ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

Laboratorio de:

Bases de datos distribuidas

Práctica No.:

3

Grupo No.:

9

Integrantes:

- Fernando Eliceo Huilca Villagómez
- Gregory Leandro Salazar Domínguez
- Mateo Nicolás Simbaña Guarnizo

Tema:

Diseño de la base de datos distribuida para una red de gimnasios

Objetivos:

1. Definir los requerimientos y el grafo relacional del escenario de una empresa con dos sucursales de gimnasios, con el fin de aplicar la metodología descendente para el diseño de una base de datos distribuida.

2. Identificar las sedes y roles que forman parte de la empresa mediante la descripción del escenario, con la finalidad de comprender la estructura y organización que debe tener el sistema distribuido.

3. Diseñar el esquema de fragmentación, replicación y ubicación a partir del esquema lógico global de la base de datos para establecer la distribución de los datos entre los distintos nodos de la red.

Marco teórico:

En el **diseño de Bases de Datos Distribuidas** (BDD), una de las metodologías que se puede usar es la **descendente**. Consiste en tener como punto de partida un esquema lógico global (GCS) de la Base de Datos (BD) que se quiere distribuir. Ese esquema se usa para construir los diferentes esquemas de fragmentación, asignación y replicación que definen cómo se distribuyen los datos por medio de los nodos de la red [1].

1. Esquema de fragmentación

Consiste en la división y creación de tablas más pequeñas. El criterio de la división se lo realiza en base a la manera en que se usan los datos por parte de las diferentes aplicaciones que acceden a la BD [1].

Este esquema se **compone** de las diferentes tablas en las que se ha dividido una tabla del GCS y de la condición empleada para esa división. Todo esto es **expresado** en álgebra relacional [1].

Los **motivos para fragmentar** son [1]:

1. Por su **utilidad**, debido a que las aplicaciones de BDD suelen operar con vistas, por lo que se pueden tener diversas tablas en diferentes nodos para generar la unidad distribuida.
2. Ofrece mayor **eficiencia**, por poder almacenar los datos en el nodo donde se usan frecuentemente.
3. Permite **incrementar la concurrencia**, debido a que fragmentar tablas consigue que una transacción se divida en subquerys que operarán sobre esos fragmentos.
4. Ofrece mayor **seguridad**, por no alojar en un nodo local datos que no se requieren, es decir, no estará disponible información para personas no autorizadas que no la requieran.

Los **motivos para no fragmentar** son [1]:

1. El uso de varios fragmentos colocados en diferentes nodos para construir una vista suele afectar el **rendimiento** del Sistema Gestor de Bases de Datos Distribuidas (SGBDD).
2. El manejo de la **semántica** de la información se complica por la existencia de la fragmentación.

En base a lo mencionado anteriormente, existen **3 normas** para asegurar la consistencia de la semántica de la BD al fragmentarla, evaluando la calidad de la fragmentación de una tabla [1]:

1. **Completitud:** Cada uno de los datos de una tabla fragmentada deben encontrarse en alguno de los fragmentos.
2. **Disyunción:** Los datos que existen en un fragmento no deben estar en otro. La única excepción es en la fragmentación vertical, donde la clave primaria sí debe estar presente en todos los fragmentos.
3. **Reconstrucción:** Siempre se debe lograr generar la BD global a partir de los fragmentos.

Las 3 formas de fragmentar una tabla son fragmentación vertical, horizontal y mixta [1].

1.1 Fragmentación vertical

Divide una tabla (o relación) R en grupos de columnas que son los atributos de la tabla original. Su sintaxis es la siguiente (se usa el operador algebraico de proyección):

$$\boldsymbol{R_i = \Pi_{Li}(R)}$$

Donde $\boldsymbol{i = 1...n}$ y $\boldsymbol{R_i}$ es el grupo de fragmentos en que se divide la tabla original $\boldsymbol{R}$ [1].

$\boldsymbol{L_i}$ es la condición por la que se realiza la fragmentación, específicamente en este caso, $\boldsymbol{L_i}$ serán todos los atributos de la tabla original que se agruparán en un fragmento (tomando en cuenta que siempre se debe colocar la clave primaria) [1].

Además, se debe cumplir que la unión de todos los $\boldsymbol{L_i}$ resulte en todas las columnas de la tabla original y que la intersección de todos los $\boldsymbol{L_i}$ sea igual a la clave primaria de la tabla original. Aquí se evidencia más la necesidad de siempre especificar la clave primaria, para así poder reconstruir la tabla original ($\boldsymbol{R}$) a partir de los fragmentos ($\boldsymbol{R_i}$) [1].

1.2 Fragmentación horizontal

Divide una tabla en subgrupos de registros (filas o tuplas), teniendo cada subgrupo un significado lógico. Este tipo de fragmentación presenta 2 subtipos:

1.2.1 Fragmentación Horizontal Primaria

Se realiza usando predicados (condición que escoge un subconjunto de filas de una tabla), que son definidos en la misma tabla por usar sus propias columnas. Su notación es [1]:

$$\boldsymbol{R_i = \sigma_{Pi}(R)}$$

Donde $\boldsymbol{P_i}$ es el predicado en base a uno o muchos atributos de la tabla original ($\boldsymbol{R}$), funcionando como la condición aplicada para escoger el contenido de los fragmentos [1].

1.2.2 Fragmentación Horizontal Derivada

Se realiza como resultado de los predicados definidos sobre atributos que se encuentran en otras tablas o fragmentos. Esto quiere decir que, la tabla a fragmentar ($\boldsymbol{R}$) depende de otra tabla ($\boldsymbol{Q}$) porque sobre sus atributos está definido el predicado para fragmentar [1].

La notación es la siguiente (usando el operador relaciona de semicombinación, que funciona de la misma manera que el operador de combinación, pero la tabla resultante solo tendrá las columnas del primer operando) [1]:

$$\boldsymbol{R_i = R \ltimes Q_i}$$

Donde $\boldsymbol{Q_i}$ hace referencia al grupo de fragmentos en los que se dividió la tabla $\boldsymbol{Q}$. La semicombinación es realizada por el atributo (o atributos) que relacionan a las 2 tablas [1].

Para verificar que la fragmentación fue realizada de forma adecuada, se debe cumplir que la unión de todos los $\boldsymbol{R_i}$ forme la tabla original $\boldsymbol{R}$ mientras que la intersección debe obtener un resultado vacío [1].

1.3 Fragmentación mixta

Surge de la posible insuficiencia del uso de una primera fragmentación vertical u horizontal para cubrir los requerimientos de las aplicaciones de usuario. Por lo tanto, se vuelve necesario volver a realizar una fragmentación sobre los fragmentos ya obtenidos. Adicionalmente, este tipo de fragmentación presenta también subtipos, los cuales son:

1.3.1 Fragmentación VH

Primero se realiza una fragmentación vertical, luego, por cada uno de los fragmentos verticales se realiza una fragmentación horizontal [1].

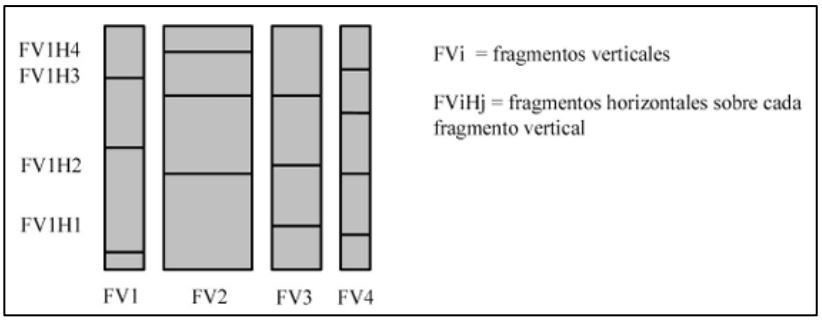

Figura 1. Visualización de una fragmentación VH.

1.3.2 Fragmentación HV

Primero se realiza una fragmentación horizontal, luego, por cada fragmento horizontal se implementa una fragmentación vertical.

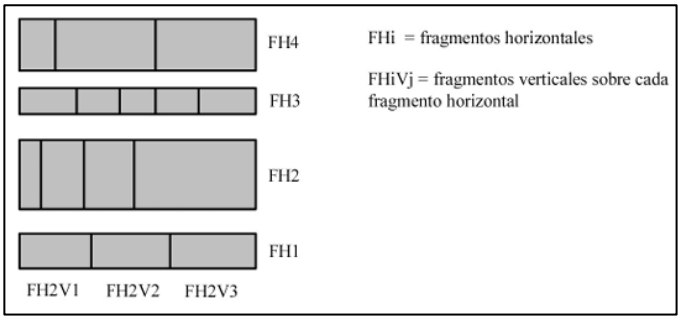

Figura 2. Visualización de una fragmentación HV.

1.3.3 Red de celdas

Es una aplicación de una fragmentación vertical y horizontal al mismo tiempo [1].

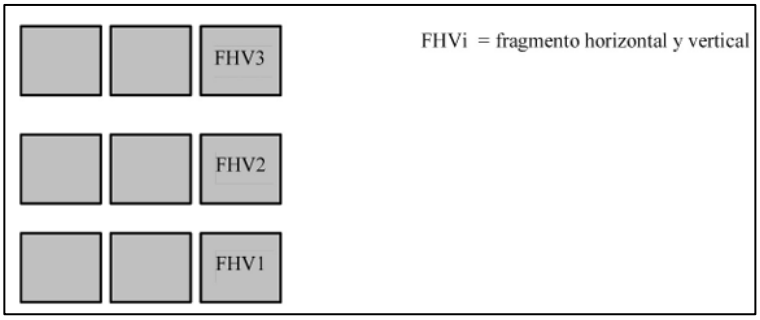

Figura 3. Visualización de una red de celdas.

2. Esquema de Replicación

El esquema de replicación determina desde qué nodo se solicitan los datos y el tipo de operación que se realiza (consulta o actualización), para que estas operaciones se puedan llevar a cabo de forma local, reduciendo así el tráfico en la red y evitando que este cause demoras [1].

Por lo tanto, la replicación es útil para mejorar la disponibilidad de los datos. Por otro lado, existen diferentes tipos de replicación, los cuales se mencionan a continuación [2]:

2.1 Replicación total

Se refiere a la replicación de la base de datos completa en cada sitio del sistema distribuido. Esto puede mejorar la disponibilidad, dado que el sistema puede seguir funcionando mientras al menos un sitio esté activo. También mejora el

rendimiento de recuperación para consultas globales, puesto que los resultados de dichas consultas pueden obtenerse localmente desde cualquier sitio.

Sin embargo, esto puede ralentizar las operaciones de actualización, dado que se debe realizar una única actualización lógica en cada copia de la base de datos para mantener la consistencia de las copias. Por otro lado, la replicación completa hace que las técnicas de concurrencia y recuperación sean más costosas de lo que serían si no hubiera replicación.

2.2 Sin replicación

Implica la ausencia de replicación, lo que quiere decir que cada fragmento se almacena en un solo sitio. Esto requiere que todos los fragmentos sean disjuntos, excepto por la repetición de las claves primarias entre fragmentos verticales o mixtos.

2.3 Replicación parcial

La replicación parcial abarca una gran variedad de posibilidades, donde ciertos fragmentos de la base de datos pueden tener copias en varios nodos, mientras que otros permanecen sin replicar. La cantidad de réplicas por fragmento puede ir desde una sola copia hasta estar presente en todos los sitios del sistema distribuido.

La decisión sobre cuántas copias mantener de cada fragmento depende de los objetivos de rendimiento y disponibilidad del sistema, así como los tipos y frecuencias de las transacciones realizadas en cada nodo [2].

Cuando se necesita alta disponibilidad y la mayoría de las transacciones son de consulta, resulta conveniente utilizar una base de datos totalmente replicada, ya que permite procesar las operaciones desde cualquier nodo. En cambio, si existen transacciones que acceden a fragmentos específicos y estas se originan principalmente desde un mismo sitio, es recomendable asignar esos fragmentos únicamente a dicho nodo. Cuando los datos son consultados desde varios lugares, pueden replicarse en esos mismos nodos, sin embargo, si hay muchas actualizaciones, conviene limitar la replicación [2].

A pesar de estas recomendaciones, encontrar una solución óptima, o incluso una buena, es un problema de optimización complejo. Por tal motivo, al tomar decisiones sobre qué replicar y dónde hacerlo, es necesario considerar los siguientes parámetros [1]:

- **Mínimo coste:** Busca reducir el costo de almacenamiento de cada fragmento en su nodo asignado, el costo de realizar modificaciones cuando el fragmento está replicado en varios sitios, y el costo asociado a la transferencia de datos a través de la red.

- **Rendimiento:** Busca reducir los tiempos de respuesta y aumentar la capacidad de procesamiento del sistema en cada nodo.

2.4 Estructura

El término **esquema de replicación** generalmente se utiliza para referirse a la descripción de cómo se replican los fragmentos dentro de un sistema distribuido [2]. Esta descripción incluye los siguientes apartados, los cuales deben considerarse al definir la replicación de cada relación o fragmento:

Tabla “Nombre de la tabla”

- **Justificación:** Motivo por el cual se replica dicha tabla o fragmento.
- **Nodos involucrados:** Nodos donde estará presente la réplica.
- **Tipo:** Puede ser unidireccional o bidireccional. Si es unidireccional, debe indicarse cuál es el nodo de gestión, es decir, el nodo responsable de aplicar las actualizaciones y propagar los cambios.

En la replicación unidireccional, también llamada de un solo maestro, las actualizaciones se realizan únicamente en un nodo principal (maestro) y los cambios se propagan hacia las réplicas, que solo pueden realizar lecturas. En cambio, la replicación bidireccional o multi-maestro permite que varios nodos realicen actualizaciones, sincronizando los datos en ambos sentidos [3].

3. Esquema de Asignación

El esquema de ubicación (o asignación física) establece la correspondencia entre los fragmentos lógicos, derivados de la fragmentación, y los nodos físicos de la red en un sistema de bases de datos distribuidas [1].

Al igual que el esquema de replicación, su diseño también busca optimizar los mismos parámetros mencionados anteriormente (coste mínimo y rendimiento) [1].

Aunque existen algoritmos para resolver este problema, su complejidad los hace NP-Completos, por lo que en la práctica se prefieren heurísticas subóptimas sobre soluciones teóricamente perfectas.

3.1 Heurísticas de diseño

Para simplificar la asignación, se aplican las siguientes estrategias:

Priorizar procesamiento local:

- Asignar cada fragmento al nodo donde se originan la mayoría de sus operaciones.

Replicación selectiva:

- Replicar solo si predominan operaciones de **consulta** desde múltiples nodos.
- Evitar replicación si las actualizaciones son frecuentes o el coste de sincronización es alto.

Desarrollo de la práctica:

1. Escenario (requerimientos)

La empresa FITEC es una red de gimnasios con 2 sucursales en Quito (Norte y Sur). Cada sucursal cuenta con su propio equipo de instructores, clientes y suplementos de nutrición.

Se desea modelar una base de datos distribuida que permita gestionar toda la información de esta empresa que posee una base de datos centralizada. Para ello, se debe tomar en cuenta los siguientes **requerimientos**:

a. Las ***SUCURSALES*** se identifican por un **código único** (QUITO_NORTE, QUITO_SUR), además poseen **nombre, horario de atención y número de teléfono**.

b. En cada sucursal trabajan varios ***INSTRUCTORES***, quienes solo pueden trabajar en una sucursal a la vez. De cada instructor se almacenan: **cédula, nombre completo, teléfono, email, fecha de Nacimiento, salario, fecha de contratación y dirección** (descripción del sector, calle principal y secundaria).

c. Los ***CLIENTES*** pueden asistir solo a la sucursal en la cual se inscribieron. Además, se guarda la siguiente información: **cédula, tipo de suscripción, nombre completo**, **número de teléfono, email, fecha de nacimiento, fecha de registro,** y **dirección**.

d. Existen 3 tipos de ***SUSCRIPCIONES***: *básica, premium, platinium*. Para cada suscripción se guarda un **código, tipo, descripción, precio** y **duración de los meses del plan**.

e. En las sucursales se realizan ventas de ***SUPLEMENTOS*** deportivos. De cada producto se almacena: **código, código de la sucursal a la que pertenece,**

nombre, categoría (proteína, hidratación o pre-entreno), **precio, cantidad disponible y fecha de vencimiento**. Es necesario especificar en cada suplemento el código de la sucursal a la que pertenece porque cada sucursal vende los suplementos que poseen en su respectiva sede.

f. Los clientes pueden adquirir suplementos deportivos en cualquier sucursal. Por ello, se desea mantener un registro de ***COMPRAS***. Cada compra debe tener un **código único** y debe permitir identificar al **cliente que compró por su cédula, el suplemento adquirido por su código, la cantidad de unidades vendidas, la fecha de la compra** y **el precio unitario por el que fue comprado el suplemento**, dado que, en un futuro el precio de estos puede variar.

g. Adicionalmente, la **sucursal Norte** maneja el uso de **nóminas** para el pago de los salarios de todos los instructores. Por otro lado, la **sucursal sur** es la encargada de realizar **cambios en la información de las sucursales** en el caso de llegar a darse.

2. Grafo relacional

En los requerimientos del negocio se identificó la existencia de dos sucursales independientes, dado que cada una contaba con sus propios instructores, clientes y suplementos. Por tal motivo, se propuso distribuir los datos de manera que cada sede tuviera control sobre la información que le competía directamente.

No obstante, primero se analizó el grafo relacional centralizado de la empresa, el cual se muestra en la Figura 4, para determinar la posibilidad de aplicar técnicas de diseño orientadas a una base de datos distribuida.

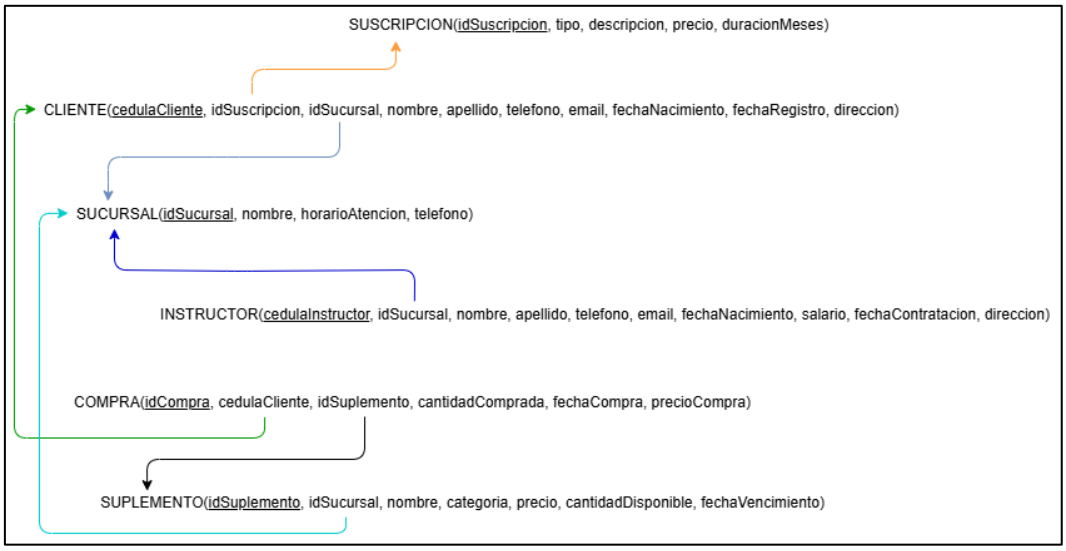

Figura 4. Grafo relacional para la empresa de gimnasios FITEC.

Dentro del grafo relacional, cada sucursal se identificaba mediante un código único, y tanto los instructores, clientes como suplementos se asociaban a una sede mediante claves foráneas, lo cual permitiría aplicar una fragmentación horizontal por sucursal.

También se consideró el caso de compras realizadas por un cliente en distintas sucursales mediante la tabla *COMPRA*. Adicionalmente, en la tabla *INSTRUCTOR* se incluyeron los campos necesarios para gestionar tanto la nómina como la información personal de cada empleado de forma separada.

Por otro lado, los atributos definidos en cada tabla coincidían con la información solicitada por el escenario, y las claves primarias y relaciones se definieron correctamente, lo que aseguraba la integridad de los datos.

De este modo, se concluyó que el diseño del grafo relacional, además de encontrarse alineado con los requerimientos de la empresa, resultaba adecuado para distribuir la información de forma eficiente y segura entre las sedes de FITEC.

3. Sedes y Roles

Se tiene la **Sede Norte** con el rol de nóminas (*idSucursal = QUITO_NORTE*) y la **Sede Sur**, considerando que debe existir independencia local de cada sucursal con respecto a sus datos locales.

4. Campo y condición de fragmentación

- **Campo de fragmentación:** *idSucursal*.

  Se escogió a *idSucursal como* campo de fragmentación debido a que la cantidad de sus valores fueron ser menores o iguales al número de nodos o sedes.

- **Condición de fragmentación:** *idSucursal* = i = {QUITO_NORTE, QUITO_SUR}

  La condición de fragmentación consisitió en una comparación entre los valores de *idSucursal* para así obtener los fragmentos correspondientes.

5. Esquema de fragmentación

a. Fragmentación vertical: Se realizó sobre la tabla ***INSTRUCTOR*** y ***CLIENTE***.

INSTRUCTOR*(cedulaInstructor, idSucursal, nombre, apellido, telefono, email, fechaNacimiento, salario, fechaContratacion, direccion)*

$$NOMINA\_INSTRUCTOR = \Pi_{cedulaInstructor,\ salario,\ fechaContratacion}(INSTRUCTOR)$$

$$INFORMACION\_INSTRUCTOR = \Pi_{cedulaInstructor,\ idSucursal,\ nombre,\ apellido,\ telefono,\ email,\ fechaNacimiento,\ direccion}(INSTRUCTOR)$$

CLIENTE*(cedulaCliente, idSuscripcion, idSucursal, nombre, apellido, telefono, email, fechaNacimiento, fechaRegistro, direccion)*

$$CEDULA\_CLIENTE = \Pi_{cedulaCliente}(CLIENTE)$$

$$CLIENTE = \Pi_{cedulaCliente,\ idSuscripcion,\ idSucursal,\ nombre,\ apellido,\ telefono,\ email,\ fechaNacimiento,\ fechaRegistro,\ direccion}(CLIENTE)$$

b. Fragmentación horizontal primaria: Se realizó sobre el fragmento vertical ***CLIENTE***.

CLIENTE*(cedulaCliente, idSuscripcion, idSucursal, nombre, apellido, telefono, email, fechaNacimiento, fechaRegistro, direccion)*

$$CLIENTE_i = \sigma_{idSucursal=i}(CLIENTE)$$
$$i = \{QUITO\_NORTE, QUITO\_SUR\}$$

c. Fragmentación horizontal primaria: Se realizó sobre la tabla ***SUPLEMENTO***.

SUPLEMENTO(*idSuplemento, idSucursal, nombre, categoria, precio, cantidadDisponible, fechaVencimiento*)

$$SUPLEMENTO_i = \sigma_{idSucursal=i}(SUPLEMENTO)$$
$$i = \{QUITO\_NORTE, QUITO\_SUR\}$$

d. **Fragmentación horizontal derivada:** Se realizó sobre la tabla ***COMPRA***.

Se cumplieron las 2 condiciones necesarias: La tabla ***COMPRA*** no tenía el campo de fragmentación *idSucursal*. Además, la tabla ***COMPRA*** estaba relacionada con la tabla previamente fragmentada ***CLIENTE***.

COMPRA*(idCompra, cedulaCliente, idSuplemento, cantidadComprada, fechaCompra, precioCompra)*

$$\boldsymbol{COMPRA_i = COMPRA \ltimes CLIENTE_i}$$
$$\boldsymbol{i = \{QUITO\_NORTE, QUITO\_SUR\}}$$

e. **Fragmentación mixta**

Se realizó una fragmentación VH:

- **Fragmentación Vertical:** Se usó la tabla ***INFORMACION_INSTRUCTOR*** que ya surgió de la fragmentación vertical previamente realizada en la tabla ***INSTRUCTOR*****.**
- **Fragmentación Horizontal:** Se realizó sobre la tabla ***INFORMACION_INSTRUCTOR***

$$\boldsymbol{INFORMACION\_INSTRUCTOR_i = \sigma_{idSucursal=i}(INFORMACION\_INSTRUCTOR)}$$
$$\boldsymbol{i = \{QUITO\_NORTE, QUITO\_SUR\}}$$

6. Esquema de replicación

A continuación, se describen las tablas replicadas dentro del sistema distribuido, incluyendo su justificación, los nodos involucrados y el tipo de replicación aplicada:

Tabla *SUCURSAL*:

- **Justificación:** La tabla se replica porque es referenciada por tablas como ***CLIENTE, INSTRUCTOR*** y ***SUPLEMENTO***, específicamente en el campo *idSucursal* para identificar en qué sede se encuentra un cliente, instructor o suplemento. Además, la tabla solo contiene datos informativos que suelen permanecer exactamente iguales durante toda la vida funcional de la sucursal. Por lo tanto, realmente son solo datos de consulta.
- **Nodos involucrados:** QUITO_NORTE, QUITO_SUR
- **Tipo:** Unidireccional
- **Nodo de gestión:** QUITO_SUR

Se optó por una replicación unidireccional gestionada desde el nodo QUITO_SUR con el fin de equilibrar la carga operativa entre los nodos, debido a que el nodo QUITO_NORTE asume funciones adicionales como la gestión de nóminas para los pagos de los instructores.

Tabla *SUSCRIPCION*:

- **Justificación:** La tabla se replica porque es referenciada por la tabla ***CLIENTE*** para identificar el tipo de suscripción contratada por cada cliente. Adicionalmente, al contener únicamente tres registros y no presentar cambios frecuentes, se considera una tabla de consulta cuya disponibilidad local en cada sucursal facilitaría el acceso a la información de los planes existentes.
- **Nodos involucrados:** QUITO_NORTE, QUITO_SUR
- **Tipo:** Bidireccional

Se estableció que el tipo de replicación sea bidireccional para la tabla ***SUSCRIPCIONES*** con la finalidad de permitir que cualquiera de los nodos involucrados pueda gestionarla. De esta forma, cualquier modificación realizada se sincroniza en ambos nodos, evitando inconsistencias en la información de los planes ofrecidos por cada sucursal.

Tabla CEDULA_CLIENTE:

- **Justificación:** La tabla, que en realidad es un fragmento de la tabla ***CLIENTE,*** a pesar de estar sujeta a registros y modificaciones frecuentes, debe ser replicada para cumplir con el requerimiento de permitir que un cliente realice compras de suplementos en cualquiera de las dos sucursales. De esta manera, se garantiza que las referencias de los clientes estén disponibles en ambas sedes y que la compra se registre en la sede donde se efectuó.
- **Nodos involucrados:** QUITO_NORTE, QUITO_SUR
- **Tipo:** Bidireccional

Se eligió la replicación bidireccional para la tabla ***CEDULA_CLIENTE***, puesto que cada nodo debe poder gestionar las modificaciones en la tabla en función de los clientes que maneja. No obstante, es importante tener en cuenta que esta tabla solo contiene la cédula del cliente y se utiliza para asociar a los clientes con sus compras, por lo que no se deben realizar actualizaciones directas en ella.

Las modificaciones deben ocurrir únicamente a través de los cambios que se realicen en los fragmentos ***CLIENTE_QUITO_NORTE*** y ***CLIENTE_QUITO_SUR***, que son los encargados de gestionar los datos de los clientes en cada sucursal, de modo que la información se mantenga consistente en todo momento.

7. Esquema de ubicación/asignación

A continuación, se presenta el esquema de ubicación en una tabla que muestra las relaciones originales y la distribución de sus respectivas réplicas y fragmentos en los nodos Norte y Sur:

	QUITO_NORTE	QUITO_SUR
SUCURSAL	SUCURSAL	SUCURSAL
SUSCRIPCION	SUSCRIPCION	SUSCRIPCION
CLIENTE	CLIENTE_QUITO_NORTE CEDULA_CLIENTE	CLIENTE_QUITO_SUR CEDULA_CLIENTE
SUPLEMENTO	SUPLEMENTO_QUITO_NORTE	SUPLEMENTO_QUITO_SUR
COMPRA	COMPRA_QUITO_NORTE	COMPRA_QUITO_SUR
INSTRUCTOR	INFORMACION_INSTRUCTOR_QUITO_NORTE NOMINA_INSTRUCTOR	INFORMACION_INSTRUCTOR_QUITO_SUR

Tabla 1. Esquema de ubicación para la empresa FITEC.

Conclusiones y recomendaciones:

1. Conclusiones

- Se logró el establecimiento del diseño de una base de distribuidas a partir de los requerimientos y el grafo relacional de la empresa FITEC aplicando una metodología descendente.

- Se consiguió identificar la sede Norte con el rol de nóminas y la sede Sur a través del análisis del escenario. Esto también permitió entender la organización y estructura del sistema distribuido.

- Se diseñaron los esquemas de fragmentación, replicación y ubicación de forma correcta a partir del esquema lógica global. De esta manera se pudo establecer la distribución de las relaciones, los fragmentos y las réplicas entre los nodos de la base de datos distribuida.

2. Recomendaciones

- Antes de comenzar con el diseño de una base de datos distribuida a partir de un esquema conceptual global, es necesario comprender los requerimientos de la empresa y verificar que el modelo relacional sea coherente con los datos que se desean almacenar y sus relaciones. Esto permitirá definir una estrategia de distribución adecuada y alineada con las necesidades del sistema.

- Se recomienda tener la menor cantidad de réplicas en una base de datos distribuida, dado que el mantenimiento de la consistencia entre ellas implica un alto costo. Además, la lógica de distribución busca que cada nodo almacene únicamente los datos que necesita para su operación; de lo contrario, no habría ventaja frente a una base de datos centralizada con acceso remoto.

- Es fundamental realizar un análisis cuidadoso para decidir la ubicación óptima de los fragmentos de datos, considerando factores como la frecuencia de acceso, la localización de los usuarios y los patrones de consulta. Una buena estrategia de ubicación mejora significativamente el rendimiento del sistema, reduce la latencia en las operaciones y optimiza el uso del ancho de banda en la red.

Bibliografía:

[1] CASTRO *et al.*, *Desarrollo de Bases de Datos. Casos prácticos desde el análisis a la implementación 2ª edición actualizada.* Alfaomega - Rama, 2013.

[2] R. Elmasri y S. Navathe, *Fundamentals of Database Systems*, 7ª ed. Pearson, 2008.

[3] EnterpriseDB. “PostgreSQL Replication and Automatic Failover Tutorial”. EDB Postgres AI. Accedido el 21 de junio de 2025. [En línea]. Disponible: https://www.enterprisedb.com/postgres-tutorials/postgresql-replication-and-automatic-failover-tutorial#section-7