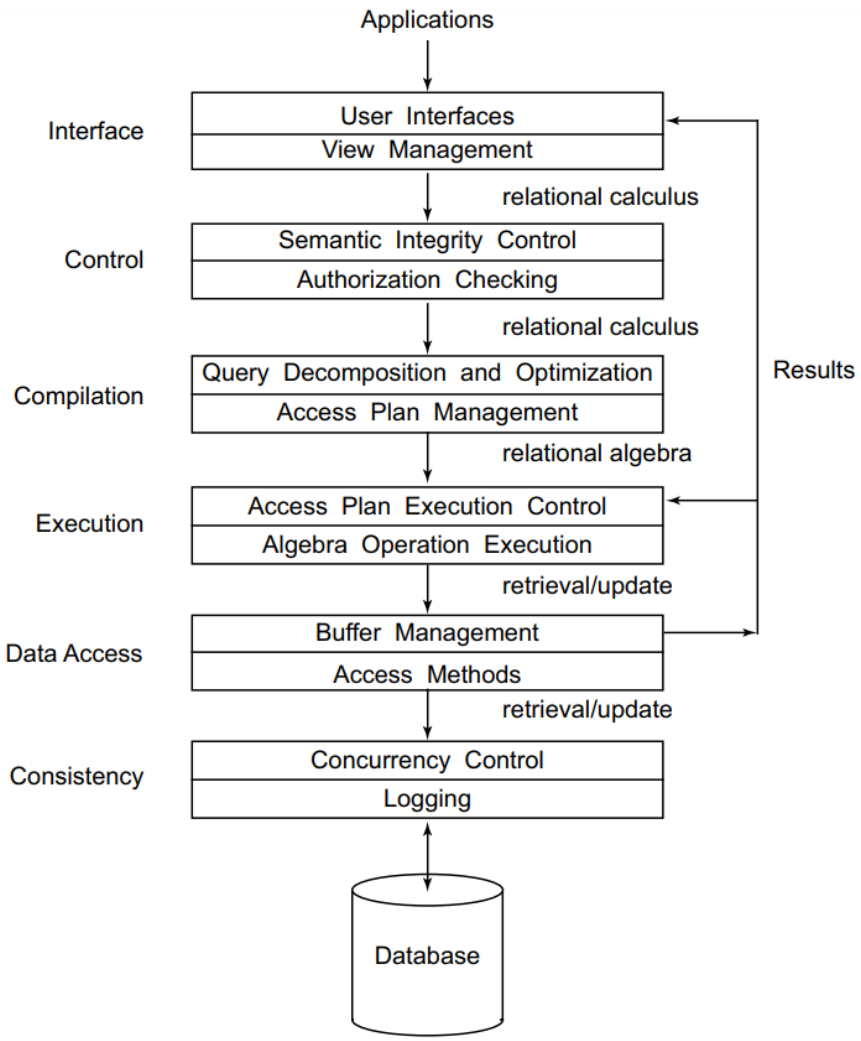
En el nivel más bajo se encuentra la vista interna, la cual trata con la definición física y organización de los datos. En el otro extremo está la

vista externa la cual se encarga de la forma en la que los usuarios ven la base de datos. Entre estos dos niveles está el esquema conceptual, el cuál es una definición abstracta de la base de datos. Esta parte es la vista del "mundo real" de la empresa siendo modelada en la base de datos.

Arquitectura Centralizada Genérica

Cuando el DBMS se ejecuta en una computadora este es interconectado con dos componentes: El subsistema de comunicación y el sistema operativo.

El subsistema de comunicación permite a la DBMS intercomunicarse con otros subsistemas para poder comunicarse con las aplicaciones. Por ejemplo, el monitor de la terminal necesita comunicarse con la DBMS para poder ejecutar las transacciones. Y el sistema operativo proporciona la interfaz entre el DBMS y la computadora.



Como se puede observar en la imagen anterior, la DBMS funciona en 6 capas que son la interfaz, el control, la compilación, la ejecución, el acceso a los datos y la gestión de consistencia.

La capa de interfaz se encarga administrar las interfaces de las aplicaciones.

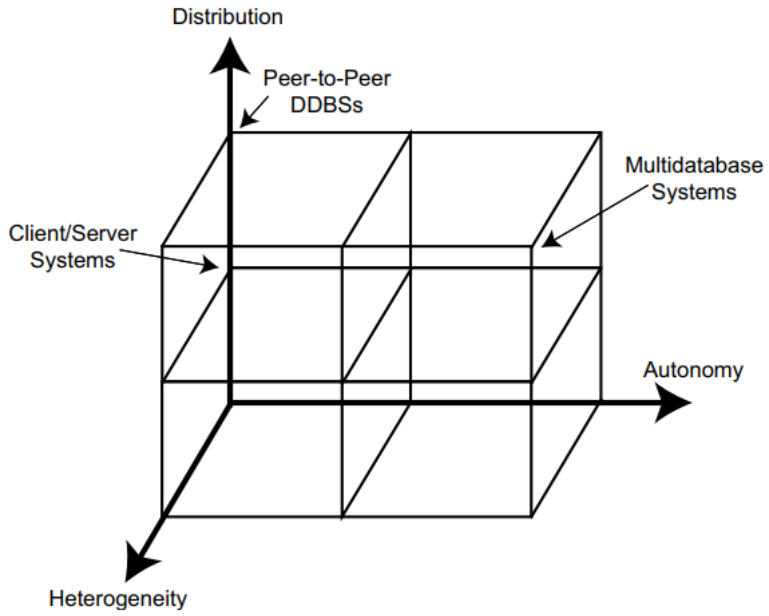
La capa de control se encarga de controlar la consulta agregando predicados de integridad semántica y predicados de autorización. Por Integridad Semántica queremos decir que es la adaptación de la base de datos con las restricciones derivadas de nuestro conocimiento de lo que está y no está permitido en aquella parte del universo que esté representada por los datos de la base de datos. El mantenimiento de la Integridad semántica implica evitar que se inserten en la base de datos, los datos que representen un estado no permitido del universo. Y el predicado de autorización se encarga de autorizar la consulta, valga la redundancia.

La capa de compilación se encarga de asignar la consulta a una secuencia optimizada de operaciones del sistema binario. Descompone la consulta en un árbol de operaciones de algebra e intenta encontrar el orden óptimo de las operaciones. La salida de esta capa es una consulta expresada sistema binario.

La capa de ejecución se encarga de dirigir la ejecución administrando la gestión de transacciones y sincronizando las operaciones algebraicas. Interpreta las operaciones llamando a la capa de acceso de datos a través de las solicitudes de recuperación y actualización. La capa de acceso a datos gestiona las estructuras de datos que implementan los archivos, índices, etc. También gestiona los búferes almacenando en caché los datos a los que se accede con mayor frecuencia. El uso cuidadoso de esta capa minimiza el acceso a los discos para obtener o escribir datos. Finalmente, la capa de consistencia administra el control de concurrencia y el registro de solicitudes de actualización. Esta capa permite la recuperación de transacciones, sistemas y medios después de una falla.

Arquitecturas para una Bases de Datos Distribuidas

Ahora consideramos las posibles formas en que se puede diseñar una DBMS distribuida. La imagen siguiente organiza los sistemas con respecto a la autonomía de los sistemas locales, su distribución y su heterogeneidad.



Autonomía

La autonomía se refiere a la distribución de control, no de los datos. Indica el grado en que los DBMS individuales puedan operar de forma independiente. Los DBMS individualmente pueden intercambiar información, ejecutar transacciones de forma independiente y se permite modificarlas. Para que un sistema distribuido sea autónomo debe cumplir los siguientes requisitos:

- Las operaciones locales de los DBMS individuales no se ven afectadas por su participación en el sistema distribuido.
- La forma en que los DBMS individuales procesan las consultas y las optimizan no deben verse afectadas por la ejecución de consultas globales que acceden a múltiples bases de datos.
- La funcionalidad del sistema no debe verse comprometida cuando los DBMS individuales se unen o abandonan el sistema distribuido.

Por otro lado, existen diferentes dimensiones de autonomía, que son:

- Autonomía de diseño: los DBMS individuales son libres de usar los modelos de datos y las técnicas de gestión de transacciones que prefieran.
- Autonomía de comunicación: Cada uno de los DBMS es libre de tomar su propia decisión sobre qué tipo de información quiere proporcionar a los otros DBMS o al software que controla su ejecución global.
- Autonomía de ejecución: Cada DBMS puede ejecutar las transacciones de la forma que desee.

También existen los DBMS semiautónomos que consisten en DBMS que pueden operar de forma independiente, sus datos locales se puedan compartir. Cada uno de estos DBMS determina qué partes de su propia base de datos harán accesibles a los usuarios de otros DBMS. No son sistemas completamente autónomos porque necesitan ser modificados para permitirles intercambiar información entre sí. La última alternativa que consideramos es el aislamiento total, donde los sistemas individuales son DBMS independientes que no conocen la existencia de otros DBMS. En dichos sistemas, el procesamiento de transacciones de usuarios que acceden a múltiples bases de datos es especialmente difícil ya que no existe un control global sobre la ejecución de DBMS individuales.

Distribución

La distribución hace referencia al manejo y localización física de los datos. En la arquitectura cliente/servidor el servidor se encarga de los deberes del manejo de datos mientras que los clientes proveen aplicaciones de entorno. En el caso de la arquitectura peer-to-peer cada nodo puede encargarse de todas las funciones del sistema gestor de bases de datos, comunicando con otras máquinas consultas y transacciones.

Heterogeneidad

El fenómeno de la heterogeneidad no es un fenómeno propio de los sistemas de bases de datos, sino una característica de un sistema distribuido, en este sentido, existe heterogeneidad tanto en hardware (Arquitectura de procesadores, cables y conectores, memorias, etc.) como en software (Tipos de datos, tamaño de datos, Sistema operativo, acceso a memoria, set de instrucciones, etc.).

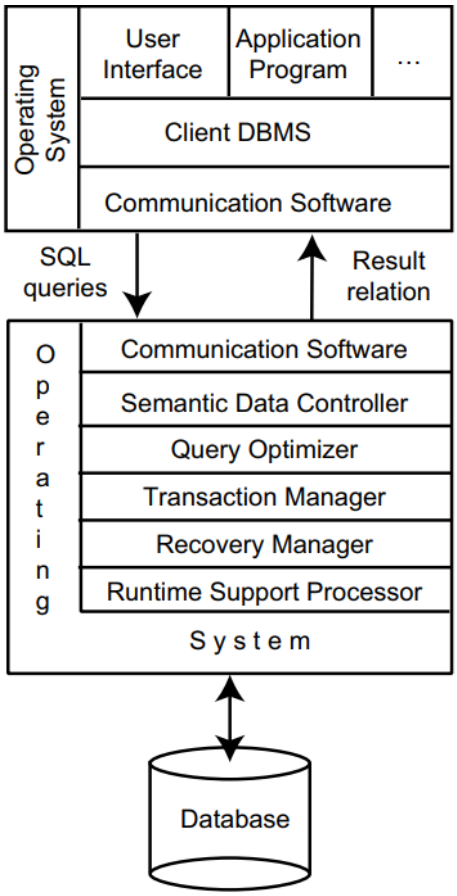
Los aspectos heterogéneos más relevantes en el contexto de bases de datos son, la heterogeneidad de los lenguajes de consulta y los paradigmas de acceso a los datos, incluso dentro del estándar SQL existen variaciones no solo a nivel de implementación, sino a nivel sintáctico ya que cada implementación (MySQL, MS-SQL-SERVER, POSTGRESQL, entre otros. Dentro de un mismo gestor podemos

tener diferentes mecanismos de almacenamiento de datos, tal es el caso de MySQL, en el que los datos se pueden almacenar mediante InnoDB, MyISAM y otros mecanismos)

Hay que resaltar que el problema de la heterogeneidad no es un problema trivial, por ejemplo, ¿Qué debería suceder si de un dato tengo un float 64 y del otro lado solo soporto floats de 32 bits? Este es un concepto importante en el contexto de las bases de datos distribuidas, de ahí que tengamos que seleccionar el driver correcto para poder realizar la comunicación entre servidores de bases de datos y que existan implementaciones diferentes, para diferentes arquitecturas y sistemas operativos.

Sistemas Cliente/Servidor

A principios de la década de 1990 surgió los DBMS Cliente/Servidor, tuvo un impacto muy significativo dentro de la tecnología DBMS. La idea del Cliente/Servidor es simple y elegante, se basa en distinguir las funciones del servidor y las funciones del cliente. Esto proporciona una arquitectura de dos niveles que facilita la gestión de los DBMS. Pero DBMS cliente/servidor no se refiere a procesos, si no a maquinas(usuarios). Por lo tanto, nos enfocaremos que software debe ejecutarse en las máquinas cliente y en las máquinas servidor. Dicho de esta manera, el problema es más claro y podemos comenzar a estudiar las diferencias en la funcionalidad del cliente y del servidor. En los sistemas relacionales, el servidor realiza la mayor parte del trabajo de gestión de datos. Esto significa que todo el procesamiento y la optimización de consultas, la gestión de transacciones y la gestión de almacenamiento se realizan en el servidor. El cliente, además de la aplicación y la interfaz de usuario, tiene un módulo de cliente DBMS que es responsable de administrar los datos que se almacenan en caché en el cliente y administrar los bloqueos de transacciones que también pueden haberse almacenado en caché.



La arquitectura de la imagen anterior es bastante común en los sistemas relacionales donde la comunicación entre los clientes y el (los) servidor(es) se encuentra al nivel de las sentencias SQL. En otras palabras, el cliente pasa consultas SQL al servidor sin intentar comprenderlas. El servidor hace la mayor parte del trabajo y devuelve los resultados al cliente. Hay varios tipos arquitectura cliente/servidor. El más simple es el caso en el que solo hay un servidor al que acceden varios clientes. A esto lo llamamos cliente múltiple/servidor único. Desde la perspectiva de la gestión de datos, esto no es muy diferente de las bases de datos centralizadas, ya que la base de datos se almacena en una sola máquina (el servidor). Sin embargo, existen diferencias con respecto a los sistemas centralizados en la forma en que se ejecutan las transacciones y se administran los cachés. Una arquitectura cliente/servidor más sofisticado es aquella en la que hay múltiples servidores en el sistema. En este caso, existen dos estrategias de gestión: cada cliente gestiona su propia conexión con su propio servidor o cada cliente conoce solo su "servidor doméstico" y luego se comunica con otros servidores. El primer enfoque simplifica el código del servidor, pero carga las máquinas cliente con responsabilidades adicionales. Esto conduce a lo que se ha denominado sistemas de “cliente pesado”. El último enfoque concentra la funcionalidad de gestión de datos en los servidores. Por lo tanto, la

transparencia del acceso a los datos se proporciona en la interfaz del servidor, lo que lleva a "clientes ligeros".

Sistemas Peer-to-Peer

[3]"Una arquitectura de red distribuida puede ser llamada una red peer-to-peer (P-to-P, P2P0, etc.) si los participantes comparten una parte de sus propios recursos de hardware (poder de procesamiento, capacidad de almacenamiento, capacidad de enlace de red, impresoras…). Estos recursos compartidos son necesarios para proveer el servicio y contenido ofrecido por la red."

En una arquitectura P2P se tiene varios módulos idénticos, que corren en diferentes computadoras. Los diferentes módulos almacenados en diferentes sitios se comunican entre ellos para completar los procesos requeridos. En esta arquitectura cada nodo puede acceder a servicios tanto como proveerlos a otros nodos. Estos módulos pueden ser divididos en tres capas:

capa de superposición de base: Se ocupa del problema de descubrir otros participantes en el sistema y crear un mecanismo para que todos los nodos se comuniquen entre sí. Se encarga de que todos los participantes en la red estén al tanto de otros participantes.

capa de middleware: Incluye componentes de software adicional que pueden ser potencialmente rehusados por muchas otras aplicaciones.

capa de aplicación: Provee paquetes de software para ser usados por usuarios y desarrolladores para explotar la naturaleza distribuida de la infraestructura p2p.

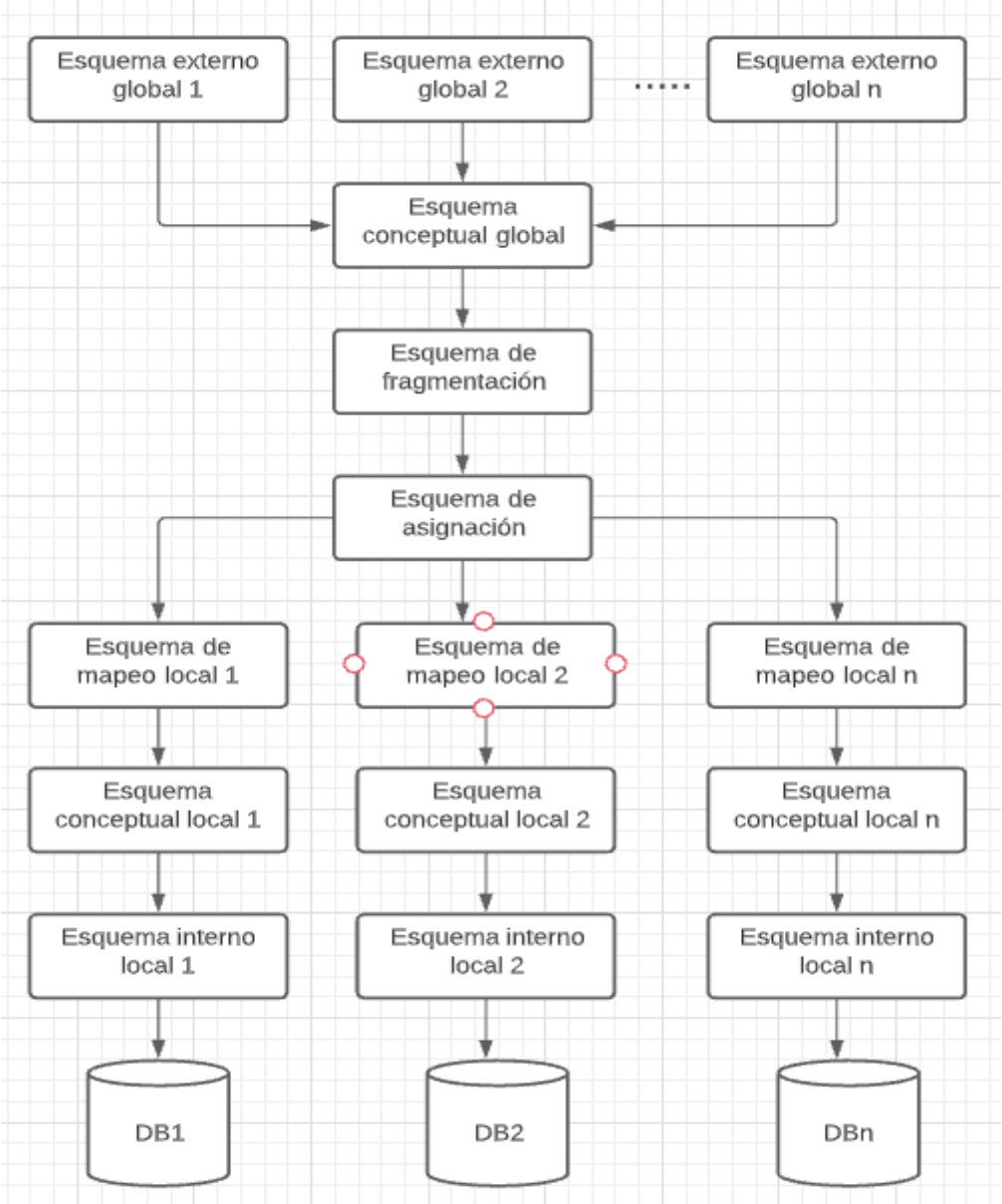
Como en un sistema cliente/servidor, en p2p los datos son vistos como una única base de datos lógica, aunque los datos estén distribuidos en el nivel físico. En este contexto, la identificación de la arquitectura de referencia para un sistema de bases de datos distribuidas es necesario.

Arquitectura de referencia de DBMSs distribuidos.

[1]"Debido a la diversidad de DBMSs distribuidos es complicado presentar una arquitectura común aplicable a para todas las aplicaciones. Los datos en un sistema distribuido están usualmente fragmentados y replicados. Considerando estos casos de fragmentación y replicación, la arquitectura de referencia consiste en los siguientes esquemas:"

- Un conjunto de esquemas globales externos.
- Un esquema conceptual global (GCS)
- Un esquema de fragmentación y un esquema de asignación

- Un conjunto de esquemas por cada DBMS local, conforme la arquitectura ANSI/SPARC.



Arquitectura de referencia de DBMS distribuidos.

Esquema externo global: Este es el nivel más alto en la arquitectura de referencia. Este nivel describe que parte de la base de datos distribuida es relevante para diferentes usuarios.

Esquema conceptual global: El GCS representa la descripción lógica de la base de datos entera como si no estuviera distribuida. Este nivel corresponde al nivel conceptual de la arquitectura ANSI/SPARC y contiende la definición de todas las entidades, sus relaciones y seguridad e integridad de información para toda la base de datos.

Esquema de fragmentación y esquema de asignación: En una base de datos distribuida, los datos pueden estar divididos en fragmentos. El *esquema de fragmentación* describe como los datos se dividirán lógicamente los datos en una base de datos distribuida. El GCS consiste en un conjunto de relaciones globales y el mapeo entre las relaciones globales y los fragmentos está definido en el esquema de fragmentación. El *esquema de asignación* es una descripción de donde los fragmentos deben ser localizados, tomando en cuenta cualquier replicación.

Esquemas locales: Cada DBMS local tiene su propio conjunto de esquemas. Los esquemas local conceptual y local interno corresponden a los niveles equivalentes de ANSI/SPARC. La organización física de los datos es posiblemente diferente en cada máquina, por lo tanto, se requiere una definición interna esquemática en cada sitio, llamado *esquema interno local*. Para encargarse de los problemas de fragmentación y replicación la organización lógica de los datos en cada sitio está descrito como una tercera capa en la arquitectura, llamado *esquema conceptual local,* el GCS es la unión de todos los esquemas conceptuales locales, los esquemas conceptuales locales son un mapeo de del esquema global. Este mapeo está hecho por esquemas locales de mapeo.

Conclusiones

Las diferentes arquitecturas son importantes para darnos un acercamiento a las bases de datos distribuidas, dependiendo el grado en el que queremos que el sistema lo sea. Además, es importante conocer las arquitecturas para el diseño de los esquemas según nuestro enfoque. Además, que son importantes para guiarnos en diseñar los diferentes esquemas para el diseño del sistema.

Referencias

[1] C. Ray, «Distributed DBMS Architecture,» de *Distributed database systems*, NOIDA, Dorling Kindersley, 2009, pp. 87-102.

[2] G. d. T. d. M. d. A. d. B. d. D. (DAFTG), «Modelo de referencia para estandarización de DBMS,» Departamento de Comercio de Estados Unidos, 1985.

[3] R. Schollmeier, «Una definición de redes de P2P para la clasificación de arquitecturas y aplicaciones de P2P,» de *Primera Conferencia Internacional sobre Computación Peer-to-Peer*, Munchen, Alemania., 2002.

[4] M. Tamer Özsu y Patrick Valduriez, Principles of Distributed Database Systems, Springer, 2011.

[5] R. M., «ComputerWeekly,» 14 09 2019. [En línea]. Available: https://www.computerweekly.com/es/definicion/Sistema-de-gestion-de-bases-de-datos-o-DBMS. [Último acceso: 09 03 2022].

[6] E, «Monografias,» 12 03 2021. [En línea]. Available: https://www.monografias.com/trabajos58/procesamiento-datos/procesamiento-datos2. [Último acceso: 09 03 2022].