Capitulo II

CAPÍTULO II: MARCO CONCEPTUAL O REFERENCIAL

2.1 El Modelo Relacional y la Plataforma de Implementación

Este proyecto se fundamenta en el **Modelo Relacional** para la representación y gestión de los datos del sistema de control de stock. La elección de este modelo se basa en su probada solidez, flexibilidad y en la garantía de **Integridad de Datos** a través de la formalización de las relaciones entre entidades.

Como **Sistema Gestor de Bases de Datos (SGBD)**, se ha seleccionado **Microsoft SQL Server**. Esta elección está justificada por su robustez, su escalabilidad para manejar grandes volúmenes de datos y por ser la plataforma que soporta la implementación de **Índices Columnares**, uno de los temas técnicos asignados a la investigación.

2.2 Integridad y Consistencia de los Datos

La calidad de un sistema transaccional como el propuesto (Sistema de Gestión de Stock) depende directamente de la capacidad del SGBD para mantener la integridad y la consistencia de los datos.

2.2.1 Integridad Referencial

La integridad referencial es garantizada por las **Claves Primarias (PK)** y **Claves Foráneas (FK)** definidas en el Modelo Relacional.

- **Claves Primarias:** Aseguran que cada fila en una tabla sea única.
- **Claves Foráneas:** Mantienen la coherencia en las relaciones entre tablas (asegura que cada registro de la tabla `Detalles_Ticket` apunte a un `Ticket` de venta existente y a un `Producto` válido, evitando datos huérfanos).

2.2.2 Propiedades ACID y Transacciones

Las operaciones críticas del sistema, como el registro de una venta o la recepción de un proveedor, deben ser gestionadas como **Transacciones** para cumplir con el estándar **ACID**:

- **Atomicidad:** Una transacción se ejecuta completamente o no se ejecuta en absoluto. En una venta, si la inserción del `Ticket` falla, la actualización del `Stock` debe revertirse (rollback).
- **Consistencia:** La transacción lleva la base de datos de un estado válido a otro, manteniendo la integridad referencial y las reglas de negocio.
- **Aislamiento:** Múltiples transacciones concurrentes no deben interferir entre sí.
- **Durabilidad:** Una vez que la transacción es confirmada (`COMMIT`), los cambios son permanentes en el almacenamiento persistente.

El **Manejo de Transacciones** (Tema 3) y las **Transacciones Anidadas** se implementarán para asegurar la **Atomicidad** y la **Consistencia** en procesos multifase.

2.3 Componentes Programables del Servidor (Procedimientos y Funciones)

La lógica de negocio compleja y las operaciones de manipulación de datos son encapsuladas y centralizadas en el servidor de la base de datos mediante componentes programables.

- **Procedimientos Almacenados (PAs):** Son conjuntos de sentencias SQL precompiladas que residen en el servidor. Se utilizarán para las operaciones **CRUD** (`Insertar`, `Modificar`, `Borrar` registros), ya que mejoran la seguridad y reducen el tráfico de red, pues solo se envía la llamada al PA y no las sentencias SQL completas.
- **Funciones Almacenadas:** Son bloques de código que devuelven un valor. Se emplearán para realizar cálculos o retornar información específica (e.g., el cálculo de un precio final o la edad de un usuario a partir de su fecha de nacimiento), pudiendo ser llamadas dentro de sentencias SQL.

El **Tema 1** se enfocará en demostrar cómo estos componentes programables aumentan la eficiencia y el control sobre la manipulación de datos en el sistema de stock.

2.4 Optimización de Consultas e Indexación

La optimización de consultas es crítica para la performance de cualquier sistema de información. La base de esta optimización es la **Indexación**.

- **Índices Tradicionales (Orientados a Fila):** Los índices son estructuras de datos que aceleran la búsqueda de registros en tablas voluminosas. Se implementarán **Índices Agrupados (Clustered)** y **No Agrupados (Non-Clustered)** (Tema 2) en tablas con alta frecuencia de consulta (como `Productos` o `Tickets`) para reducir drásticamente los tiempos de respuesta. Las pruebas compararán el **Plan de Ejecución** de las consultas antes y después de aplicar los índices para cuantificar la mejora de rendimiento.
- **Índices Columnares (SQL Server):** A diferencia de los índices tradicionales que almacenan datos fila por fila (ideales para transacciones), los índices columnares almacenan los datos columna por columna. Esta arquitectura está optimizada para tareas de **Data Warehousing** y consultas analíticas que procesan grandes agregaciones de datos. El **Tema 4** investigará la mejora de rendimiento que ofrecen estos índices en la tabla con **un millón de registros** para tareas de reportes y análisis, en contraposición con los índices orientados a fila.