Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

upiita-ipn

Ingeniería Telemática.

Bases de Datos Distribuidos.

Integrantes : Lisardo René Morgado Resendiz.

Moreno Galicia Jesús Antonio.

López Navarrete Sergio Hidekel

Profesor: De La Cruz Sosa Carlos.

Grupo: 3TM3.

Fecha: Miércoles 09 de Marzo de 2022.

TAREA.

(Reporte de lectura del tema arquitecturas de BDD.)

Conceptos relevantes de la lectura

- *Independencia de datos:*

Es un concepto de bases de datos que nos habla de una forma de gestión en la cuál nuestros datos están separados de otros programas que podrían estarlos usando. Esto a forma de protección de datos.

- *Arquitectura ANSI/SPARC:*

Creada por un grupo de estudio del ANSI (American National Standards Institute) propone que las interfaces sean estandarizadas, y define una arquitectura con 43 interfaces, de las cuales 14 trabajarán directamente con el subsistema físico de almacenamiento de la computadora y por lo cuál, no serán parte esencial de la arquitectura.

Una versión simplificada de esta arquitectura nos dice que debe haber 3 vistas simplificadas de los datos, una vista “externa” para los programadores o usuarios finales, una vista “interna” para la máquina o el sistema y una vista “conceptual” para la empresa, por lo cuál se requiere un esquema definido para cada vista.

En el nivel más bajo de la arquitectura tenemos la vista interna, que trabaja con la definición física y la organización de los datos. Los problemas a resolver en este nivel de la arquitectura son los mecanismos de acceso a los diferentes dispositivos de almacenamiento y la ubicación de los datos en estos.

En la vista externa se gestiona la vista que tendrá un usuario y las relaciones de datos a las que podrá acceder. Una vista puede ser compartida por diferentes usuarios creando así con la colección de estos la vista externa general.

En medio de estas dos definiciones tenemos el esquema conceptual, que es una definición abstracta de la base de datos, la cuál deberá representar los datos y las relaciones entre estos considerando las restricciones físicas de los dispositivos de almacenamiento y las aplicaciones individuales.

Estas tres definiciones se complementan por medio de mapeos en los cuáles se especifica como una definición de un nivel puede ser obtenida mediante una de otro. Esta perspectiva es importante ya que provee las bases para la **independencia de datos**.

- DBMS:

Un DBMS (Data Base Manager System) es un programa compartido con múltiples procesos llamados transacciones el cuál tiene dos dependencias, el **subsistema de comunicación** y el **sistema operativo**.

- Autonomía:

La autonomía, en este contexto, se refiere a la distribución del control, no de los datos. Indica el grado en que los DBMS individuales pueden operar de forma independiente. Autonomía es una función de una serie de factores tales como si los sistemas componentes (es decir, DBMS individuales) intercambian información, si pueden ejecutar transacciones de forma independiente y si se permite modificarlas.

- Autonomía de diseño: La capacidad de que cada Base de Datos decida los aspectos concernientes con su diseño, es decir los desarrolladores son libres de decidir cualquier particularidad e incluso decidir qué DBMS usar.
- Autonomía de comunicación: La habilidad de que una BD pueda comunicarse o no con otro componente de una misma entidad.
- Autonomía de ejecución: Es la habilidad de una BD para ejecutar operaciones locales sin la interferencia de operaciones externas, en la orden que la BD lo decida.
- Autonomía de asociación: Cada BD decide cuándo y cuánto compartir su funcionalidad y recursos con otros componentes, incluso la capacidad de asociarse o retirarse de más federaciones.

- Distribución:

Mientras que la autonomía se refiere a la distribución (o descentralización) del control, la dimensión de distribución de la taxonomía se ocupa de los datos. Por supuesto, estamos considerando la distribución física de datos en múltiples sitios; Como discutimos anteriormente, el usuario ve los datos como un grupo lógico. Hay varias formas en que los DBMS se han repartido. Abstraemos estas alternativas en dos clases: distribución cliente/servidor y distribución de igual a igual (o distribución completa). Junto con los no distribuidos opción, la taxonomía identifica tres arquitecturas alternativas.

- Heterogeneidad:

La heterogeneidad puede ocurrir en varias formas en los sistemas distribuidos, que van desde heterogeneidad de hardware y diferencias en los protocolos de red a variaciones en los datos gerentes Los más importantes desde la perspectiva de este libro se relacionan con los modelos de datos, lenguajes de consulta y protocolos de gestión de transacciones. La representación de datos con diferentes herramientas de modelado crea heterogeneidad debido a la expresividad inherente a las facultades y limitaciones de los modelos de datos individuales.

La *heterogeneidad* se debe a que los nodos que lo componen pueden utilizar diferentes sistemas hardware, protocolos de red, lenguajes de consulta, protocolos de control de transacciones, e incluso diferentes modelos de datos como puede ser el relacional y el orientado a objetos. Un DBMS puede soportar diferentes niveles de heterogeneidad haciendo que todos los problemas que eso conlleva sean transparentes para los usuarios. Muchas veces la heterogeneidad de los sbdd viene definida por los requisitos de las aplicaciones que deben soportar ya que pueden requerir manejar datos con formatos muy diferentes. Sin embargo, en otras muchas ocasiones el nivel de heterogeneidad del sistema es consecuencia de que el

sbdd a construir resulta de la composición de unos sistemas no distribuidos que ya existían previamente y que habían sido desarrollados independientemente.

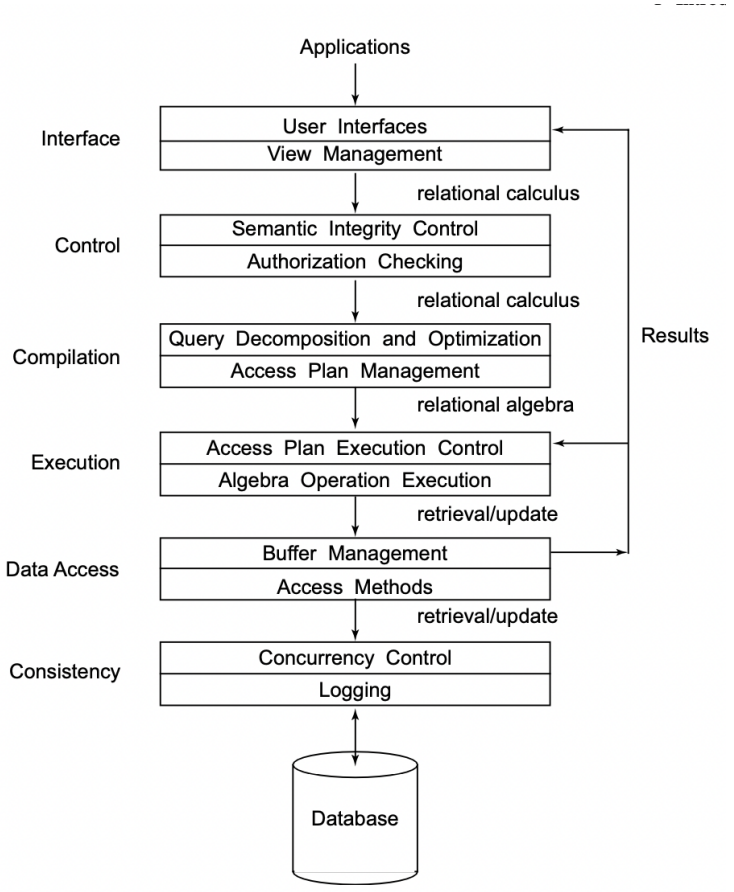
- Función del subsistema de comunicación:

Este permite la comunicación mediante interfaces del DBMS con otros subsistemas para comunicarse con otras aplicaciones.

- Función del sistema operativo:

El sistema operativo nos proporciona una manera de comunicar nuestro DBMS con los recursos del ordenador.

- Capas funcionales de un DBMS Centralizado:



La **capa de la interfaz** maneja lo que es transmitido a las aplicaciones, ya sea mediante lenguajes de programación, consultas realizadas aplicando operaciones de álgebra relacional, etc. Prácticamente consiste en traducir la búsqueda del usuario de datos externos a datos conceptuales.

La **capa de control** controla las búsquedas añadiendo predicados semánticos y de autorización en un lenguaje declarativo. Enriquece las búsquedas en lenguaje de alto nivel hechas **por la capa de interfaz.**

La **capa de compilación o procesamiento de consultas** convierte nuestro código en operaciones de bajo nivel mediante la descomposición de la búsqueda en un árbol de operaciones algebráicas y la optimización de dichas búsquedas. El resultado de esta capa se guarda en un plan de acceso con instrucciones de bajo nivel.

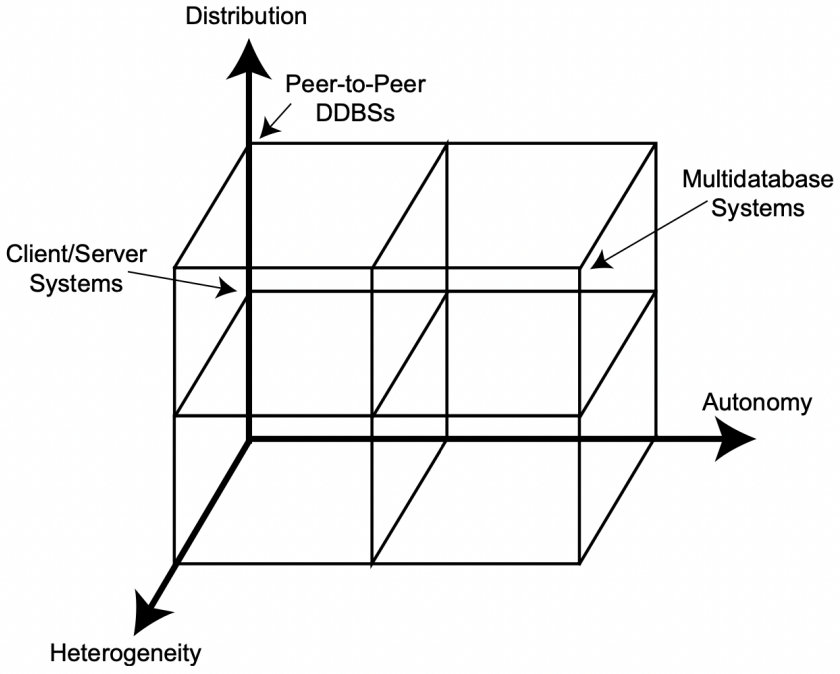
La **capa de ejecución** dirige la ejecución de los planes de acceso generados en la capa anterior.

La **capa de acceso a los datos** maneja las estructuras de datos y el caché que generan las consultas frecuentes.

Por último, **la capa de consistencia** maneja el control de la concurrencia y el registro de solicitudes de actualización.

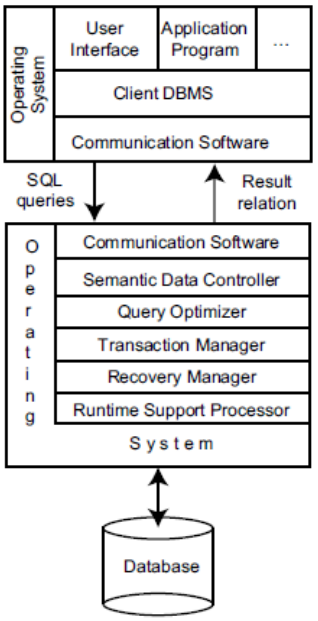
- Modelos de arquitectura para los manejadores de bases de datos distribuídas.

Ahora que sabemos que tenemos diferentes maneras de estructurar la arquitectura de nuestro DBMS, usamos una clasificación que organiza el sistema con respecto de la autonomía de los sistemas locales, su distribución y su heterogeneidad.



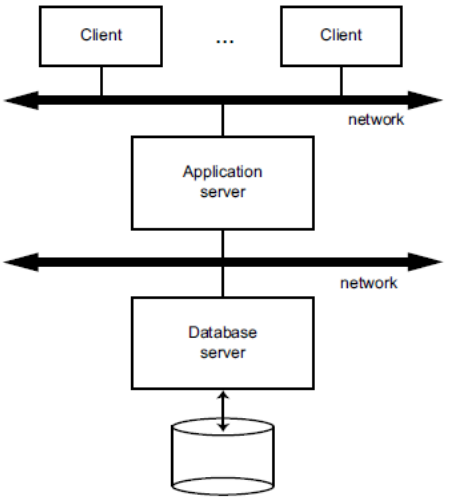
- Sistemas Cliente/Servidor.

Es bastante común en sistemas relacionales donde la comunicación entre los clientes y el (los) servidor(es) está(n) al nivel de sentencias SQL. En otras palabras, el cliente pasa Consultas SQL al servidor sin tratar de entenderlas u optimizarlas. El servidor hace la mayor parte del trabajo y devuelve la relación de resultado al cliente.



Hay varios tipos diferentes de arquitectura cliente/servidor. El más simple es el caso en el que solo hay un servidor al que acceden varios clientes. Llamamos a este cliente múltiple/servidor único. Desde una perspectiva de gestión de datos, esto no es muy diferente de las bases de datos centralizadas ya que la base de datos se almacena en un solo máquina (el servidor) que también alberga el software para administrarlo. Sin embargo, hay algunas diferencias (importantes) con los sistemas centralizados en la forma en que se realizan las transacciones, se ejecutan y se gestionan los cachés. No consideramos tales cuestiones en este punto. A la arquitectura cliente/servidor más sofisticada es aquella en la que hay varios servidores en el sistema (el llamado enfoque de cliente múltiple/servidor múltiple).

El enfoque del servidor de aplicaciones (de hecho, un enfoque distribuido de n niveles) puede ser ampliado por la introducción de múltiples servidores de bases de datos y múltiples aplicaciones servidores, como se puede hacer en las arquitecturas clásicas cliente/servidor. En este caso, lo normal es que cada servidor de aplicaciones esté dedicado a una o varias aplicaciones, mientras que los servidores de bases de datos operan en la forma de servidor múltiple discutida encima.



- Sistemas punto a punto

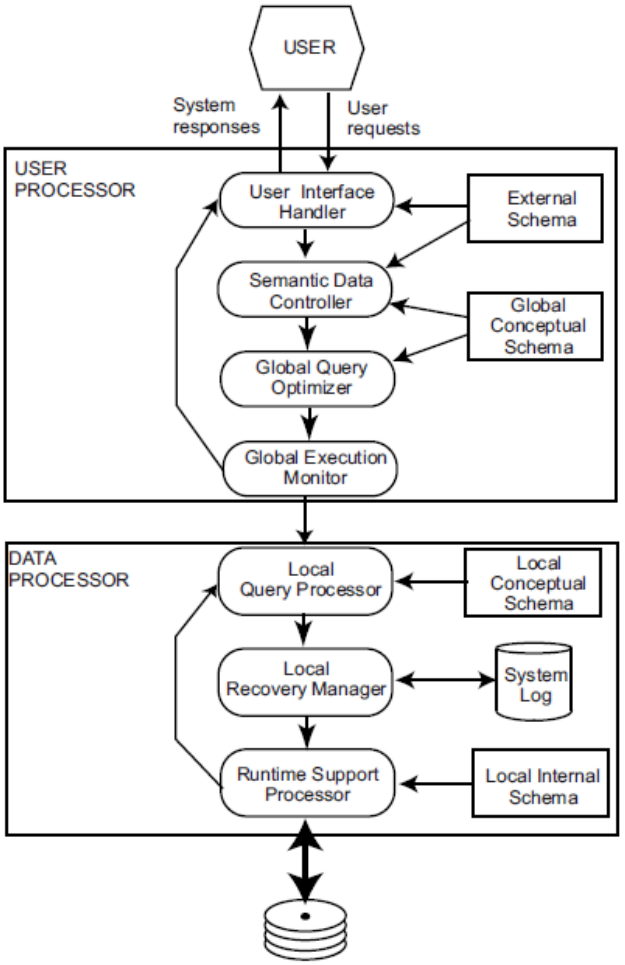
Los sistemas peer-to-peer modernos tienen dos diferencias importantes con sus parientes anteriores. El primero es la masiva distribución en los sistemas actuales. Mientras que en los primeros días nos enfocamos en unos pocos (quizás como máximo decenas de) sitios, los sistemas actuales consideran miles de sitios. El segundo es la heterogeneidad inherente de cada aspecto de los sitios y su autonomía. Mientras esto tiene siempre ha sido una preocupación de las bases de datos distribuidas, como se discutió anteriormente, junto con la distribución masiva, la heterogeneidad del sitio y la autonomía adquieren un significado adicional, rechazando algunos de los enfoques de la consideración.

Los componentes detallados de un DBMS distribuido. Un componente maneja la interacción con los usuarios y otro se ocupa del almacenamiento. El primer componente principal, al que llamamos procesador de usuario, consta de cuatro elementos:

1. El controlador de la interfaz de usuario
2. El controlador de datos semánticos
3. El optimizador y descomponedor de consultas globales
4. El monitor de ejecución distribuida

El segundo componente principal de un DBMS distribuido es el procesador de datos y consta de tres elementos:

1. El optimizador de consultas loca
2. El gerente de recuperación local
3. El procesador de soporte



- Arquitectura del sistema de base de datos múltiple

Los sistemas de bases de datos múltiples (MDBS) representan el caso en el que los DBMS individuales (distribuidos o no) son completamente autónomos y no tienen el concepto de cooperación; que puede ni siquiera “saber” de la existencia del otro o cómo hablar entre ellos. Las diferencias en el nivel de autonomía entre los DBMS múltiples distribuidos y los DBMS distribuidos también se reflejan en sus modelos arquitectónicos. La diferencia fundamental se relaciona con la definición del esquema conceptual global. En el caso de los DBMS distribuidos lógicamente integrados, el esquema conceptual global define la vista conceptual de toda la base de datos, mientras que en el caso de los DBMS múltiples distribuidos, representa solo la colección de algunas de las bases de datos locales que cada DBMS local desea compartir.

Conclusiones de la lectura

Hoy en día nos resulta tan común acceder a información almacenada en una base de datos que no dimensionamos la cantidad de movimientos y arquitecturas que tuvieron que ser creadas para hacer esto posible, una de ellas es la arquitectura ANSI/SPARC, la cuál nos ayuda a mantener la independencia de nuestros datos para brindar un nivel extra de protección mediante una serie de procesos en cascada que ayudan a concretar una consulta.

La implementación de las bases de datos distribuidas hoy en día son fundamentales para todas las organizaciones que se especializan en el desarrollo de sistemas de alta calidad, como podemos analizar en base a la lectura un DBMS tiene la capacidad de ser autónomo lo cual nos permite la seguridad y la integridad de los datos. Cabe mencionar que lo fundamental para un buen desarrollo es la interacción y la arquitectura entre el cliente-servidor.

Las bases de datos distribuidas son cada vez más usadas por las empresas y suponen una ventaja competitiva frente a los sistemas centralizados, siempre y cuando la empresa en cuestión tenga necesidad de usar una base de datos de este tipo.

Bibliografía

[1] M. T. Özsu y P. Valduriez, Principles of Distributed Database Systems. Springer, 2019.

[2] Bases de datos avanzadas. Castelló de la Plana, Spain: Universitat Jaume I, 2013.

[3] INSTITUTO TECNOLÓGICO Y. DE ESTUDIOS SUPERIORES DE MONTERREY. "DISTRIBUCIÓN DE DATOS PARA BASES DE DATOS DISTRIBUIDAS, UNA ARQUITECTURA BASADA EN COMPONENTES DE SOFTWARE". repositorio.tec.mx/bitstream/handle/11285/568576/DocsTec_1100.pdf?sequence=1&isAllowed=y. http://repositorio.tec.mx/bitstream/handle/11285/568576/DocsTec_1100.pdf?sequence=1&isAllowed=y (accedido el 7 de marzo de 2022).