Bases de datos distribuidas

Es momento de discutir acerca de un tipo especial de implementación de bases de datos, en el cual, aunque se conservan las mismas estructuras lógicas de almacenamiento, los datos físicamente no residen en un solo lugar geográficamente hablando.

Es un tipo de implementación que resulta de mucha utilidad sobre todo para grandes corporaciones con presencia en muchos sitios, que manejan grandes volúmenes de datos y altos niveles de concurrencia. Analizaremos la definición de este tipo de implementaciones, las técnicas de almacenamiento distribuido, lo que implica acceder a una base de datos distribuida desde la perspectiva del usuario y/o aplicación, las limitaciones y retos para la gestión de transacciones en estos casos y por último veremos cómo actualmente se está aprovechando este tipo de implementaciones para ofrecer servicios de almacenamiento de datos en la nube.

Luego de analizar y discutir los temas programados, usted estará en capacidad de describir los usos y beneficios y escenarios donde es útil implementar sistemas de bases de datos distribuidas. También podrá proponer un esquema de almacenamiento distribuido para un escenario específico.

1. Panorámica de un Sistema de Bases de Datos Distribuidas

Primero es importante que usted tenga claro la diferencia entre las diversas arquitecturas de los sistemas de base de datos, las cuales denotan distintas maneras de acceder, procesar y almacenar los datos. Son 4 arquitecturas, que de alguna manera han marcado la evolución de los sistemas de bases de datos:

- **Arquitectura centralizada**: es aquella donde en un solo servidor funciona tanto la base de datos como las aplicaciones, no interactúan con servidores externos. Todo el procesamiento relacionado a los datos se realiza en ese servidor central.
- **Arquitectura cliente-servidor**: en este caso ciertas tareas relativas al procesamiento de los datos se realizan en equipos diferentes al servidor de base de datos. En lo que respecta a la base de datos en sí, sigue siendo centralizada, pues existe un solo servidor donde opera el SGBD, pero las aplicaciones, pueden correr directamente en los equipos de usuario final (arquitectura de 2 capas) o en servidores de aplicaciones (arquitectura de 3 capas).
- **Arquitectura paralela**: el multiprocesamiento ha permitido que actualmente se cuenten con implementaciones hardware con disponibilidad de muchos procesadores, lo que implica que un servidor sea capaz de ejecutar varios procesos paralelamente. Ello permite que un SGBD sea capaz de atender y despachar más peticiones de acceso a los datos en un menor tiempo. Aunque desde el punto de vista lógico la base de datos seguiría centralizada en una sola ubicación.
- **Arquitectura distribuida**: Los datos en este caso físicamente se encuentran diseminados en distintos servidores dentro de una misma sede o en lugares geográficamente distantes, pero integrados en un mismo sistema de gestión de bases de datos.

Nuestro estudio se centrará justamente en los sistemas de base de datos de arquitectura distribuida.

Connolly y Begg (2005) definen a una base de datos distribuida como "Una colección lógicamente interrelacionada de datos compartidos (junto con una descripción de estos datos) físicamente distribuidos por una red informática", y a un *Sistema de Gestión de Bases de Datos Distribuidas* (**SGBDD**) lo definen como "el sistema software que permite gestionar la base de datos distribuida y hace que dicha distribución sea transparente para los usuarios"(p.626).

Complementariamente a lo explicado, otra de las razones para implementar una base de datos distribuida es la escalabilidad, ya que, en un entorno distribuido es mucho más fácil manejar la expansión, se pueden añadir nuevas localizaciones (nodos) a la red sin afectar las operaciones de las otras.

La Figura 1, ilustra la topología de un SGBDD. Donde cada nodo es un servidor físicamente alojado en una determinada ubicación, que tiene un SGBD corriendo localmente, y opcionalmente un conjunto de datos almacenados localmente (por ejemplo, en el nodo 3 estaría corriendo el SGBD, pero sin datos en su almacenamiento secundario). La base de datos global la componen la suma de las bases de datos locales.

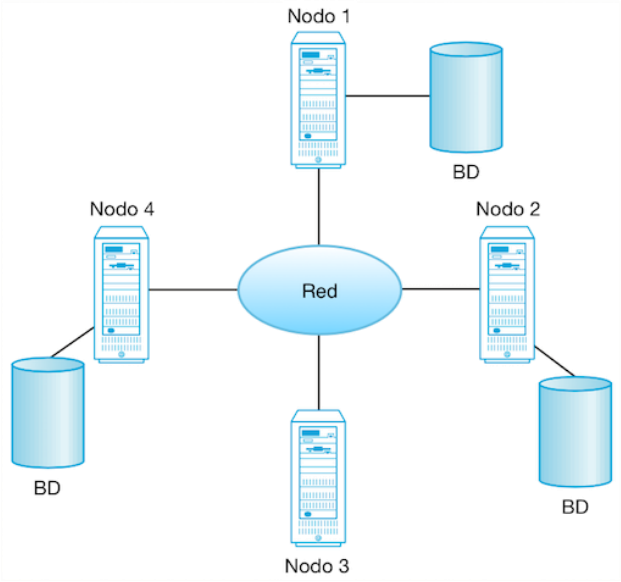


Figura 1. Sistema de gestión de bases de datos distribuidas
Fuente: adaptado de Connolly y Begg, 2005, p.627
Elaborado: Encalada, E.

Además, no necesariamente en todos los nodos debe estar instalado el mismo motor, aunque es deseable para que se pueda garantizar en mayor grado la consistencia de los datos.

Todas estas consideraciones, Elmasri y Navathe (2016) la tienen en cuenta al plantear las condiciones mínimas que debe cumplir una base de datos distribuida para ser considerada como tal:

- Conexión de nodos de bases de datos a través de una red informática. Hay múltiples ordenadores, llamados sitios o nodos. Estos sitios deben estar conectados por una red subyacente para transmitir datos y comandos entre los sitios.
- Interrelación lógica de las bases de datos conectadas. Es esencial que la información en los distintos nodos de la base de datos esté lógicamente relacionada.
- Posible ausencia de homogeneidad entre los nodos conectados. No es necesario que todos los nodos sean idénticos en términos de datos, hardware y software (p.842).

Desde el punto de vista de las aplicaciones de usuario final, al margen de cómo se distribuyan y almacenen físicamente los datos, un SGBDD se comporta de la misma forma que uno centralizado, es lo que se denomina **transparencia**.

	**Actividad propuesta:** Consulte cuales son las ventajas y desventajas de un sistema de base de datos distribuida y base a ello identifique en que escenarios sería propicio este tipo de implementaciones.

2. Almacenamiento distribuido de datos

Tenga en cuenta que en una implementación distribuida las estructuras lógicas de almacenamiento no cambian ni desaparecen. En bases de datos relacionales, seguimos manejando las conocidas tablas o relaciones; solo que en este caso su contenido (los datos) estaría distribuido en distintos servidores.

Son dos lo métodos de distribución: replicación y fragmentación.

La forma más sencilla de distribuir una relación es mediante *replicación*, es decir mantener una copia íntegra de la tabla en todos los nodos del sistema distribuido, de manera que cuando una aplicación requiera acceder a los datos de esa tabla lo hará siempre de forma local. Esta técnica implica mayor disponibilidad y mejor rendimiento en consultas; pero vuelve más complejas las operaciones de actualización, dado que la actualización de un dato deberá realizarse en todos los nodos del sistema distribuido, y en consecuencia se incrementan los tiempos de respuesta y el nivel de concurrencia en el procesamiento de transacciones. Por consiguiente, solo se deberían replicar aquellas tablas que no son actualizadas con frecuencia y su contenido en su totalidad es susceptible de ser accedido desde todos los nodos; por ejemplo, tablas de catálogos (países, tipos de cuenta, categorías de empleados, marcas de productos, etc.).

La *replicación completa* (aquella de toda la base de datos) merece especial atención, ya que en algunos casos si suele ser muy útil para asegurar la disponibilidad de la base de datos. Por ejemplo, una empresa en la que realmente no existan nodos geográficamente distantes, sino simplemente un esquema de dos servidores alternativos, replicados y ubicados en el mismo sitio, con el propósito de realizar un balanceo de carga, y que a la vez constituya un esquema de backup, de modo que, si falla uno de los servidores, el sistema seguirá funcionando con el otro sin necesidad de detener servicios.

La *fragmentación* en cambio implica particionar la tabla de manera que en cada nodo exista una porción de los datos. Lo más común es la *fragmentación horizontal*, en la cual se divide la tabla en subconjuntos de tuplas, y cada subconjunto se almacena en un nodo distinto de acuerdo con el criterio de distribución que se haya elegido. En este punto, es importante acotar que hay un tipo especial de fragmentación horizontal llamada ***fragmentación horizontal derivada***, que es cuando se divide las tuplas de una tabla en función de la fragmentación de otra tabla.

Tenga presente que el hecho de que en un nodo solo exista una parte de una tabla, no significa que quien se conecta a ese nodo solo puede acceder a esa porción de datos; puede acceder a todos los datos de la tabla, si la tupla que busca no está en ese nodo, el SGBDD accederá al nodo donde esté y la obtendrá, la diferencia en ese caso es que la respuesta será un poco más demorada.

También tenga en cuenta que se pueden combinar ambas técnicas, unas tablas se replican y otras se fragmentan, que de hecho es lo más idóneo. Se debería fragmentar tablas en las cuales desde cada nodo se accede frecuentemente a una porción de sus filas. Los metadatos sobre la distribución de los datos en los diferentes nodos (réplicas y fragmentos) se guarda en un catálogo global que lo gestiona el SGBDD.

Veamos ahora un ejemplo que complementa lo explicado. Para el caso de estudio PEDIDOS descrito en el Anexo, suponga que la empresa implementa una base de datos distribuida de manera que se instala un nodo (un servidor local) en cada ciudad donde opera. En este caso el criterio de distribución es la CIUDAD y a partir de allí se planifica la distribución.

Para poder fragmentar en este caso, la tabla debería contener la columna que corresponda al criterio de distribución (fragmentación horizontal directa) o estar asociada por llave foránea (FK) a una tabla ya fragmentada con base en ese criterio. En nuestro caso OFICINAS es la tabla que contiene el atributo

CIUDAD por lo tanto se haría una fragmentación horizontal. EMPLEADOS en cambio no tiene el atributo CIUDAD, sin embargo, está relacionada a OFICINAS (cada empleado pertenece a una oficina), por lo que se haría una fragmentación horizontal derivada, y así sucesivamente. A continuación, se muestra la especificación de cada fragmento para esas dos tablas:

$$\texttt{oficinas\_UIO} = \sigma_{\text{ciudad = 'UIO'}}(\text{oficinas})$$

$$\texttt{oficinas\_GYE} = \sigma_{\text{ciudad = 'GYE'}}(\text{oficinas})$$

$$\texttt{empleados\_UIO} = (\rho_{e}\ \text{empleados}) \bowtie_{\text{e.idoficina = o.idoficina}} (\rho_{o}\ \text{oficinas\_UIO})$$

$$\texttt{empleados\_GYE} = (\rho_{e}\ \text{empleados}) \bowtie_{\text{e.idoficina = o.idoficina}} (\rho_{o}\ \text{oficinas\_GYE})$$

Cada tabla fragmentada debería tener tantos fragmentos como valores distintos tenga el atributo con base en el cual se realiza la distribución. En nuestro caso estamos asumiendo que solo son dos las ciudades donde opera la empresa (Quito y Guayaquil).

A continuación, se muestra el tipo de distribución que correspondería realizar a cada tabla:

Tabla	**Tipo de distribución**	**Motivación**
OFICINAS	Fragmentación horizontal	Contiene el atributo CIUDAD, con base en el cual se realiza la distribución.
EMPLEADOS	Fragmentación horizontal derivada	Derivada a partir de la fragmentación de oficinas.
PEDIDOS	Fragmentación horizontal derivada	Derivada a partir de la fragmentación de empleados.
ITEMS	Fragmentación horizontal derivada	Derivada a partir de la fragmentación de pedidos.
CLIENTES	Fragmentación horizontal derivada	Derivada a partir de la fragmentación de empleados. Se entiende que los clientes normalmente están asociados a ciudad del vendedor asignado.
PRODUCTOS	Replicación	Se asume que en todas las ciudades se oferta la misma lista de productos.
FABRICANTES	Replicación	Si productos se replica entonces fabricantes también ya que están asociados. Además, porque se trata de un catálogo.

Aquí hay que tomar en cuenta que, si estamos hablando de que hay sucursales en cada ciudad que tienen su propia bodega y stock de productos, haría falta en el modelo de datos PEDIDOS asociar las sucursales con los productos para poder llevar el inventario en cada localidad.

Es necesario aclarar por último que, no se debe confundir FRAGMENTOS con VISTAS. Un fragmento no es una vista de datos. Como sabemos en bases de datos las vistas no son estructuras de almacenamiento, son relaciones virtuales que ejecutan la consulta SQL subyacente para obtener una porción de los datos. Los fragmentos en cambio son estructuras de almacenamiento que corresponden a partes de una tabla (subconjuntos de tuplas y/o columnas de una relación).

Actividades propuestas:

1. Suponga las siguientes tablas de una base de datos de una empresa de servicios:

```
clientes (cedula, apellidos, nombres, telefono, direccion, sucursal,
tipo_cliente, id_nacionalidad)

sucursales (sucursal, nombre, provincia, ciudad, direccion, telefono)

tipos_cliente (tipo_cliente, descripcion, descuento)

nacionalidades (id_nacionalidad, nombre)

vehiculos (id_vehiculo, placa, chasis, marca, modelo, año, color, sucursal)
```

Decidimos implementar un sistema SGBD distribuido, para lo cual establecemos que se hará una distribución geográfica de los datos, basados en la provincia. Es decir, tendremos un nodo en cada provincia donde opere la empresa. Con base en lo anterior analice y diga: en este caso ¿se debería aplicar fragmentación, replicación, ambas? ¿Cuáles tablas se deberían fragmentar y cuáles replicar?

2. Plantee un ejemplo de organización o negocio cuya naturaleza de operación sería idónea para implementar un sistema de base de datos distribuida. Indique las razones.

3. Transparencia en un SGBDD

La transparencia se refiere a que, desde el punto de vista del usuario final, a nivel lógico un sistema distribuido deberá ser idéntico a un sistema no distribuido. Es decir, los usuarios de un sistema distribuido deberán comportarse exactamente como si fuera un sistema centralizado.

1. Autonomía local Los nodos o localidades deben ser independientes entre si en el mayor grado posible	**2. No es necesario un sitio central.** Todos los sitios/nodos deben ser tratados como iguales	**3. Operación continua** Un SGBDD no debería estar nunca fuera de servicio
4 Independencia de localización Los usuarios y las aplicaciones no necesitan conocer la ubicación física de los datos.	**5. Independencia de fragmentación de datos** Los usuarios pueden comportarse como si los datos no estuvieran fragmentados	**6. Independencia de la replicación de datos** El usuario debe comportarse como si los datos no estuvieran replicados
7. Procesamiento de consultas distribuidas La performance de una consulta debe ser independiente del sitio donde se realiza la consulta	**8. Gestión de transacciones distribuidas** Debe disponer de mecanismos para el control de concurrencia y la recuperación de transacciones distribuidas	**9. Independencia de hardware** Es necesario tener la posibilidad de ejecutar el mismo SGBD en diferentes plataformas de Hardware.
10. Independencia del Sistema Operativo Es necesario tener la posibilidad de ejecutar el mismo SGBD en sitios con diferentes sistemas operativos	**11. Independencia de red** El SD debe poder operar con diferentes redes de comunicaciones	**12. Independencia del SGBD** Debe permitirse la heterogeneidad, es decir, que cada sitio pueda funcionar con un SGBD diferente.

Figura 2. Las doce reglas de Date para un SGBDD
Fuente: Connolly y Begg, 2005, p.661
Elaborado: Encalada, E.

En el ejemplo que se propuso en el apartado anterior, sobre la fragmentación de OFICINAS y EMPLEADOS, el hecho que existan unos fragmentos llamados "oficinas_UIO", "oficinas_GYE", "empleados_UIO", "empleados_GYE", no significa que a la vista del usuario final aparezcan tablas con esos nombres, para el usuario siguen existiendo las tablas OFICINAS y EMPLEADOS, y nada más. Quiere decir, que aquella complejidad de la distribución es interna dentro del SGBDD y no es visible para los usuarios o aplicaciones que acceden a manipular los datos. Dicho de otro modo, en términos de SQL, la lógica de las operaciones SELECT, INSERT, UPDATE y DELETE no deberá sufrir cambios.

Revise ahora lo que se expone en la Figura 2, corresponde a las 12 reglas propuestas por Christopher J. Date[1], que debe cumplir todo SGBD distribuido (SGBDD).

4. Transacciones distribuidas

Recuerde la topología de un sistema de base de datos distribuida (presentada en la Figura 1). Tenemos varios nodos cada uno corriendo una instancia del SGBD que se haya elegido y normalmente una base de datos local con una porción de datos de la base de datos global. Y además existe un SGBD distribuido (SGBDD) que actúa como coordinador y orquestador de todos nodos. Una aplicación o usuario accede a uno de los nodos (al SGBD local), el cual se encarga procesar sus peticiones. Si los datos requeridos para atender la petición están todos en el nodo local, no requerirá acceder a otros nodos, pero si la petición implica conexión a otros servidores del sistema distribuido, el SGBDD se encargará de coordinarlo. En ese contexto, ya se podrá imaginar la complejidad que implica el procesar una transacción en bases de datos distribuidas, y el garantizar que se cumplan las 4 propiedades ACID ya estudiadas, de manera que la integridad de los datos no sea afectada.

Si una transacción de actualización de datos involucra en su ejecución únicamente la base de datos local del nodo al que se conecta, el procesamiento en ese caso será igual al de una transacción normal como las que estudiamos en la unidad sobre transacciones. El problema surge cuando esa transacción involucra a varios nodos (porque la tabla actualizar esta replicada en varios nodos, o porque el registro involucrado está en otro nodo); en ese caso dicha transacción se convierte en una transacción distribuida; implica que será necesario ejecutar varias transacciones locales en nodos distintos, y el control en este caso lo asumirá el SGBDD, cuyo trabajo será orquestar la ejecución de la transacciones locales de manera que se asegure que previo a confirmar la transacción global, se hayan comprometido exitosamente las transacciones locales involucradas (compromiso en dos fases).

Si a lo anterior sumamos la concurrencia en los nodos involucrados, posibles fallos o demoras en la comunicación entre servidores, interbloqueos, etc., la complejidad de procesar la transacción se multiplica. Al final el SGBDD deberá asegurar las propiedades ACID, pero es importante que el DBA se asegure de diseñar una distribución de datos óptima de manera que se reduzca la probabilidad de transacciones distribuidas.

5. Bases de datos en la nube

Con el surgimiento de *cloud computing*, aparecieron muchos servicios disponibles para que personas y organizaciones aprovechen las mega infraestructuras de hardware que poseen corporaciones como Google, Amazon, Microsoft y muchos otros, para albergar en ellas sus aplicaciones y sus datos, quitándose con ello el peso de tener que instalar y mantener una infraestructura local, que se deprecia con los años, que tiene una vida útil limitada y que siempre está sujeta a fallos.

[1] Christopher J. Date ha sido uno de los más prominentes investigadores del modelo relacional de bases de datos.

Uno de los servicios ya disponibles actualmente en la nube, es el de almacenamiento de datos. De manera que las empresas puedan capturar y almacenar datos a través de aplicaciones web, con rapidez, con alta disponibilidad y sin preocuparse por el aumento de la demanda en el uso de sus plataformas.

A continuación, algunas reflexiones adicionales:

a) Este tipo de implementaciones se centran sobre todo en la captura de datos.
b) Se aprovechan los conceptos del procesamiento distribuido y de los sistemas de bases de datos distribuidas para implementar este tipo de servicios.
c) Al igual que en bases de datos distribuidas tradicionales, las bases de datos en la nube deben garantizar la transparencia. Es decir, el usuario no necesita conocer donde se ubican los datos físicamente y como están distribuidos.
d) La principal ventaja radica en que el proveedor garantiza una alta disponibilidad de la base de datos (99.9% en la mayoría de casos), lo que para una empresa se traduce en no tener que preocuparse por el mantenimiento de una infraestructura local. El tener una infraestructura local que garantice esos niveles de disponibilidad, resulta bastante costoso.
e) También es importante la posibilidad que dan estos servicios para que la inversión esté orientada al uso. Es decir, la empresa solo paga los recursos que necesita de acuerdo a la demanda y volumen de datos a almacenar, si estos crecen, podrá ir incrementando recursos de acuerdo a sus necesidades.
f) Una desventaja radica en el tema de la confidencialidad de los datos. El que cierta información crítica y reservada pueda reposar físicamente en servidores de otra empresa puede resultar un problema para ciertas organizaciones.
g) Otro problema puede radicar en tratar de integrar esa base de datos en la nube, con otras que tengamos localmente, el establecer sesiones remotas desde máquinas locales puede requerir conexiones a internet de alta velocidad. Lo que es un costo adicional para la empresa.

Actividad propuesta:

Investigue a través de internet al menos 5 de los más populares servicios de almacenamiento de datos en la nube (bases de datos en la nube). Por cada uno identifique sus características tales como: proveedor, motor de base de datos que usan, tipo de base de datos, capacidades de almacenamiento, precios, opciones de escalamiento, etc. Elabore un mapa conceptual con la caracterización de cada producto.

Como usted ha podido evidenciar los SGBDD representan una interesante alternativa de aprovechamiento del procesamiento distribuido para lograr contar con un esquema de almacenamiento de datos donde la carga transaccional no esté concentrada en un solo servidor central. Sobre todo, útil para empresas que operan en lugares geográficamente dispersos y con grandes volúmenes de datos. Muchas corporaciones han aprovechado este tipo de implementaciones para ofrecer a su vez servicios de almacenamiento de datos a gran escala. La principal complejidad que se ve evidente en este tipo de implementaciones es el procesamiento de transacciones distribuidas y el aseguramiento de la integridad de los datos.

Como ve, es posible organizar los datos de una forma descentralizada, con el fin garantizar mejor la disponibilidad de los datos y mejorar el desempeño de las aplicaciones de gestión. Espero que lo estudiado le haya ayudado a valorar las ventajas de un sistema distribuido y también a ser consciente de sus limitaciones.

Referencias bibliográficas

Connolly, T. y Begg, C. (2005). *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión*. Madrid: Pearson Education

Anexo

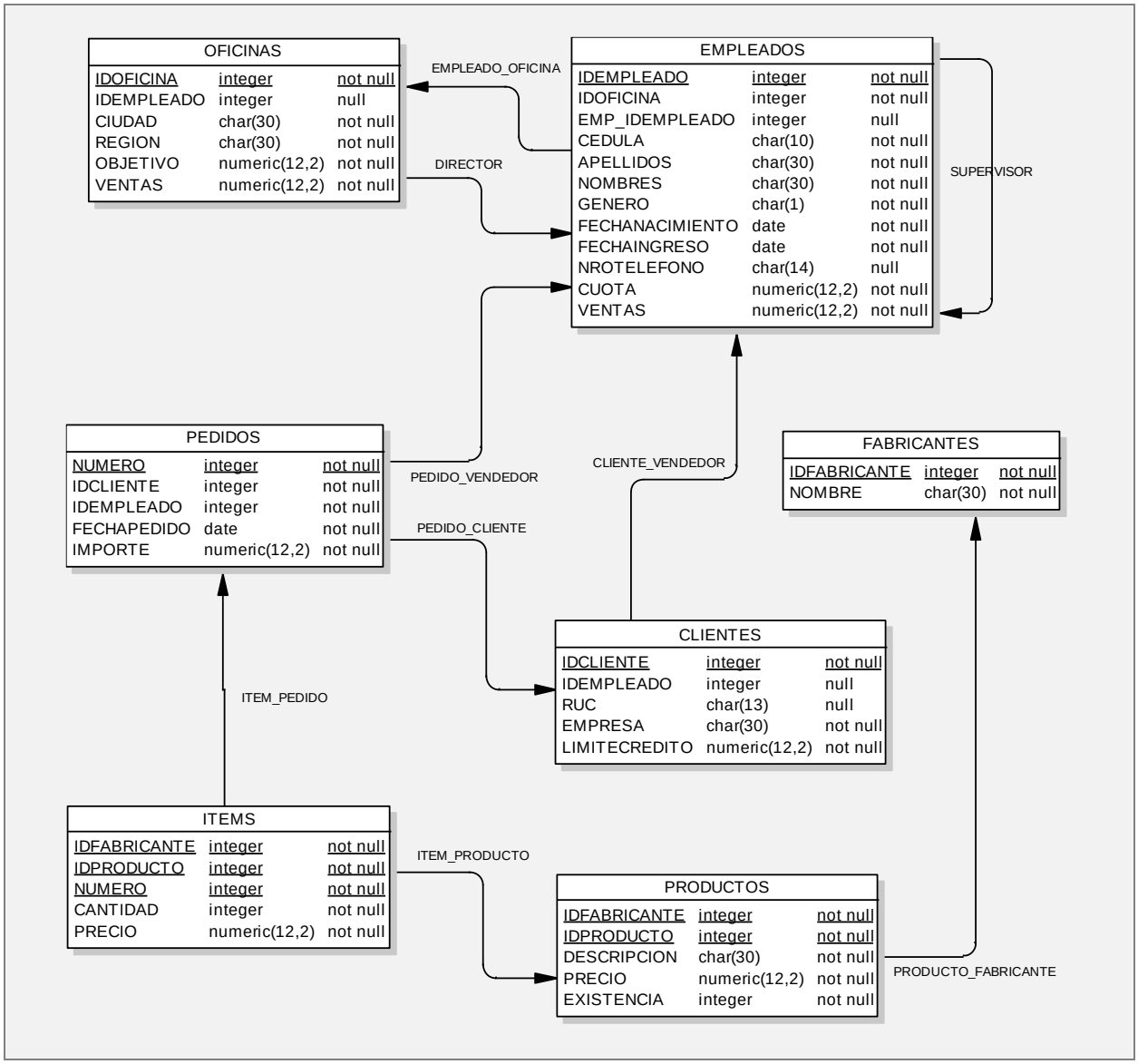
CASO DE ESTUDIO PEDIDOS

Descripción del problema

Imaginémonos en una empresa de distribución que desea informatizar el procesamiento de pedidos. Las consideraciones básicas para tomar en cuenta en un principio son las siguientes:

- La empresa tiene varias oficinas o localidades donde recepta los pedidos.
- Cada oficina cuenta con un equipo de vendedores y empleados quienes se encargan de procesar los pedidos de cada cliente.
- A su vez cada vendedor tiene un supervisor, el cual a su vez tiene bajo su control a varios vendedores.
- Un cliente puede solicitar uno o más productos en un solo pedido.
- Cada producto se codifica basándose en el fabricante y a un número interno.

Esquema relacional



Restricciones de integridad para campos derivados

- La suma del importe de los pedidos receptados por un vendedor debe corresponder a valor de ventas para ese vendedor (empleados.ventas).
- La suma de las ventas de los vendedores de una misma oficina debe corresponder al total de ventas de esa oficina (oficinas.ventas).
- El importe total de cada pedido (pedidos.importe) es igual a la suma del importe de cada ítem correspondiente a ese pedido.