UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE CONTROL Y PROCESAMIENTO DE INFORMACIÓN PARA SALIDA DE PRODUCTOS TERMINADOS EN UNA EMPRESA TEXTIL: CASO REAL

TESIS

QUE PARA OBTENER EL GRADO ACADÉMICO DE
INGENIERO EN COMPUTACIÓN

P R E S E N T A

CARLOS DAVID SERVÍN LAGARDE

DIRECTORA

DRA, ANA MARÍA VÁZQUEZ VARGAS

MÉXICO, D.F. JULIO, 2012

AGRADECIMIENTOS

Quiero agradecerle primero que a nadie a mi padre, pilar fundamental en todo lo que soy, por su amor y apoyo.

A mi madre por darme la vida, por sus enseñanzas, determinación y cariño.

A mi tía Adriana y los que se encuentran paradójicamente tan cerca.

A mi familia que siempre ha estado conmigo, mis hermanos, mi abuela, mis primos, tíos, a Gaby.

A la Dra. Ana Maria Vazquez, así como a los profesores y personas comprometidas con la Universidad Nacional Autónoma de México y sus principios. Transmitiendos dia a dia con hechos que un México mejor es posible.

A Argos Navis y los buzos, con quienes en las profundidades, me he reencontrado.

A mis compañeros de bloque y amigos, hermanos que uno escoge.

A Camilo, Alfonso y al abuelo que se encuentran en mi corazón.

PRÓLOGO

Esta tesis trata sobre la implantación de un sistema de control de mercancía desde que esta es solicitada por un cliente hasta su recepción, así como el procesamiento de la información generada. Permitiendo reducir costos y tiempos de entrega, así como mejorar el intercambio comercial entre una empresa real y sus clientes, incrementando la competitividad de la misma.

El proyecto incluyó modelos de trazabilidad, y mecanismos que permiten agilizar la facturación, la salida de mercancía, el manejo de inventarios y la relación con los clientes. Orquestado mediante la implementación de una base de datos durante más de cuatro años continuos de ejercicio profesional.

A partir del éxito obtenido por otros sistemas que, con anterioridad, habían resuelto problemas diversos para esta empresa textil. La dirección de la misma me encargó, con mucha autonomía, la creación y desarrollo del sistema. La libertad que ofrecía el proyecto me permitió plantearlo como investigación y defenderlo en esta tesis para obtener el grado de Ingeniero en Computación.

La participación de mi tutora, la Dra. Ana María Vázquez, especialista en bases de datos, me permitió desarrollar y probar este modelo con una visión crítica y universitaria, profundizando en la lectura y análisis de las teorías que fundamentan la operación actual de las bases de datos.

En este ejercicio se observó que una necesidad fundamental para nuestra práctica profesional es la capacidad de generar sistemas eficientemente y dinámicamente integrados al proceso empresarial, su entorno y su desarrollo. Exponiendo la problemática implícita en la creación y operación de una base de datos que controla una considerable cantidad de información que ingresa de forma simultánea, y describe la solución de su operación mediante disparadores o *triggers* derivados del análisis de las transacciones de los *queries* y su ajuste progresivo.

En el Capítulo 1 referente al planteamiento, se hace una breve introducción exponiendo la problemática de la empresa y los estudios realizados para definirla, así como la propuesta de solución a la misma.

En el Capítulo 2 concerniente al marco teórico, se hace una breve introducción a la teoría de las bases de de datos relacionales, de los modelos de trazabilidad y de los servicios WEB.

En el Capítulo 3 describo brevemente los elementos que utilicé para el diseño del sistema. Se plantea la hipótesis y los requerimientos materiales necesarios para llevarlo a cabo. A continuación propongo la nueva secuencia propuesta para el modelo comercial con la finalidad de corregir los errores del modelo anterior, así como su integración dentro de la empresa. Esto implica definir el alcance que tiene la implementación del nuevo sistema y sus fases de desarrollo. Una vez establecido lo anterior paso a la creación de la base de datos, en este apartado explico con más detalle el manejo de los pedidos dentro de la empresa, asi como la integración de los mismos en una base de datos. Hago una descripción de los *trigger*s y de los tiempos de respuesta de la misma. Por último doy una breve explicación de las interfaces más importantes del sistema y de sus componentes, así como su funcionamiento dentro de mismo modelo.

El capítulo 4 aborda los resultados positivos que encontramos al introducir procesos dentro del sistema comercial de la empresa y finalmente en el capítulo 5 se discuten los resultados a la luz de mejores acciones correctivas y preventivas, pensando en el reto de crear un proceso de mayor estabilidad.

Esperamos que éste trabajo ofrezca una nueva perspectiva sobre el manejo de bases de datos para un alumno que se encuentre cursando la carrera de Ingeniería en Computación, así como referencia y punto de partida para posteriores investigaciones.

ÍNDICE TEMÁTICO

ÍNDICE DE ILUSTRACIONES

ÍNDICE ANEXOS

1. PLANTEAMIENTO

1.1. INTRODUCCIÓN

La presente tesis pretende mostrar de forma general soluciones reales, cuantificables, a una problemática medular en las empresas, la creación e implementación -sin interrumpir el proceso de producción- de un mejor sistema de salida de producto terminado. Es un modelo muy complejo que abarca desde la capacitación del personal y la implementación de una metodología de trabajo, pasando por el desarrollo de interfaces y la integración a diferentes sistemas, hasta el desarrollo y administración de una base de datos.

El trabajo consiste en el diseño de un sistema que simplifica la preparación de pedidos evitando fallas en la elaboración y que de seguimiento a la mercancía, de forma dinámica, mediante un sistema de trazabilidad; este sistema también simplifica la entrega de documentación a los clientes y genera información útil en la toma de decisiones, manteniéndola integrada y a disposición del personal responsable. Finalmente el sistema debe integrar esta aplicación al sistema general mediante una filosofía de *Sistemas Orientados a Servicios*.[1]

La tesis *Sistema De Control Y Procesamiento De Información Para Salida de Productos Terminados En Una Empresa Textil: Caso Real,* es producto de una experiencia laboral real y pretende sintetizar varios años de trabajo sobre la gestación, desarrollo e implementación de software de intercambio comercial. Se plantean en él las demandas reales de la empresa -haciendo hincapié en ellas-, las diferencias entre estas demandas y las soluciones propuestas, y la manera en que estas diferencias se conciliaron. El trabajo se desarrolla en cinco partes fundamentales.

1) Definición y diseño del *Sistema Gestor de Bases de Datos*. Abarcando, la base de datos relacional, el diseño conceptual, el modelo de diseño entidad-relación, el diseño lógico, el diseño físico, los procedimientos almacenados, y los disparadores ó *triggers*.
2) Implementación de la base de datos.
3) Revisión de: optimización de consultas, tipos de cursores de búsqueda, algoritmos de búsqueda y ordenamiento en la base de datos, y colisiones; establecimiento de los mecanismos de comunicación *System.Data.SqlClient, ActiveX Data Objects (ADO)* y *Open Data Base Connectivity (ODBC)* para posibilitar el acceso a la información.
4) Diseño de una interfaz gráfica y de programas *Visual Basic.Net*, resaltando su importancia en la solución del problema planteado.
5) Análisis del impacto de la utilización de *Servicios WEB* en la tendencia de intercambio entre proveedores y clientes, en los resultados y en la situación actual real de la empresa.

No debe pasarse por alto que el sistema que nos ocupa se encuentra en funcionamiento y que se desarrolló partiendo de formas de producción y control establecidas previamente por la dirección de la empresa. El proceso de elaboración de este nuevo sistema, al ser un ejercicio realizado *"en vivo*", permite mostrar las interacciones de los procesos de análisis y de desarrollo teórico y las limitantes impuestas por *"la realidad*" o por *"terceros"* (la dirección, los clientes, los procesos de producción).

[1] En inglés *Service Oriented Architecture*, es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

1.2. PLANTEAMIENTO DEL PROBLEMA

Durante más de 10 años he trabajado como consultor en la implementación de Sistemas Comerciales para empresas medianas del ramo textil. Hace aproximadamente 4 años la directiva de una de estas empresas, con más de 50 años de existencia y dedicada a la fabricación de calcetines para bebés, niñas, niños, damas y caballeros en todas sus gamas, solicitó mis servicios para analizar su disminución de competitividad frente a sus clientes. El objetivo de la dirección era mejorar los tiempos de entrega y los márgenes de ganancia optimizando la detección de las necesidades de sus clientes para proporcionarles el producto que les interesaba en la forma, el lugar y el tiempo que éstos desearan. La propuesta debía tener en cuenta los nuevos modelos de distribución *JIT* [2] que varios de los clientes ya habían establecido.

El primer paso consistió en el análisis de la cadena de abastecimiento y en particular del canal distribución[3]. Se analizaron los siguientes ejes:

- *Clientes y desempeño.*
- *Organización o estructura del sistema de distribución.*
- *Tecnología e infraestructura.*
- *Integración al modelo productivo.*

1.1.1. *CLIENTES Y DESEMPEÑO.*

Para determinar los alcances del nuevo sistema fue necesario hacer una evaluación de los requerimientos de los clientes, y de la forma en que éstos los satisfacían, mediante el análisis detallado de las características y necesidades de la clientela y del servicio que se le daba. El primer resultado de este trabajo fue la clasificación de los clientes en grupos con características afines de acuerdo a 7 condiciones:

1) Forma de ingresar el pedido.
2) Cantidad de mercancía solicitada.
3) Control de calidad y verificación de artículos surtidos.
4) Forma de entrega.
5) Tiempo de entrega.
6) Información comercial demandada.
7) Cumplimiento de requerimientos por parte de la empresa.

El resultado del análisis de ventas mostró la existencia de tres grupos distintos de clientes: el primero conformado por cadenas de autoservicio representó el 75%, el segundo grupo conformado por almacenes y tiendas comerciales con un 17% y, finalmente, el tercer grupo incluyó al público en general con un 8% de las ventas totales.[4] Como se muestra en la figura 1

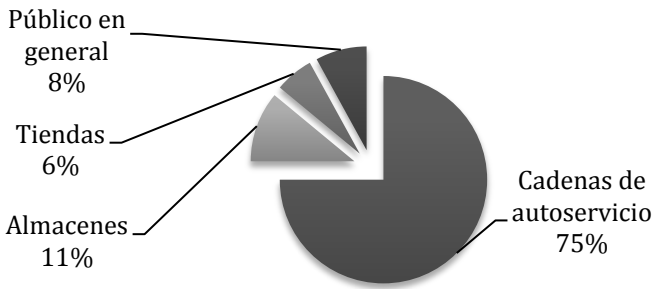


FIG. 1 VENTAS TOTALES POR TIPO DE CLIENTE

[2] Just in time.
[3] **Canal de Distribución** o **Proceso de Distribución** (en inglés, *Supply Chain*) se entiende la compleja serie de procesos de intercambio o flujo de materiales y de información que se establece tanto dentro de cada organización o empresa como fuera de ella, con sus respectivos proveedores y clientes. El punto de partida del canal de distribución es el productor. El punto final o de destino es el consumidor.
[4] Ver Anexo I Piezas promedio por tipo de cliente (2008)

La complejidad en la entrega de mercancía y el procesamiento de la información varía considerablemente dependiendo de cada grupo.

El caso más sencillo es el del grupo de los clientes conformado por público en general.

- Un vendedor de la empresa textil ingresa los pedidos solicitados por el cliente manualmente en el sistema.
- La cantidad de mercancía es limitada: en promedio 3,300 piezas por pedido.
- La entrega se hace directamente en la planta.
- El cliente verifica personalmente su mercancía.
- La facturación se entrega en papel y se elabora en el momento.
- El pedido se elabora y entrega el mismo día.
- El sistema comercial cumple con los requerimientos para este tipo de clientes.

El siguiente grupo es el de las tiendas y almacenes.

- Los pedidos se ingresan de forma manual al *ERP*[5], ya sea a través de un vendedor al cual un cliente le entregó un pedido, o personal del *Departamento de Venta* al cual le hicieron una solicitud de mercancía a través de una llamada telefónica o un correo electrónico.
- La cantidad de mercancía surtida varía entre 18,000 y 45,600 piezas por pedido.
- La mercancía se envía al comercio.
- El control de calidad y la verificación de los artículos surtidos se hace en el sitio de entrega.
- La facturación se hace en papel y se elabora una vez empaquetado el pedido. La factura se entrega con el pedido, no se pide información adicional.
- Los tiempos de entrega son holgados (varían de una semana a un mes).

Aunque existen algunas quejas por la entrega de mercancía y la facturación, el sistema comercial de este grupo no requiere de una atención especial y el sistema comercial soporta este tipo de pedidos.

Por último está el grupo de las cadenas de autoservicio, que son los clientes cuyas necesidades requirieron de la construcción del sistema que conforma esta tesis. El sistema de ventas de este grupo consiste en:

- La recepción de pedidos se hace generalmente de forma automática mediante *EDI*.[6]
- La cantidad de mercancía surtida es elevada (en promedio es de 118,000 piezas por pedido).
- Dependiendo del tipo de cliente, el control de calidad se puede hacer directamente en los centros de distribución o en las tiendas.
- La mercancía se envía a *Centros de Distribución (CEDIS)* o directo a tienda.
- La Facturación y ASN[7] se envían por medios electrónicos, es necesario descargar información adicional para el empaquetado de la mercancía.
- Los tiempos de entrega varían dependiendo del cliente (pueden ser de 3 días a 1 mes).
- El sistema comercial no cumple con los estándares de calidad impuestos por algunos clientes que han adoptado modelos *JIT.*

[5] ERP, Programa para la planificación de los recursos de la empresa del inglés: *Enterprise Resource Planning*.
[6] EDI **intercambio electrónico de datos** es la transmisión estructurada de datos entre organizaciones por medios electrónicos. Se usa para transferir documentos electrónicos o datos de negocios de un sistema computacional a otro. El intercambio electrónico de datos puede realizarse en distintos formatos: EDIFACT, XML, ANSI ASC X12, TXT.
[7] ASN Advanced Shipping Notice ó Aviso Anticipado de Embarque. Es una notificación de entregas pendientes, similar a una lista de empaque. Por lo general se envían en formato electrónico y es un documento EDI común. Se puede utilizar para listar el contenido de un envío de mercancías, así como información adicional relacionada con el transporte, tales como información de la orden, la descripción del producto, características físicas, tipo de embalaje, marcas, información del transportista, y la configuración de los bienes dentro de la equipo de transporte. El ASN permite al remitente describir el contenido y la configuración de un envío de varios niveles de detalle.

1.1.2. *TECNOLOGÍA E INFRAESTRUCTURA DE LA EMPRESA.*

La empresa cuenta con varios sistemas propios entre ellos Nómina, Producción, y un *ERP*. El sistema de *ERP* no es lo suficientemente flexible para las demandas de los clientes, tampoco cuenta con mecanismos internos de control y distribución de mercancía. Sin embargo, si cubre las necesidades administrativas básicas como administrar los inventarios, los pedidos del cliente, la facturación y la contabilidad de la empresa.

1.1.3. *ANÁLISIS DEL SISTEMA DE DISTRIBUCIÓN ANTERIOR.*

La siguiente parte del estudio requirió varios análisis y se centro en la organización del canal de distribución y, en particular, del empaquetado, del envío de la mercancía y de la información comercial. Se estudiaron cinco áreas: el Departamento de Ventas, el *APT*[8], el Departamento de Sistemas a cargo del *ERP*, el Departamento de Facturación y por último el Departamento de Envíos.

El canal de distribución estaba organizado de la manera siguiente: el proceso se inicia en el Departamento de Ventas al realizarse un pedido que ingresa al sistema de forma manual, si es un vendedor quien lo registra, o mediante *EDI, que* integra el pedido de manera automática a la base de datos. El estándar que utiliza la empresa para el intercambio electrónico de datos es *EDIFACT*[9] pero algunos clientes han empezado a migrar sus modelos de transferencia electrónica a *Servicios Web* o en inglés *Web Services* así como a descarga de pedidos de sitios en Internet.

El proceso comienza en el Departamento de Ventas cuando ingresa un pedido; como ya comenté este puede ingresar de dos diferentes maneras: mediante *EDI* o de forma manual. Cuando el pedido ingresa de forma manual el cliente contacta a un vendedor que es quien lo ingresa al sistema.

Una vez ingresado el pedido, el Departamento de Ventas verifica inventarios y fechas de vencimiento asignando prioridades e imprimiendo guías de preparación para enviarlas al almacén *APT*. Estas guías de preparación contienen los artículos y cantidades de empaque solicitados.

El jefe de Almacén recibe los pedidos y asigna un preparador responsable que, dependiendo del tamaño del pedido, puede tener uno o más ayudantes. La mercancía solicitada se distribuye, globalmente, en diferentes mesas de trabajo y se introduce en cajas; el personal encargado sella y anota el contenido de cada caja manualmente. Una vez terminado este proceso el preparador hace un conteo de todos los artículos empacados que deben coincidir con los artículos extraídos del Almacén.[10] Las cantidades surtidas se anotan en la guía de embarque y se envían al Departamento de Facturación.

Basándose en las guías de embarque, el Departamento de Facturación anota las cantidades surtidas y factura manualmente. Una vez facturado el pedido, el Departamento de Ventas o el área de Códigos imprimen etiquetas por caja[11], genera y envía el *ASN* a los clientes. Simultáneamente, el Área de Facturación imprime las guías de salida globales[12] por pedido.

Posteriormente se entregan las etiquetas al *APT* y se pegan caja por caja. Una vez etiquetadas las cajas, se remiten al Departamento de Envíos donde se acomodan nuevamente, se les asigna una ruta de

[8] *APT, Almacén de Producto Terminado*.
[9] Es un estándar de la Organización de las Naciones Unidas para el intercambio de documentos comerciales en el ámbito mundial. Existiendo sub estándares para cada entorno de negocio (distribución, automatización, transporte y aduana) o para cada país.
[10] Cuando los pedidos son muy grandes existe un encargado de revisar que la mercancía se encuentre correctamente empaquetada en las cajas, inclusive llega a ser revisada dos veces la última revisión por el jefe de *APT*.
[11] Las etiquetas por caja se muestran las cantidades y artículos por caja, así como los requerimientos adicionales de cada cliente, como pueden ser sucursal, centro de distribución etc.
[12] Las guías de salida indican la cantidad de artículos, número total de piezas, cantidad de cajas, centro de distribución, horario de la cita de entrega, he información adicional por cliente.

distribución, y se espera su transporte ya sea propio o a través de compañías externas como se muestra en la figura 2 “Diagrama secuencial de la salida y facturación de pedidos”.

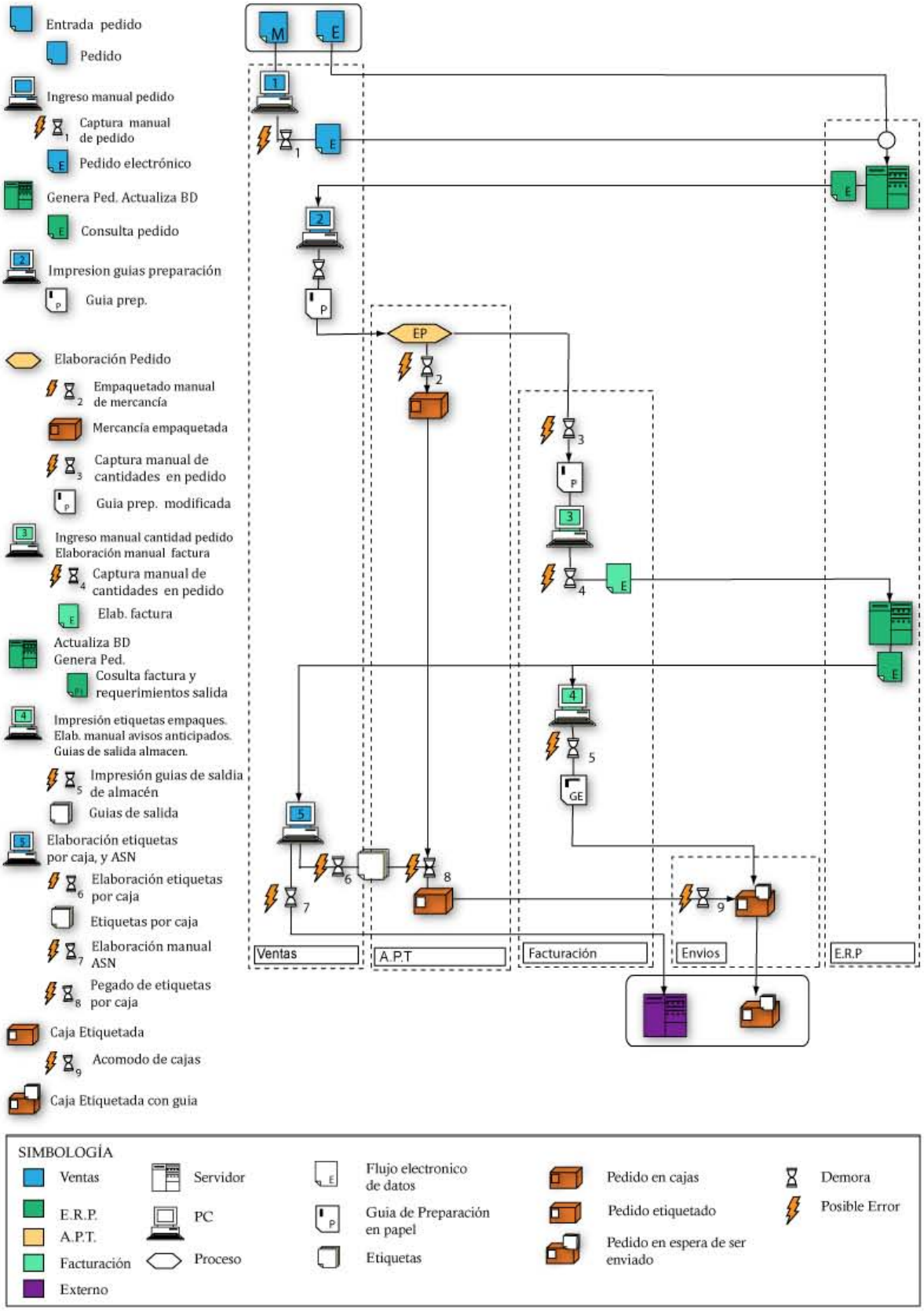


FIG. 2 DIAGRAMA SECUENCIAL DE LA SALIDA Y FACTURACIÓN DE PEDIDOS

1.1.4. *HIPÓTESIS SOBRE LOS RESULTADOS DEL ANÁLISIS.*

Una vez concluido un primer análisis se observó que un alto porcentaje de las ineficiencias organizativas se debían a:

- Predominio de procesos manuales con una posibilidad de error elevada.
- Sistemas ineficaces para compartir la información, en el seno de la organización, que impedían hacer un seguimiento pulcro de los procesos de negocio de principio a fin.
- El cumplimiento de las normas legales obligaba a manejar grandes cantidades de información en formatos complicados; e,
- Ineficiencias propias del Servicio a Clientes.

Cada una de estas situaciones impactaba negativamente la productividad de los empleados, poniendo en riesgo la capacidad de crecimiento y competitividad de la empresa.

Las aplicaciones y las líneas de negocio no compartían información y, por si fuera poco, no contaban con sistemas que permitieran tener un control de sus procesos. Por consiguiente, no podían aportar una visión general de los procesos de negocio cuando éstos abarcaban varias áreas funcionales.

Antes de la implementación del nuevo sistema, se requería indefectiblemente de la intervención humana para introducir a mano los datos entregados por un sistema a otro distinto e incompatible, para que la información se moviera a través de sistemas distintos, dentro y fuera de las fronteras de la organización.

Otro análisis, más detallado, mostró que la gran mayoría de fallas se encontraban en las últimas etapas del sistema de distribución, y que el esfuerzo total de la empresa no estaba orientado hacia la satisfacción del cliente sino a la producción y venta de mercancía.

1.1.5. *PROPUESTA A LA DIRECCIÓN.*

Para corregir los problemas encontrados propuse tres pasos:

1) Desarrollar mecanismos de control y detección de fallas que evitaran el error involuntario mediante la automatización de los procesos y la generación de sistemas de alarma. Este esfuerzo no sólo se realizaría a través de herramientas de software, sino también mediante el desarrollo de una metodología de trabajo apoyada en dichas aplicaciones, y también, desarrollando manuales, capacitando al personal, estandarizando procesos y generando modelos de gestión apoyándonos en bases de datos e interfaces sólidas y fáciles de utilizar.

2) Integrar sus sistemas de gestión[13] desarrollando un modelo basado en la *Arquitectura orientada a servicios*, mediante sistemas distribuidos y/o tecnología *ADO.Net.*

   Así, sí segmentamos la empresa en forma modular, por departamentos, tendremos herramientas que nos ayudarán a gestionar, organizar, dirigir, planificar, controlar y conocer cada área, apoyados en herramientas diseñadas con funciones específicas y programadas con los lenguajes más adecuados para cada trabajo. Esto facilita la relación interna (entre los departamentos) y la externa (entre las áreas y el mundo exterior). Cuando alguna de estas áreas necesita algún dato, lo solicita como servicio y el módulo solicitado lo provee.

[13] Son todos los sistemas, aplicaciones, controles, soluciones de cálculo, metodología, etc., que ayudan a la gestión de una empresa en los siguientes aspectos generales:
Herramientas para el registro de datos en cualquier departamento empresarial.
Herramientas para el control y mejora de los procesos empresariales.
Herramientas para la consolidación de datos y toma de decisiones.

Existen diferentes formas para la solicitud de datos a los servicios y para las respuestas:

- SOA
- SOAP
- Servicio Web
- XML

Éstos servicios interaccionan o intercambian información a gran velocidad permitiendo tomar decisiones "*on Edge"* y sin intervención humana; extienden, además, esta funcionalidad no sólo a los diferentes departamentos sino, de igual manera, a los diferentes clientes que cuentan con modelos similares, como se muestra en esquema de la figura 3.

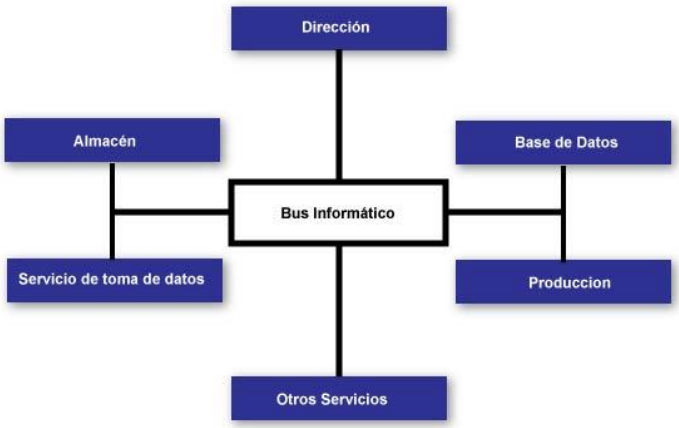


FIG. 3 MODELO ORIENTADO A SERVICIOS

3) Implementar modelos de trazabilidad[14] tanto interna[15] como externa[16]; esto permite mejorar los tiempos de respuesta exigidos y los volúmenes de información administrados. Propuse como la mejor opción un sistema basado en el *Estándar de Trazabilidad GS1*.

   Este estándar maximiza el uso de herramientas del *Sistema GS1* establecidas e implementadas a nivel mundial para identificar en forma singular cualquier "artículo rastreable", describiendo la creación de registros de transacciones y permitiendo una rápida comunicación de los datos correspondientes al artículo rastreable entre socios comerciales. Cumple también con el requisito fundamental tanto reglamentario como comercial de rastrear el origen (un paso atrás) y el destino (un paso adelante) de un producto en forma costo-efectiva en cualquier punto de la cadena de abastecimiento, independientemente de la cantidad de socios comerciales y pasos involucrados en el proceso comercial.

   El *Estándar de Trazabilidad GS1* es una descripción de alto nivel del proceso de trazabilidad que permite y promueve la colaboración en la cadena de abastecimiento. Al mismo tiempo, permite a cada compañía diseñar su propio sistema de trazabilidad en lo que respecta a alcance, precisión y automatización, en función de sus propios objetivos comerciales, maximiza la inter-operabilidad entre sistemas de trazabilidad a lo largo de toda la cadena de abastecimiento ayuda en la

[14] "Se entiende trazabilidad como el conjunto de aquellos procedimientos prestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto o lote de productos a lo largo de la cadena de suministros en un momento dado, a través de unas herramientas determinadas." Comité de Seguridad Alimentaria de AECOC. Ver Anexo II Trazabilidad.
[15] Obtener la traza que va dejando un producto por todos los procesos internos de una compañía, con sus manipulaciones, su composición, la maquinaria utilizada, su turno, su temperatura, su lote, etc., es decir, todos los indicios que hacen o pueden hacer variar el producto para el consumidor final.
[16] Externalizar los datos de la traza interna y añadirle algunos indicios más si fuera necesario, como una rotura del embalaje y un cambio en la cadena.

adecuación a la reglamentación y en el cumplimiento de los requisitos comerciales específicos para el sector de la industria.

Este estándar sirve de fundamento para que todos sus miembros puedan utilizarlo como punto de partida para identificar sus requisitos particulares; es, también, un marco de referencia que permite que las empresas y gobiernos de todo el mundo cuenten con un enfoque común y compartan sus principios clave. Además, contar con un proceso de trazabilidad basado en estándares es un elemento que puede demostrar que una organización cumple con los requisitos de responsabilidad empresarial.

Un sistema de trazabilidad que está soportado por esta infraestructura se basa en las *Tecnologías de la Información y las Comunicaciones (TIC)* y puede brindar importantes utilidades a los diferentes actores de una cadena de valor. Estas utilidades pueden ser una gestión eficiente de la logística y del suministro o un aumento en la productividad. Así, la incorporación de tecnología informática es una inversión que asegura la llegada de los productos a los mercados.

Esta práctica es factible de certificación. Por ejemplo, en los sistemas de gestión de calidad, de gestión medioambiental y sistemas de control, conocidos como cadena de custodia.

1.1.6. *MODELO PROPUESTO PARA EL DESARROLLO DE SOFTWARE*

A partir de las sugerencias anteriores, y en concordancia con la dirección, consideré que el mayor impacto y el menor costo se obtendrían mediante el desarrollo de un sistema en espiral que se explica en el anexo I.III y se ilustra en la figura 4 , focalizado en el sitio donde divergen la mayoría de los procesos del sistema comercial: el Almacén de Producto Terminado *APT,* como se ilustra en la figura 4. Se desarrollaría un prototipo que fuera capaz de generar en forma automática la documentación comercial necesaria para los clientes y que, a su vez, se pudiera integrar a los sistemas aislados que en ese momento operaban en la empresa, dándose así un primer paso hacia un modelo de trazabilidad externa.

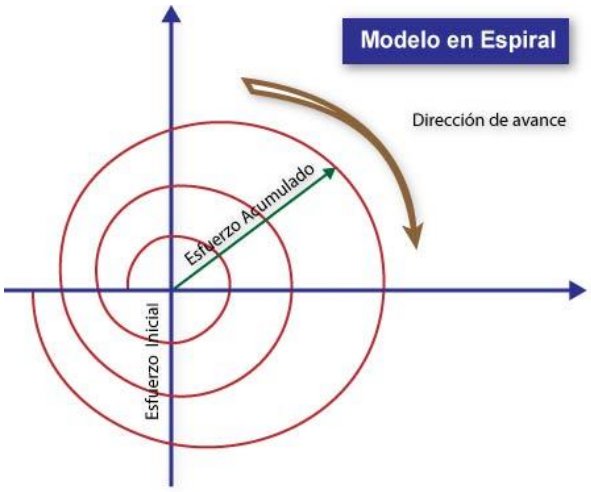


FIG. 4 MODELO EN ESPIRAL

Encontramos que planificar un proyecto con esta metodología era muy complicado debido a la incertidumbre en el número de iteraciones necesarias para terminarlo. En este contexto la evaluación de riesgos es de la mayor importancia y, para grandes proyectos, dicha evaluación requiere la intervención

de profesionales de amplia experiencia. En el caso de la empresa textil una forma de disminuir el riesgo, que implica un modelo de estas características, es que cada etapa o ciclo presente mejoras significativas que reditúen por sí mismas.

Alcances y limitaciones de la tesis

El proyecto real abarca 7 etapas o espirales:

1. Creación de un prototipo en *Visual Basic 6.*
2. Migración de *Visual Basic 6 a .Net*
3. Etapas de preparación y consolidación de pedidos.
4. Creación de aduana en el Departamento de Envíos.
5. Creación de salidas de envíos y guías de preparación.
6. Integración del nuevo sistema al *ERP* mediante *SQL* ó Servicios Web.
7. Asignación y facturación de pedidos de forma automática.

Esta tesis abarca las primeras cinco espirales consolidándolas, sin mostrar cada una de las etapas que se realizaron y muestra el primer modelo funcional del sistema.

El desarrollo de un sistema de distribución y trazabilidad abarca una infinidad de puntos, que van desde la investigación de operaciones y la capacitación del personal hasta la el control de calidad, éstos no serán abordados en su totalidad en este trabajo, pues no forman parte del objetivo fundamental de la tesis. Sin embargo, haré una mención general de algunos de los requerimientos que se fueron presentando con la finalidad de enriquecer la fundamentación de esta tesis.

En esta tesis nos centraremos fundamentalmente en la implementación y desarrollo de una base de datos en *Msqiserver*, así como algunas interfaces del sistema desarrolladas en *Visual Basic .Net*. La extensión del tema me impide tratar aquí con amplitud los temas de seguridad y mantenimiento.

En relación a los *Servicios Web*, nos limitaremos a mostrar algunos ejemplos de conexiones y envió de documentación. Daré algunos ejemplos de conexión desde el punto de vista del enlace de cada uno de ellos.

Objetivos previstos y metas reales (Empresa)

- Desarrollar un sistema que permita, a la empresa y a sus clientes, intercambiar información y controlar y certificar la mercancía enviada, y; que produzca transacciones comerciales en menos tiempo y con menos errores que el sistema actual, basado en papel, reduciendo gastos de envío -al suprimir los formularios y el consolidado de pedidos en papel- y, reduciendo el espacio de almacenamiento, tanto de cajas como de registros de documentación.
- Capacitar al personal y generar mecanismos automáticos; reducir los procesos administrativos para limitar el error humano.
- Agilizar los procesos de negocio acortando, por ejemplo, los plazos de entrega de días a horas.
- Mejorar el control del inventario generando procesos más inteligentes de producción puntual de mercancía ligada a la demanda real y permitiendo la entrega JIT.
- Diseñar un modelo flexible que se ajuste rápidamente a las demandas tanto de la empresa como de los clientes.
- Desarrollar aplicaciones que reduzcan la brecha comercial entre el comprador y el cliente, puede utilizarse como una estrategia de marketing, que permitan para fortalecer o formalizar alianzas comerciales.
- Se espera, para finales de 2012, una reducción del 90% en quejas y devoluciones.

- Se espera, para Julio de 2012, contar con sistemas de consolidado de pedidos y de blindaje certificado por los clientes permitiendo vías de distribución más beneficiosas para la empresa.

Limitaciones empresa

En esta empresa los sistemas funcionan en ambiente *Windows*. Por lo que todos los programas que se desarrollen deben ser *Microsoft*.

Toda la compañía se encuentra centralizada en una sola locación, enlazada mediante una red LAN[17], por lo que no fue solicitada la creación de una aplicación basada en tecnología WEB. Las conexiones distantes se elaboran mediante aplicaciones de escritorio remoto.

La estructura y configuración del sistema depende en gran medida de los lineamientos de los clientes, y de los vínculos con el ERP.

[17] LAN, Red de área local, del inglés local área network.

2. MARCO TEÓRICO

2.1. INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES

2.1.1. *MODELADO DE DATOS*

Todo modelo es una representación limitada de una realidad específica, que tiene un propósito en particular y que se centra en la representación de aquellos aspectos que son importantes para ese propósito. (Wes, 2010)

En el caso de los modelos de datos se busca reproducir la información real, que deseamos almacenar en un sistema informático, definiendo la estructura y el significado de los datos. (Wes, 2010) En 1975 el *Comité de estándares y requerimientos Estadounidense* (SPARC por sus siglas en inglés: *Standards Planning And Requirements Committee*) del *Instituto Nacional Estadounidense de Estándares* (ANSI: *American National Standards Institute*) propuso una arquitectura en la que están basados la mayoría de los sistemas comerciales modernos. Esta arquitectura plantea tres niveles: el esquema[18] físico o de máquina, el nivel externo o de usuario, y el esquema conceptual. Así mismo describe las interacciones entre éstos tres niveles y todos los elementos que los conforman. (Jardine, 1976)

- **Nivel externo**. Es el punto de partida -una visión del mundo real desde una perspectiva particular- para un determinado propósito que es la perspectiva de la aplicación. (Wes, 2010) En el nivel externo se sitúan las diferentes visiones lógicas que los procesos usuarios (programas de aplicación y usuarios directos) tendrán de las partes de la BD que utilizarán. Estas visiones se denominan esquemas externos. (Paré, 2005) Los puntos de vista externos son el lugar adecuado para mantener los requisitos de datos para un contexto y las reglas que se aplican a ella.

- **Esquema conceptual**: Describe la estructura de toda la base de datos para una comunidad de usuarios, independientemente del sistema de gestión de base de datos que se utilizará. Trabaja con elementos lógicos como entidades, atributos y relaciones, manejando una sola descripción conceptual. Sirve de referencia para el resto de los esquemas (Paré, 2005).

  Un esquema conceptual es un modelo básico o neutro que es capaz de soportar cualquier validez externa (y tal vez el cambio) que cae dentro de su alcance (Wes, 2010).

  Los modelos Entidad Relación y Entidad Relación Extendido, son modelos que permiten crear esquemas conceptuales y se explicaran en las siguientes secciones.

- **Esquema físico o interno**: Representa la forma en que los datos se almacenan físicamente. Puede haber muchos modelos físicos válidos para un modelo conceptual. El requisito es que un modelo físico debe ser capaz de apoyar al modelo conceptual.

[18] Se denomina esquema a una descripción específica en términos de un modelo de datos. El conjunto de datos representados por el esquema forma la base de datos.

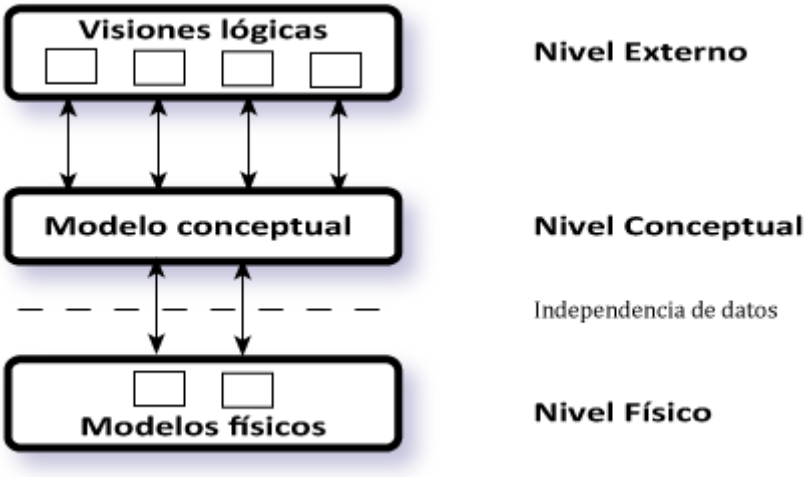


FIG. 5 ARQUITECTURA ANSI-SPARC TRES NIVELES

En la ilustración anterior (fig. 5) aparecen los distintos esquemas que llevan desde el mundo real hasta la base de datos física.

La arquitectura de tres niveles es útil para explicar el concepto de *independencia de datos* que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Para lograr esa independencia se requiere tener independencia física y lógica de los datos. Esto significa:

- **Independencia física de los datos**. Aunque el esquema físico cambie, el esquema conceptual no debe verse afectado. En la práctica esto significa que aunque se añadan o cambien discos u otro hardware, se modifique el sistema operativo o se realicen otros cambios relacionados con la física de la base de datos, el esquema conceptual permanece invariable (Codd, 1990).

- **Independencia lógica de los datos**. Significa que aunque se modifique el esquema conceptual, la vista que poseen las aplicaciones (los esquemas externos) no será afectada. La adición o eliminación de los nuevos conceptos no debe cambiar los elementos que no hacen referencia explícita. (Codd, 1990)

Para lograr esa independencia se utilizan *Sistemas de Gestión de Bases de Datos* (SGBD) cuyas características veremos en la sección 2.1.3, la mayoría de dichos sistemas están basados en el Modelo Relacional que explicaremos en la sección 2.1.2

Modelo entidad relación

Introducción

El *modelo entidad relación* fue ideado por Peter Chen (entre 1976 y 1977) quien lo público por primera vez en el artículo *The Entity Relationship Model - Toward A Unified View of Data*, que es considerado uno de los artículos más influyentes en informática.

Se trata de un modelo que sirve para crear esquemas conceptuales de bases de datos a partir de los esquemas externos del nivel anterior. Es prácticamente un estándar para realizar esta tarea. Se le llama modelo *E/R* o *EI (Entidad / Relación)*. Sus siglas más populares son *E/R* y sirven tanto para el inglés como para el español.

Entidades

Una entidad es algo que se puede identificar de forma distintiva. Una persona en particular, una empresa o un evento son ejemplos de entidades. (CHEN, 1976) No es una propiedad concreta sino un objeto que puede poseer múltiples propiedades (*atributos*).

Ejemplo: Sr. Guillermo; factura número 32456; coche matrícula 3452BCW

Conjunto de entidades

Las entidades que poseen las mismas propiedades forman conjuntos de entidades que serán diferentes en función de dichas propiedades. Ejemplos de conjuntos de entidades son los conjuntos: EMPLEADOS, PROYECTOS, DEPARTAMENTOS. (CHEN, 1976)

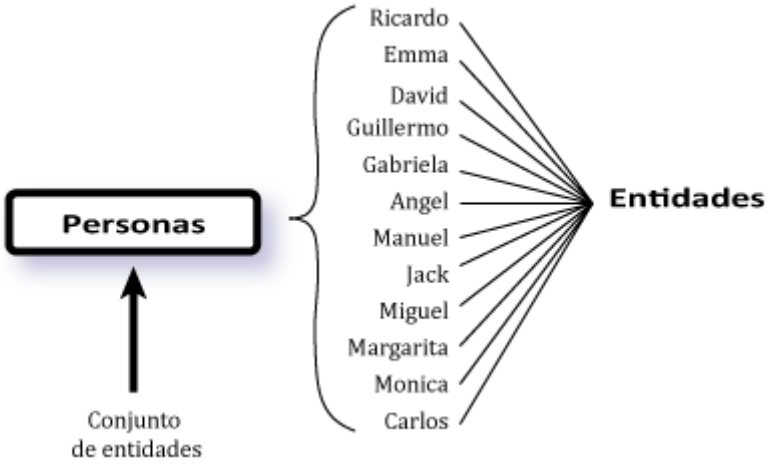


FIG. 6 ENTIDADES Y CONJUNTO DE ENTIDADES

Existen dos tipos diferentes de entidades: regulares (o fuertes) y débiles. (Date, 2001)

- *Regulares o fuertes***.** Son las entidades que tienen existencia por sí mismas sin depender de otras. Por ejemplo EMPRESA.
- *Débiles*. Su existencia depende de otras entidades. Por ejemplo la entidad EMPLEADO sólo podrá tener existencia si existe la entidad EMPRESA.

Atributos

La información sobre una entidad o un relación se obtiene mediante la observación o la medición, y se expresa mediante un conjunto valores. (CHEN, 1976) Por ejemplo:"003548", "GGUI710834D1", "Guillermo" y "García", "54-34-81-95-44", "$70,000". Valores que se clasifican en grupos diferentes, tales como Número de Empleado, RFC, Nombre, Apellido, Teléfono y Sueldo. Las propiedades que conforman una entidad se denominan atributos.

Las propiedades (atributos) se pueden clasificar de diferentes maneras: (Date, 2001)

- **Simples o compuestas.** Por ejemplo la propiedad compuesta "nombre del empleado" podría estar conformada por las propiedades simples "nombre", "apellido paterno" y "apellido materno".
- **Clave** (única, posiblemente, dentro de algún contexto). Por ejemplo el número de empleado podría ser único dentro del contexto de un empleado dado.
- **Monovaluada o multivaluada** (permiten grupos repetitivos). Por ejemplo un empleado dado podría tener varios números de teléfono, entonces "teléfono" puede ser una propiedad multivaluada.
- **Base o derivada.** Por ejemplo, la "cantidad total" de una parte en particular podría ser derivada como la suma de las cantidades de los envíos individuales de esa parte.

Representación gráfica de las entidades

En el modelo entidad relación los conjuntos de entidades se representan con un rectángulo dentro del cual se escribe el nombre de la entidad. Una entidad regular o fuerte es representada mediante un rectángulo con contorno grueso, mientras que para representar una entidad débil el contorno del rectángulo se forma con dos líneas delgadas. Los atributos se representan mediante su nombre en minúsculas unido con un guión al rectángulo de la entidad a la que pertenecen. Muchas veces los atributos, si hay muchos para cada entidad, se listan aparte del diagrama (fig. 6) para no complicarlo. (Paré, 2005)

FIG. 7 REPRESENTACIONES GRÁFICAS DE ENTIDADES

Relaciones

Una relación es una asociación entre las entidades. Por ejemplo, el "padre-hijo" es una relación entre dos personas "entidades". (CHEN, 1976) Es el elemento del modelo que permite relacionar en sí los datos del modelo. Por ejemplo, en el caso de que tengamos una entidad "Personas" y otra entidad "Trabajos", ambas se realizan ya que las personas trabajan y los trabajos son realizados por personas.

La representación gráfica de las *relaciones* se realiza con un rombo al que se le unen líneas que se dirigen a las entidades (CHEN, 1976), las *relaciones* tienen nombre (se suele usar un verbo). En el ejemplo anterior podría usarse como nombre de relación, trabajar:

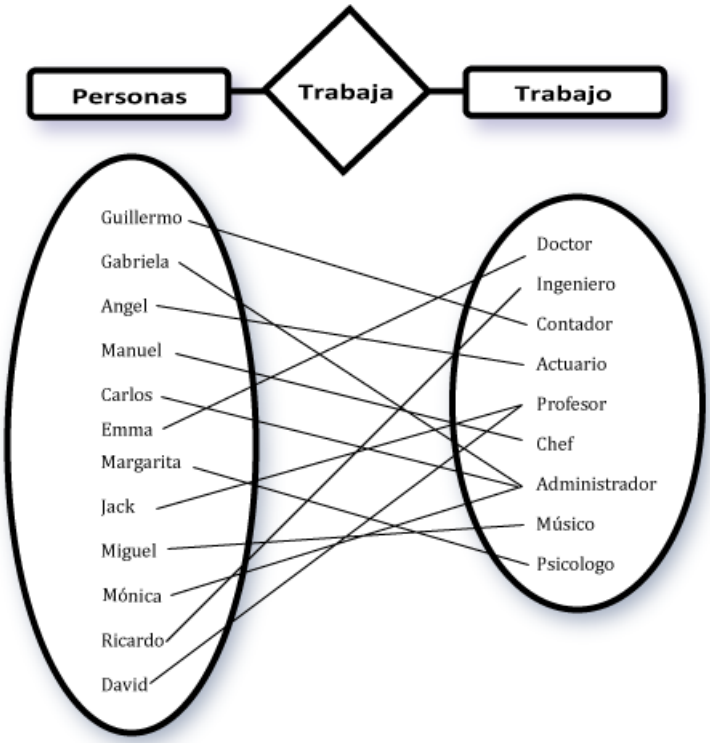


FIG. 8 EJEMPLO DE RELACIONES Y REPRESENTACIÓN GRÁFICA

Restricciones de cardinalidad

Los conjuntos de relaciones suelen tener restricciones como el "*cardinal de asignación*" que limita el número de entidades de un determinado conjunto con las que se puede asociar una entidad de otro conjunto. (CHEN, 1976)

Para un conjunto de relaciones binario entre dos conjuntos de entidades A y B, siendo X el número de entidades que se relacionan con B y Y el número de entidades que se relacionan con A, la "restricción de cardinalidad" se representa de la siguiente manera: (X: Y), donde X puede tomar valores de 0 a M y Y de 0 a N.

Existen muchas formas de representar gráficamente las *restricciones de cardinalidad*. En este trabajo utilicé el método "*pata de gallo*", desarrollado por C.W. Bachman, en lugar de utilizar la conectividad estilo Chen (1: M) (fig.9)

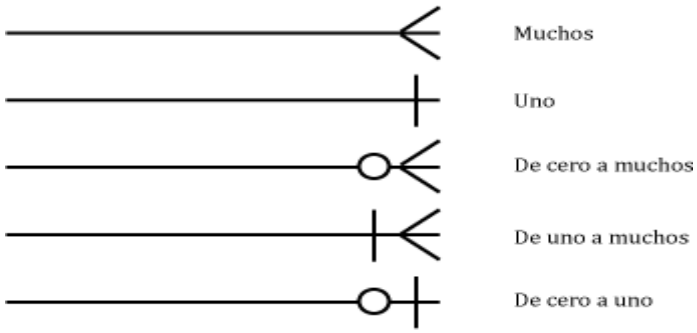


FIG. 9 RESTRICCIONES DE CARDINALIDAD TIPO PATA DE GALLO

En un Banco, por ejemplo, una persona puede tener varias cuentas y una misma cuenta puede pertenecer a varias personas; por tanto la relación que hay entre los conjuntos de entidades CUENTAS y CLIENTES es de muchos a muchos. En cambio, si consideramos a los empleados de una sucursal bancaria, observamos que en una sucursal trabajan varios empleados, pero que un empleado sólo puede trabajar en una sucursal; por tanto entre los conjuntos de entidades SUCURSALES y EMPLEADOS hay una relación uno a muchos. (Almería, 2011)(fig. 10)

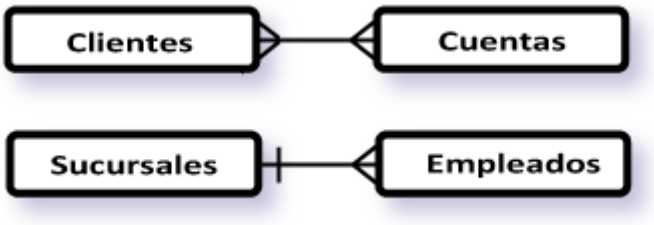


FIG. 10 EJEMPLO DE RESTRICCIONES

Identificador
Se trata de uno o más campos cuyos valores son únicos en cada ejemplar de una entidad.

Se indican subrayando el nombre del identificador.

Las condiciones para que un atributo sea considerado un buen identificador son:

1) Debe distinguir a cada ejemplar teniendo en cuenta las entidades que utiliza el modelo. No tiene que ser un identificador absoluto.
2) Todos los ejemplares de una entidad deben tener el mismo identificador.
3) Cuando un atributo es importante, aun cuando no tenga una entidad concreta asociada, entonces se trata de una entidad en sí y no de un atributo de las entidades *"is a".* Más adelante veremos en detalle este tipo de entidades *"is a"* (es un); baste por ahora decir que son aquellas que se descomponen en entidades especializadas. (fig. 11)

Ejemplo:

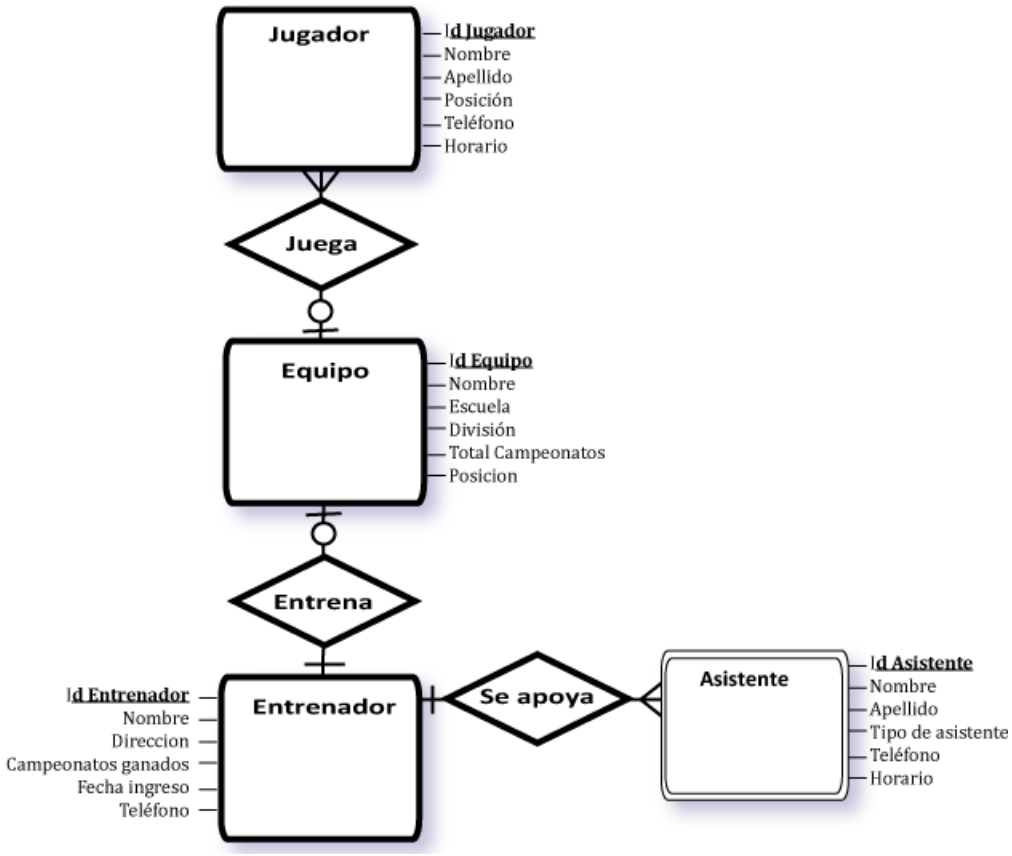


FIG. 11 EJEMPLO DEL MODELO ENTIDAD RELACIÓN

En el ejemplo de la figura 11, cada equipo cuanta con varios jugadores. Un jugador juega como mucho en un equipo y podría no jugar en ninguno. Cada entrenador entrena a un equipo (podría no entrenar a ninguno), el cual tiene un solo entrenador.

Modelo entidad relación extendido

El modelo *Entidad Relación Extendido* (EER por sus siglas en ingles: *Enhanced Entity-Relationship Model*) incorpora todos los conceptos introducidos por el modelo *ER* y además incluye los conceptos de subclase y superclase junto con los de especialización y generalización. Otro concepto incluido en este modelo es el de unión o categoría, que se utiliza para representar una colección de objetos unidos mediante diferentes tipos de entidades. Asociadas a este concepto se encuentran las importantes nociones de atributo y herencia relacionada. (Ramez Elmasri, 2003)

Relaciones ISA (es un) o relaciones de herencia

Son relaciones que indican tipos de entidades, es decir tendremos entidades que "son un" ("*is a*", en inglés) tipo de entidad.

Se utilizan para unificar entidades agrupándolas en una entidad más general (generalización) o bien para dividir una entidad general en entidades más específicas (especificación).

Se habla de generalización cuando partimos de una serie de entidades que, al ser estudiadas en detalle, resultan pertenecer a un mismo conjunto que será el que determine una nueva entidad.

Formalmente una entidad E es la generalización de las entidades de El, EP,. . E, si cada ocurrencia de E es también una ocurrencia de una y sólo una de las entidades de El, E2,. . E. (T. Teorey, 1986 Vol. 18 No. 2)

Una *jerarquía de generalización* se produce cuando una entidad (que llamamos la entidad genérica) se reparte por diferentes valores de un atributo común. Por ejemplo, la entidad EMPLEADO es una generalización del ingeniero, la secretaria y el técnico. El objeto de generalización (EMPLEADO) se llama un "*IS-A*" jerarquía exclusiva, ya que cada ocurrencia de la entidad EMPLEADO es una ocurrencia de una y sólo una de las entidades INGENIERO. (T. Teorey, 1986 Vol. 18 No. 2)

La *especialización* ocurre cuando partimos de una entidad que podemos dividir en subentidades para detallar atributos que varían en las mismas. Comparten clave con la superentidad y los atributos de la superclase se heredan en las subclases.

Por ejemplo, si algunos empleados son programadores (y todos los programadores son empleados), entonces podríamos decir que el tipo de entidad PROGRAMADOR es un subtipo del tipo de entidad EMPLEADO (o de manera equivalente, que el tipo de entidad EMPLEADO es un supertipo del tipo de entidad PROGRAMADOR). Todas las propiedades de los empleados se aplican automáticamente a los programadores, pero no sucede lo contrario (por ejemplo, los programadores podrían tener una propiedad "aptitud en lenguaje de programación" que no es aplicada a los empleados en general). En forma similar, los programadores participan automáticamente en todos los vínculos en los que participan los empleados, pero lo contrario no es cierto (por ejemplo, los programadores podrían pertenecer a alguna sociedad de computación profesional, mientras que los empleados en general no). Decimos que las propiedades y vínculos que se aplican al supertipo son heredados por el subtipo. (Date, 2001)

Un concepto importante asociado a las subclases es el de la *herencia de tipo*. Este concepto recuerda que el tipo de una entidad se define por los atributos que posee y los tipos de relaciones en los que participa. Ahora bien, debido a que una entidad en la subclase representa en el mundo real a la entidad de la superclase, esta debe poseer tanto valores de sus atributos específicos como valores de sus atributos como miembro de la superclase. Decimos que una entidad que es un miembro de una subclase *hereda* todos los atributos de la entidad como miembro de la superclase.

La entidad también hereda todas las relaciones en las que participa la superclase. (Ramez Elmasri, 2003) .En la práctica ambas se manejan de manera casi idéntica; la forma de representar una *Isa* es:

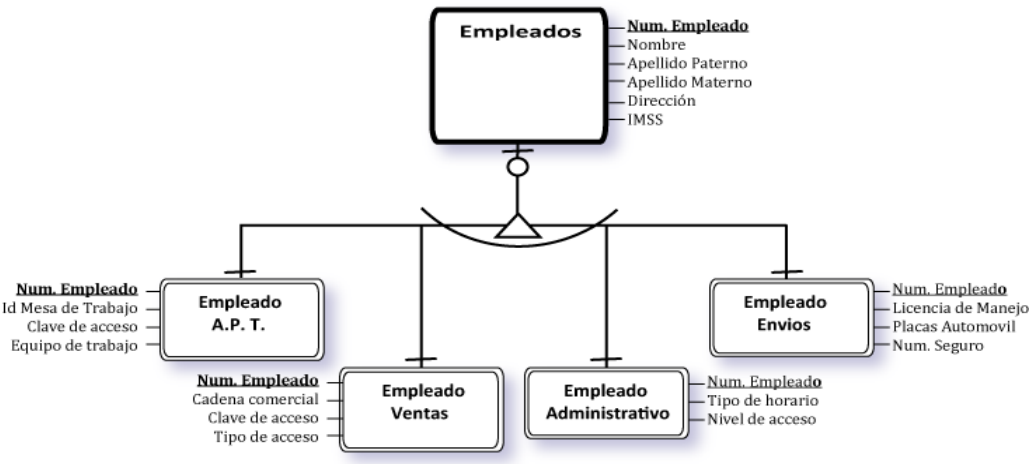


FIG. 12 RELACIÓN ISA

De cualquier modo, la cuestión de si tenemos una *generalización* o una *especialización* no es tan importante como el no errar las *cardinalidades*; las malas *cardinalidades* podrían provocar que el siguiente esquema del sistema (el esquema lógico) falle, y con él los demás esquemas y por lo tanto la base de datos en sí.

La ilustración anterior (Fig. 12), EMPLEADOS, es una generalización de empleados de APT, Ventas, Administrativos o Envíos. Incluso en este caso podría haber empleados de APT, Ventas, Administrativos o Envíos que no están relacionados con EMPLEADOS (la cardinalidad de Empleados es 0,1).

Exclusividad

En las relaciones ISA se puede indicar el hecho de que cada ejemplar sólo puede participar en una rama, entre varias, de una relación. Este hecho se marca con un arco entre las distintas relaciones. En las relaciones ISA se usa mucho. En la figura anterior, el personal sólo puede ser o Empleado de *APT* (*Almacén de Producto Terminado*), o de *Ventas*, o *Administración*, o de *Envíos*; una y sólo una de las cuatro cosas (esta es, por cierto, la forma más habitual de relación ISA). (T. Teorey, 1986 Vol. 18 No. 2)

En base a lo anterior, podemos tener los siguientes tipos de relaciones:

- **Relaciones de jerarquía solapada**. Indican que un ejemplar de la superentidad puede relacionarse con más de una subentidad (un empleado puede ser empleado de Ventas y Envíos). Ocurren cuando no hay dibujado un arco de exclusividad.
- **Relaciones de jerarquía exclusiva**. Indican que un ejemplar de la superentidad sólo puede relacionarse con una subentidad (un empleado no puede ser empleado de Ventas y Envíos). Ocurren cuando hay dibujado un arco de exclusividad.
- **Relaciones de jerarquía parcial.** Indican que hay ejemplares de la superentidad que no se relacionan con ninguna subentidad (hay personal que no pertenece a APT, o Ventas, o Administración, o Envíos). Se indican con cardinalidad mínima de cero en la superentidad.
- **Relaciones de jerarquía total.** Indican que todos los ejemplares de la superentidad se relacionan con alguna subentidad (no hay personal que no pertenezca a APT, o Ventas, o Administración, o Envíos). Se indican con cardinalidad mínima de uno en la superentidad.

Todos los posibles ejemplos de relaciones ISA atendiendo a la cardinalidad son los expuestos en la siguiente ilustración.

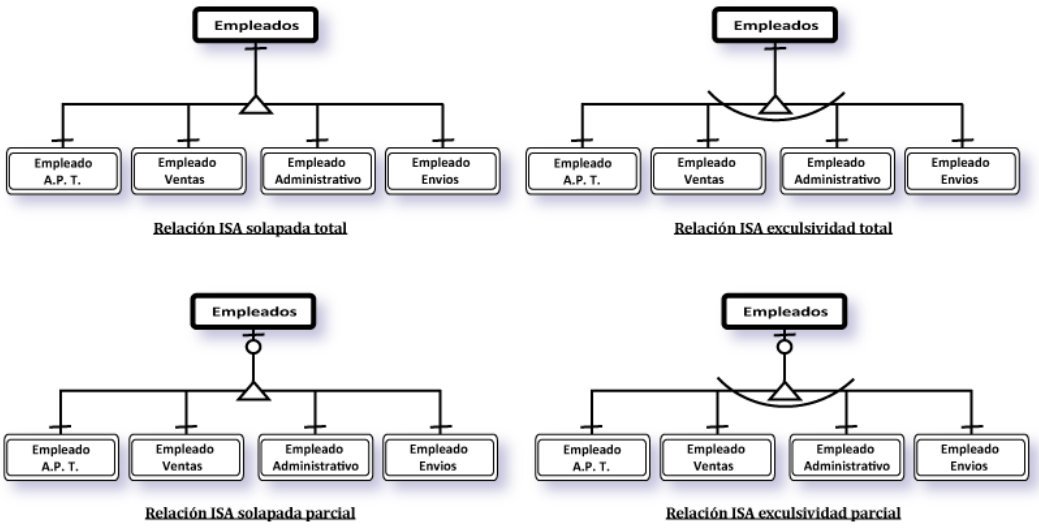


FIG. 13 TIPOS DE RELACIONES ISA

2.1.2. *MODELO RELACIONAL DE DATOS*

Edgar Frank Codd definió las bases del *Modelo Relacional* a finales de los 60 y lo publicó en 1970 en el articulo "*A Relational Model of data for Large Shared Data Banks*" (Un modelo relacional de datos para grandes bancos de datos compartidos). Al igual que el artículo de Peter Chen este es uno de los documentos más influyentes en la historia de la informática pues define las bases del llamado *Modelo Relacional de Bases de Datos*.

El *Modelo Relacional* se ocupa de tres aspectos principales de la información:

La estructura, manipulación y la integridad de los datos. (Date, 2001).

- La estructura de los datos se verá en siguiente sección.
- La manipulación de los datos se puede concebir en dos formas distintas y equivalentes: el álgebra relacional y el cálculo relacional (Date, 2001); de ellas surge el SQL que es el leguaje estándar de las bases de datos "relacionales". Explicaré tanto las formas como el lenguaje SQL en la sección 2.1.4 Lenguajes relacionales.
- La integridad de los datos. El término integridad se refiere a la exactitud o corrección de los datos en la base de datos. (Date, 2001). Codd propuso una serie de restricciones buscando que todos los valores que integran la base de datos, sean validos y únicos. Dichas restricciones se verán en la sección *Integridad de entidades, integridad referencial y claves externas.* Pág. 22

Estructura de los datos en el Modelo Relacional

Si pensamos en una relación como una tabla, entonces una *tupla* corresponde a una fila de dicha tabla, y un *atributo* a una columna; al número de *tuplas* se le llama *cardinalidad*[19] y al número de *atributos* se le denomina *grado*; un *dominio* es un conjunto de valores de donde se toman los valores de atributos específicos de relaciones específicas. (Date, 2001)

Las características inherentes más importantes de las *tuplas* y los *atributos* en una base de datos relacional son (Date, 2001):

- No puede haber dos *tuplas* iguales.
- El orden de las *tuplas* no es significativo.
- El orden de los *atributos* no es significativo.
- Cada *atributo* sólo puede tomar un valor en el *dominio* en el que está inscrito.

De manera formal

- **Dominio D**: Un dominio es un conjunto de datos (para abreviar, un tipo); posiblemente un tipo simple definido por el sistema como INTEGER o CHAR o -en forma más general- un tipo definido por el usuario como ID Pedido u Orden o Cliente. (Date, 2001)

- **Un esquema de Relación R** denotado por R (A1, A2,... An) es un conjunto de atributos R = {A1, A2,..., An}. Cada atributo Ai se asocia con un dominio Di; Ai indica el rol jugado por el Dominio en la relación R. Un esquema de relación describe una relación, y representa su nivel de interacción. El grado de la relación es el número de atributos de la misma. (Costa, 2001)

  Ejemplo*:* PEDIDO (ID Pedido, Orden, Cliente, Sucursal, Fecha Preparación,...)

- **Una relación R,** en el esquema de relación R (A1, A2,..., An), es un conjunto de tuplas "r = {t1, t2,..., tn}". Este "r" se conoce como esquema de extensión de R.

[19] El concepto de cardinalidad difiere del Modelo Relacional al Modelo Entidad Relación.

También se puede definir una relación desde el producto cartesiano de los dominios de su esquema de R:

r (R) = Dom (A1) X Dom (A2) X... X Dom (An)

Es importante tener en cuenta que un patrón de relación es casi invariante en el tiempo de modo que la ampliación representa los datos, en la base. En otras palabras, permite representar de forma completa el sistema de información, al relacionar los datos almacenados en los archivos de datos del sistema, con los procesos que transforman a éstos datos.

- **Clave Principal (Primary key):** Por definición, todas las *tuplas* de una relación son distintas. La *clave principal* o *clave primaria* marca uno o más atributos como identificadores de la tabla. De esa forma en esos atributos las filas de la tabla no podrán repetir valores ni tampoco dejarlos vacíos. (Date, 2001). Para identificar a un atributo que es clave principal este se subraya.

  Ejemplo: PEDIDO (**IDPedido**, Orden, Cliente, Sucursal, Fecha Preparación,...)

- **Un esquema de base de datos** es un conjunto de relaciones S = {R1, R2,..., Rn} y un conjunto de *restricciones de integridad* que se verán a siguiente sección.

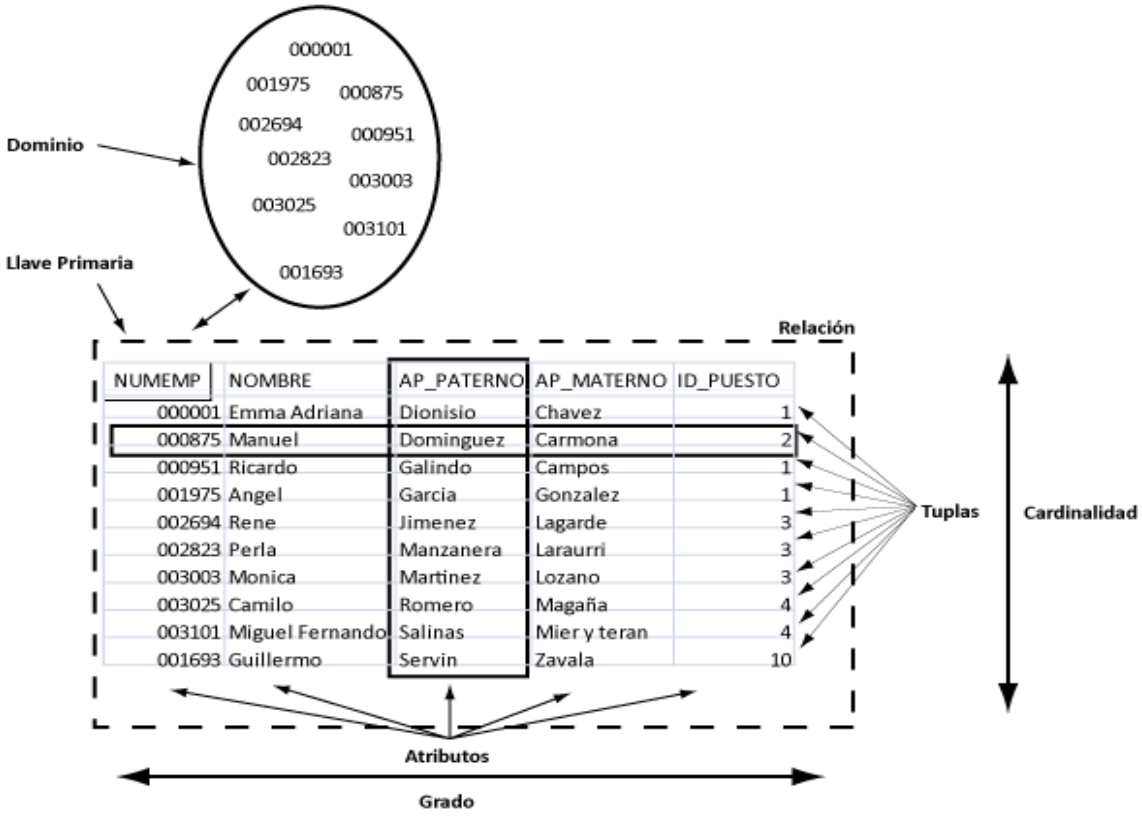


FIG. 14 CONCEPTOS DEL MODELO RELACIONAL

Integridad de entidades, integridad referencial y claves externas.

Integridad de entidades
Establece que ningún valor de *clave primaria* puede ser nulo. Esto se debe a que el valor de la clave primaria sirve para identificar la*s tuplas* de una relación, y si la *clave primaria* puede tener valores nulos, no podríamos identificar algunas *tuplas*. (Costa, 2001)

Integridad Referencia
Se trata de una restricción entre dos relaciones expresadas en la que se verifica que la información utilizada en una *tupla* para referirse a otra *tupla* sea válida. Esto se logra mediante el análisis de la relación que una *clave principal,* de la tabla que relaciona (llamada tabla principal), y una *clave externa* (también llamada secundaria y foránea) tienen con uno o más atributos de la tabla secundaria. Esta restricción implica un orden en la creación o destrucción de las entidades porque no se puede crear una *tupla*, correspondiente a la tabla secundaria, si la *tupla* que la relaciona en la tabla principal no existe ya. (Date, 2001)

Es decir, si hay una tabla de *Empleados* en la que cada fila es un empleado, existirá un atributo *IdPuesto* que indicará el código del puesto que desempeña el empleado y que estará relacionado con una tabla de *Puestos*, en la que el atributo *IdPuesto* de dicha tabla es la *clave principal*. De hecho no se podrá incluir un *IdPuesto* en la tabla *Empleados* que no esté en *Puestos*; eso es lo que prohíbe la *integridad referencial*.

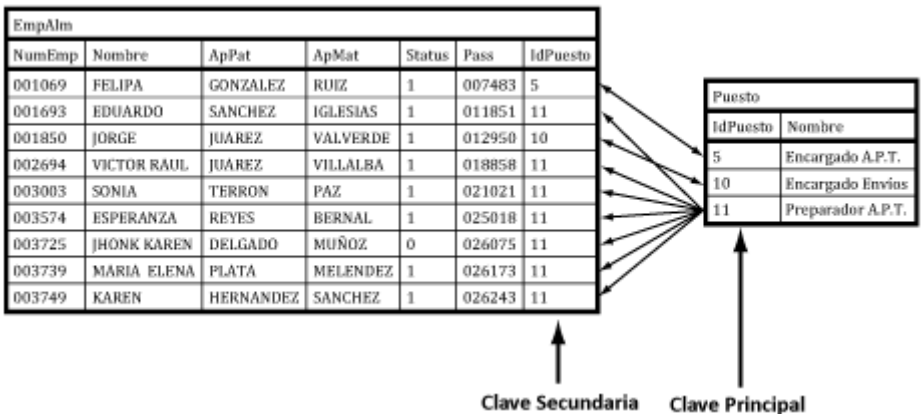


FIG. 15 INTEGRIDAD REFERENCIAL

Esto causa problemas en las operaciones de borrado y modificación de registros pues si se ejecutan esas operaciones sobre la tabla principal (si se modifica o borra un Puesto) quedarán filas en la tabla secundaria con la *clave externa* haciendo referencia a un valor que ya no existe en la tabla principal.

Para solventar esta situación los *SGBD relacionales* (tema que se verá más en detalle en la sección 0) hacen uso de las siguientes opciones:

- Prohibir la operación (*no action*).
- Transmitir la operación en cascada (*cascade*). Es decir si se modifica o borra un Puesto; también se modificarán o barrarán los Empleados relacionados con él.
- Colocar nulos (*set null*) Las referencias al cliente en la tabla de alquileres se colocan como nulos (es decir, Empleados sin puesto).

Usar el valor por defecto (*default*). Se colocan un valor por defecto en las claves externas relacionadas. Este valor se indica al crear la tabla (*opción default*).

Claves externas

Un conjunto de atributos del esquema de la relación R_1 es una *clave externa* de R_1 si satisface estas condiciones:

- Los atributos de la *clave externa* tienen el mismo *dominio* que los atributos de la *clave primaria* de otro esquema de relación R_2; se dice que los *atributos* de la *clave externa* hacen referencia o se refieren a la relación R_2.
- Un valor de *clave externa* en una *tupla* T_1 del estado actual de R_1 es el valor de *clave primaria* en alguna *tupla* T_2 del estado actual de R_2, o bien es nulo. Si no es nulo, diremos que la *tupla* T_1 hace referencia a la *tupla* T_2. R_1 será la *relación referenciante*, y R_2 la *relación referenciada*.

Una *clave externa* también puede hacer referencia a su propia relación (caso de supervisor y empleado). Es decir, cuando una entidad externa provee datos al sistema, debe existir un flujo de datos saliendo de la entidad y en dirección al sistema. Y cuando una entidad externa recibe datos del sistema, debe existir un flujo de datos que viene del sistema y termina en la entidad externa. Las entidades externas pueden duplicarse si fuera necesario darle claridad al diseño y evitar largos vectores (que representan a los flujos de datos), o bien para evitar que se produzcan muchos entrecruzamientos entre ellos.

En una base de datos con muchas relaciones suele haber muchas *restricciones de integridad referencial* que surgen de las relaciones representadas por los esquemas de relación.

No hemos incluido hasta ahora a las restricciones llamadas "***restricciones de integridad semántica"*** ya que estas no son parte directa del *Modelo Relacional.* El *SGBD* es responsable de este tipo de restricciones (del tipo: "un alumno no puede estar matriculado con más de 80 créditos") que necesitan un lenguaje de especificación de restricciones de propósito general. También hay "***restricciones de transición"*** que tratan con cambios de estado de la base de datos ("el sueldo de un empleado sólo puede incrementarse"). Estas restricciones se especifican con reglas de actividad y disparadores o *triggers* y procedimientos almacenados[20].

[20] Los procedimientos almacenados y activados, son procedimientos pre compilados que pueden ser llamados desde programas de aplicación. Dichos procedimientos pueden ser considerados de manera lógica como una extensión al SGBD(en un sistema cliente-servidor, a menudo se mantendrán y ejecutarán en el sitio del servidor).

2.1.3. *SISTEMAS DE GESTIÓN DE BASES DE DATOS*

El *Sistema de Gestión de Base de Datos SGBD ó DMBS* (por sus siglas en inglés, Data Base Management System), es básicamente una capa de software intermedia, entre la base de datos física y los usuarios del sistema, cuya finalidad general consiste en proporcionar una interfaz de usuario.

Podemos definir la interfaz de usuario como un límite en el sistema debajo del cual todo es invisible para el usuario. Por lo tanto, por definición, la interfaz de usuario se encuentra en el nivel externo; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base a peticiones expresadas en términos de un nivel más alto de percepción. (Date, 2001)

Las principales actividades que realiza un SGBD son: la definición de una base de datos (especificando tipos de datos que se almacenan), la construcción de una base de datos (almacenamiento de datos reales) y la manipulación de datos (sobre todo añadir, eliminar, buscar datos). (fig. 16)

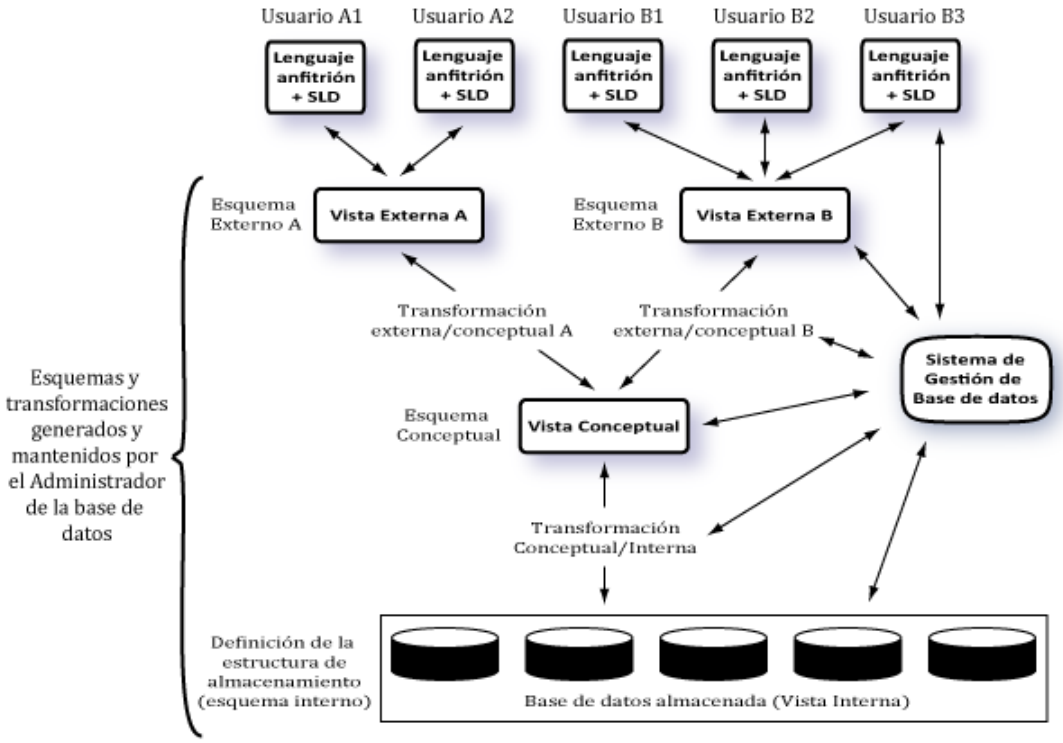


FIG. 16 ARQUITECTURA DETALLADA DEL SGBD

Analicemos ahora las funciones del SGBD con un poco más de detalle. Dichas funciones comprenderán por lo menos el manejo de todas las siguientes: (fig. 17)

- **Definición de datos.** El SGBD debe ser capaz de aceptar definiciones de datos (esquemas externos, el esquema conceptual, el esquema interno y todas las transformaciones respectivas) en la forma fuente y convertirlas a la forma objeto correspondiente. En otras palabras, el SGBD debe incluir entre sus componentes un procesador DDL[21] o compilador DDL, para cada uno de los diversos DDLs. El SGBD también debe "entender" definiciones DDL, en el sentido que, por ejemplo, "entienda" que los registros externos EMPLEADO incluyen un campo SALARIO; entonces,

[21] DDL Lenguaje de definición de datos (en inglés Data Definition Language), es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

debe poder utilizar este conocimiento para analizar y responder a las peticiones de manipulación de datos (por ejemplo: "Obtener todos los empleados con salario < $50,000"). (Date, 2001)

- **Manipulación de datos.** El SGBD debe ser capaz de manejar peticiones para recuperar, actualizar o eliminar datos existentes en la base de datos o agregar nuevos datos a ésta. En otras palabras, el SGBD debe incluir un componente procesador o compilador DML[22]. (Date, 2001)

- **Optimización y ejecución.** Las peticiones DML, deben ser procesadas por el componente optimizador, cuya finalidad es determinar una forma eficiente de implementar la petición. Las peticiones optimizadas se ejecutan entonces bajo el control del administrador en tiempo de ejecución. (Date, 2001)

- **Seguridad e integridad de los datos.** Un SGBD debe proveer varios mecanismos para garantizar la seguridad de los datos almacenados. Debe vigilar las peticiones del usuario y rechazar todo intento de violar las restricciones de seguridad y de integridad[23] definidas por el administrador de la base de datos. Estas tareas pueden realizarse durante el tiempo de compilación, de ejecución o entre ambos. (Date, 2001)

  Asegura, además, un comportamiento atómico de una secuencia de acciones (que se lleva a cabo con éxito completo, o se cancela). Por ejemplo, las transacciones pueden llevar a cabo bloqueos, sobre los registros que vayan a utilizar, impidiendo a otros usuarios la recuperación o actualización de los elementos bloqueados, pudiéndose así evitar inconsistencias en el acceso concurrente.

- **Concurrencia y recuperación de datos.** El SGBD debe imponer ciertos controles de recuperación y concurrencia. (Date, 2001) Una base de datos debe permitir a varios usuarios el acceso simultáneo a la misma información y debe proporcionar interfaces de acceso correspondientes a los diferentes tipos de usuarios que pueden hablar con él.

  Con respecto a los riesgos asociados a los fallos de disco, el SGBD utiliza un mecanismo de registro para la recuperación de una base de datos de forma automática a partir de una versión de copia de seguridad del periódico y movimientos.

- **Diccionario de datos**. El SGBD debe proporcionar una función de diccionario de datos.

  Este diccionario puede ser visto como una base de datos por derecho propio (aunque una base de datos del sistema más que como una base de datos del usuario). El diccionario contiene "datos acerca de los datos"(en ocasiones llamados metadatos o descriptores); es decir, definiciones de otros objetos del sistema, en lugar de simples "datos en bruto". En particular, todos los diversos esquemas y transformaciones (externos, conceptuales, etcétera) y todas las diversas restricciones de seguridad y de integridad, serán almacenadas en el diccionario, tanto en forma fuente como objeto. Un diccionario extenso incluirá además mucha información adicional; mostrará por ejemplo qué programas utilizan qué partes de la base de datos, qué usuarios necesitan qué informes, etcétera. El diccionario podría incluso —y de hecho, debería— estar integrado dentro de la base de datos que define, e incluir por lo tanto su propia definición. En realidad, debe ser

[22] DML Lenguaje de Manipulación de Datos (en ingles Data Manipulation Language) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.
[23] Ver sección 2.1.2-Integridad de entidades, integridad referencial y claves externas.

posible consultar el diccionario del mismo modo que cualquier otra base de datos, de manera que, por ejemplo, sea posible saber qué programas o usuarios se podrían ver afectados por un cambio propuesto al sistema. (Date, 2001)

- **Rendimiento**. Sobra decir que el SGBD debe realizar todas las tareas antes identificadas de la manera más eficiente posible.

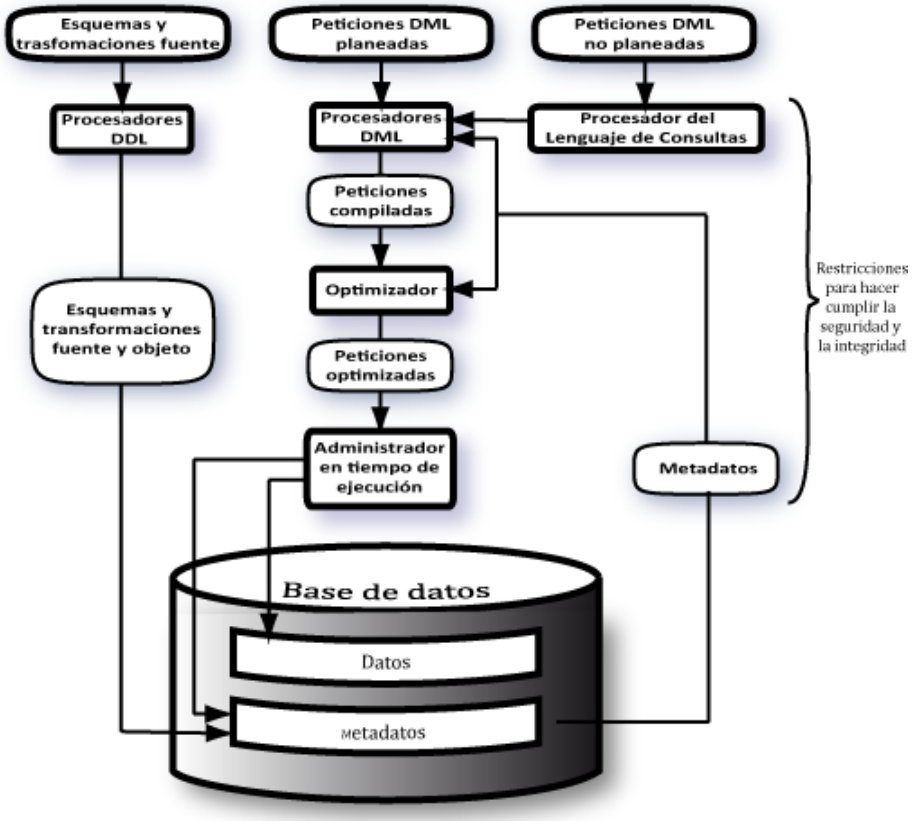


FIG. 17 FUNCIONES Y COMPONENTES PRINCIPALES DEL SGBD

Funcionamiento del SGBD

Los datos son responsabilidad del SGBD, por lo que cualquier acceso debe ser realizado por éste. Lógicamente el SGBD va a acabar comunicándose con el *Sistema Operativo* ya que el acceso a los ficheros de datos implica utilizar funciones del *Sistema Operativo*.

El esquema siguiente presenta el funcionamiento típico de un SGBD:

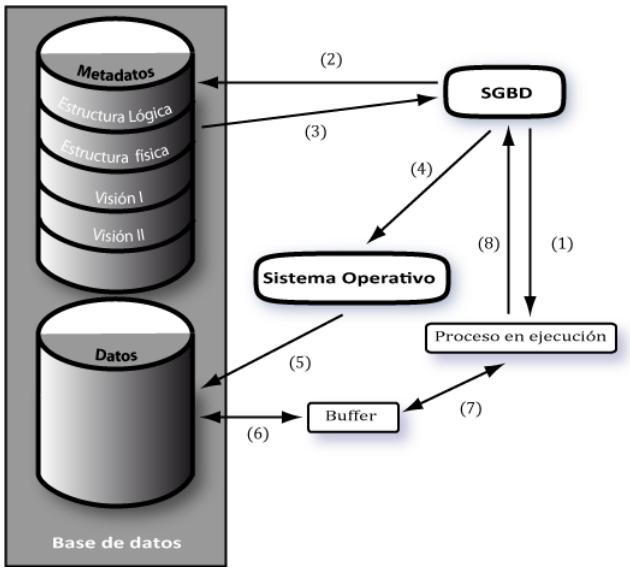


FIG. 18 ESQUEMA COMPLETO DE LA COMUNICACIÓN ENTRE PROCESOS DE USUARIO, SGBD, Y SISTEMA OPERATIVO

1. El proceso lanzado por el usuario llama al SGBD indicando la porción de la base de datos que se desea tratar.
2. El SGBD traduce la llamada a términos del esquema lógico de la base de datos. Accede al esquema lógico comprobando derechos de acceso y la traducción física (normalmente los metadatos se guardan una zona de memoria global y no en el disco).
3. El SGBD obtiene el esquema físico.
4. El SGBD traduce la llamada a los métodos de acceso del *Sistema Operativo* que permiten acceder realmente a los datos requeridos.
5. El *Sistema Operativo* accede a los datos tras traducir las órdenes dadas por el SGBD
6. Los datos pasan del disco a una *memoria intermedia* o *buffer*, ahí se almacenarán los datos según se vayan recibiendo.
7. Los datos pasan del *buffer* al área de proceso en ejecución del usuario. Los pasos 6 y 7 se repiten hasta que se envíe toda la información.
8. En el caso de que haya errores en cualquier momento del proceso, el SGBD devuelve indicadores en los que manifiesta si ha habido errores o advertencias a tener en cuenta. Esto se indica al área de comunicaciones del proceso de usuario. Si las indicaciones son satisfactorias, los datos de serán utilizables por el proceso de usuario.

2.1.4. *LENGUAJES RELACIONALES*

Formalmente hay dos clases de lenguajes para la definición y el manejo del Modelo Relacional: Lenguajes Algebraicos y Lenguajes Predicativos. Éstos idiomas son estrictamente equivalentes en términos de poder expresivo (cualquier aplicación que se pueda expresar en uno de éstos idiomas se puede expresar en el otro). (Codd E. E., 1990)

El álgebra y el cálculo son alternativos entre sí. La diferencia principal entre ellos es que mientras el álgebra proporciona un conjunto de operadores explícitos —juntar, unión, proyectar, etcétera— que pueden usarse para indicar al sistema cómo construir cierta relación deseada a partir de relaciones dadas, el cálculo simplemente proporciona una notación para establecer la definición de esa relación deseada en términos de dichas relaciones dadas. (Date, 2001)

Álgebra relacional

El álgebra relacional consiste en un conjunto de operadores que operan sobre las relaciones, generando así nuevas relaciones. Por tanto, es posible generar nueva información a partir de relaciones iniciales y una composición secuencial de operadores. Los operadores más comunes se muestran a continuación y se pueden clasificar en tres categorías (Codd E. E., 1990)

- **Los operadores unarios**: asignación, selección y proyección
- **Los operadores binarios de trabajo bajo un mismo esquema**: unión, intersección, diferencia
- **Los operadores binarios que trabajan bajo distintos esquemas**: unión, producto cartesiano, theta-join, la división

Para ejemplificar los distintos operadores utilizaremos las siguientes relaciones o tablas EmpAlm (Empleados Almacén), EmpVen (Empleados Ventas) y Puesto (Puesto Empleado) que se muestran a continuación:

EmpAlm						
NumEmp	Nombre	ApPat	ApMat	IdPuesto	Status	Pass
001069	FELIPA	GONZALEZ	RUIZ	15	1	007483
001693	EDUARDO	SANCHEZ	IGLESIAS	5	1	011851
001850	JORGE	JUAREZ	VALVERDE	10	1	012950
002694	VICTOR RAUL	JUAREZ	VILLALBA	11	1	018858
003003	SONIA	TERRON	PAZ	11	1	021021
003574	ESPERANZA	REYES	BERNAL	11	1	025018
003725	JHONK KAREN	DELGADO	MUÑOZ	11	0	026075
003739	MARIA ELENA	PLATA	MELENDEZ	15	1	026173
003749	KAREN	HERNANDEZ	SANCHEZ	11	1	026243

TABLA 1 TABLA O RELACIÓN EMPALM (EMPLEADOS DE ALMACÉN)

EmpVen						
NumEmp	Nombre	ApPat	ApMat	IdPuesto	Status	Pass
002050	VICTOR	RAMIREZ	GONZALEZ	20	1	001720
002274	MANUEL	ROMERO	MIER	20	1	000666
001069	FELIPA	GONZALEZ	RUIZ	15	1	007483
003739	MARIA ELENA	PLATA	MELENDEZ	15	1	026173
001850	JORGE	SANCHEZ	VALVERDE	2	1	013350

TABLA 2 TABLA O RELACIÓN EMPVEN (EMPLEADOS DE VENTAS)

Puesto		
IdPuesto	Nivel	DescPuesto
2	1	JEFE DE VENTAS
10	1	JEFE DE ALMACEN DE PROD. TERM.
11	2	PREPARADOR ALMACEN DE PROD. TE
15	2	PREPARADOR DE VENTAS DE ALM.
20	2	AUXILIAR VENTAS

TABLA 3 TABLA O RELACIÓN PUESTO (PUESTO EMPLEADO)

Los operadores unarios

Para ejemplificar los operadores unarios utilizaré la tabla *EmpAlm*

Asignación: R (A1,..., An) <- Seleccione expresión. La asignación permite guardar el resultado de una frase de búsqueda, o cambiar el nombre de una relación y sus atributos.

$R \leftarrow (001069, FELIPA, GONZALEZ, RUIZ, 5,1,007483)$

Selección (σ): Permite seleccionar un subconjunto de tuplas de una relación (R), que contiene todas aquellas tuplas que cumplan las condiciones P a partir de conectores lógicos "y", "o", y "no", o condiciones simples, esto es: $\sigma_P(R)$

Ejemplo: $\sigma_{ApPat=JUAREZ}(\text{EmpAlm})$

001850	JORGE	JUAREZ	VALVERDE	10	1	012950
002694	VICTOR RAUL	JUAREZ	VILLALBA	11	1	018858

Selecciona todas las tuplas que contengan *JUAREZ* como apellido materno en la relación *EmpAlm*.

Proyección (Π): Proyección A1,..., An (R) Permite extraer columnas (atributos) de una relación, dando como resultado un *subconjunto vertical* de atributos de la relación, esto es: $\Pi_{A_1,A_2,\ldots,A_n}$

Donde $A_1, A_2, \ldots, A_n$ son atributos de la relación R.

Ejemplo:

$\Pi_{NumEmp,Nombre}(\text{EmpAlm})$

NumEmp	Nombre
001069	FELIPA
001693	EDUARDO
001850	JORGE
002694	VICTOR RAUL
003003	SONIA
003574	ESPERANZA
003725	JHONK KAREN
003739	MARIA ELENA
003749	KAREN

Selecciona los atributos *NumEmp* y *Nombre* de la relación *EmpAlm*, mostrados como un subconjunto de la relación.

Los operadores binarios del mismo patrón

Los tres operadores de operación ajustados a las relaciones *EmpAlm* y *EmpVen*.

*Unión(*R U S*):* Produce una nueva relación con las tuplas RS mostrando las tuplas que están en R, o en S, o en ambas. R y S deben ser *uniones compatibles* (duplicados se eliminan).

Ejemplo: $EmpVen \cup EmpAlm$

001069	FELIPA	GONZALEZ	RUIZ	15	1	007483
001693	EDUARDO	SANCHEZ	IGLESIAS	5	1	011851
001850	JORGE	JUAREZ	VALVERDE	10	1	012950
002694	VICTOR RAUL	JUAREZ	VILLALBA	11	1	018858
003003	SONIA	TERRON	PAZ	11	1	021021
003574	ESPERANZA	REYES	BERNAL	11	1	025018
002050	VICTOR	RAMIREZ	GONZALEZ	20	1	001720
002274	MANUEL	ROMERO	MIER	2	1	000666
003739	MARIA ELENA	PLATA	MELENDEZ	15	1	026173
001850	JORGE	SANCHEZ	VALVERDE	20	1	013350
003725	JHONK KAREN	DELGADO	MUÑOZ	11	0	026075
003749	KAREN	HERNANDEZ	SANCHEZ	11	1	026243

*Diferencia(*R - S*)*: Entrega todas aquellas tuplas que están en R, pero no en S. R y S deben ser uniones compatibles, la diferencia no es conmutativa.

Ejemplo: $EmpVen - EmpAlm$

002050	VICTOR	RAMIREZ	GONZALEZ	20	1	001720
002274	MANUEL	ROMERO	MIER	2	1	000666
001850	JORGE	SANCHEZ	VALVERDE	20	1	013350

Intersección ( ): Corresponde al conjunto de todas las tuplas que están en R y en S, siendo R y S uniones compatibles.

La intersección de dos relaciones se puede especificar en función de otros operadores básicos:

. Por lo tanto la intersección es considerada derivada o secundaria.

$$EmpVen \cup EmpAlm = EmpVen - (EmpVen - EmpAlm)$$

001069	FELIPA	GONZALEZ	RUIZ	15	1	007483
003739	MARIA ELENA	PLATA	MELENDEZ	15	1	026173

Los operadores binarios de los diferentes modelos

Utilizaré las relaciones *EmpVen y Puestos*.

Producto cartesiano: R X S: El producto cartesiano entrega una relación cuyo esquema corresponde a una combinación de todas las tuplas de R con cada una de las tuplas de S, y sus atributos corresponden a los de R seguidos por los de S. Tiene en cuenta que el número de elementos del producto cartesiano es el producto de las cardinalidades de las relaciones R y S

Ejemplo:

$EmpVenXPuestos$

2050	VICTOR	RAMIREZ	GONZALEZ	20	1	1720	2	1	JEFE DE VENTAS
2051	VICTOR	RAMIREZ	GONZALEZ	20	1	1720	10	1	JEFE DE ALMACEN DE PROD. TERM.
2052	VICTOR	RAMIREZ	GONZALEZ	20	1	1720	11	2	PREPARADOR ALMACEN DE PROD. TE
2053	VICTOR	RAMIREZ	GONZALEZ	20	1	1720	15	2	PREPARADOR DE VENTAS DE ALM.
2054	VICTOR	RAMIREZ	GONZALEZ	20	1	1720	20	2	AUXILIAR VENTAS
2274	MANUEL	ROMERO	MIER	2	1	666	2	1	JEFE DE VENTAS
2275	MANUEL	ROMERO	MIER	2	1	666	10	1	JEFE DE ALMACEN DE PROD. TERM.
2276	MANUEL	ROMERO	MIER	2	1	666	11	2	PREPARADOR ALMACEN DE PROD. TE
2277	MANUEL	ROMERO	MIER	2	1	666	15	2	PREPARADOR DE VENTAS DE ALM.
2278	MANUEL	ROMERO	MIER	2	1	666	20	2	AUXILIAR VENTAS
1069	FELIPA	GONZALEZ	RUIZ	15	1	7483	2	1	JEFE DE VENTAS
1070	FELIPA	GONZALEZ	RUIZ	15	1	7483	10	1	JEFE DE ALMACEN DE PROD. TERM.
1071	FELIPA	GONZALEZ	RUIZ	15	1	7483	11	2	PREPARADOR ALMACEN DE PROD. TE
1072	FELIPA	GONZALEZ	RUIZ	15	1	7483	15	2	PREPARADOR DE VENTAS DE ALM.
1073	FELIPA	GONZALEZ	RUIZ	15	1	7483	20	2	AUXILIAR VENTAS
3739	MARIA ELENA	PLATA	MELENDEZ	15	1	26173	2	1	JEFE DE VENTAS
3740	MARIA ELENA	PLATA	MELENDEZ	15	1	26173	10	1	JEFE DE ALMACEN DE PROD. TERM.
3741	MARIA ELENA	PLATA	MELENDEZ	15	1	26173	11	2	PREPARADOR ALMACEN DE PROD. TE
3742	MARIA ELENA	PLATA	MELENDEZ	15	1	26173	15	2	PREPARADOR DE VENTAS DE ALM.
3743	MARIA ELENA	PLATA	MELENDEZ	15	1	26173	20	2	AUXILIAR VENTAS
1850	JORGE	SANCHEZ	VALVERDE	20	1	13350	2	1	JEFE DE VENTAS
1851	JORGE	SANCHEZ	VALVERDE	20	1	13350	10	1	JEFE DE ALMACEN DE PROD. TERM.
1852	JORGE	SANCHEZ	VALVERDE	20	1	13350	11	2	PREPARADOR ALMACEN DE PROD. TE
1853	JORGE	SANCHEZ	VALVERDE	20	1	13350	15	2	PREPARADOR DE VENTAS DE ALM.
1854	JORGE	SANCHEZ	VALVERDE	20	1	13350	20	2	AUXILIAR VENTAS

Unión natural (R⋈S) La operación unión natural en el álgebra relacional es la que permite reconstruir las tablas originales previas al proceso de normalización. Consiste en combinar la proyección, la selección y el producto cartesiano en una sola operación, donde la condición θ es la igualdad Clave Primaria = Clave Externa (o Foránea), y la proyección elimina la columna duplicada (clave externa).

Expresada en las operaciones básicas, queda:

$$R \bowtie S = \Pi_{A1,A2...An}(\sigma_{\theta}(R \times S))$$

Ejemplo

$$EmpVen \triangleright\triangleleft Puestos = \Pi_{A1..An}\left(\sigma_{EmpVen.IdPuesto=Puesto.IDPuesto}(EmpVenXPuestos)\right)$$

$$A1,..An = NumEmp, Nombre, ApPat, ApMat, IdPuesto, Nivel, DescPuesto, Status, Pass$$

Paso1:

$EmpVenXPuestos$

Ver Producto cartesiano

Paso 2:

$\sigma_{EmpVen.IdPuesto=Puesto.IDPuesto}(EmpVenXPuestos)$:

2054	VICTOR	RAMIREZ	GONZALEZ	**20**	1	1720	**20**	2	AUXILIAR VENTAS
2274	MANUEL	ROMERO	MIER	**2**	1	666	**2**	1	JEFE DE VENTAS
1072	FELIPA	GONZALEZ	RUIZ	**15**	1	7483	**15**	2	PREPARADOR DE VENTAS DE ALM.
3742	MARIA ELENA	PLATA	MELENDEZ	**15**	1	26173	**15**	2	PREPARADOR DE VENTAS DE ALM.
1854	JORGE	SANCHEZ	VALVERDE	**20**	1	13350	**20**	2	AUXILIAR VENTAS

Paso 3:

$\Pi_{NumEmp,Nombre,ApPat,ApMat,IdPuesto,Nivel,DescPuesto,Status,Pass}$

2054	VICTOR	RAMIREZ	GONZALEZ	20	2	AUXILIAR VENTAS	1	1720
2274	MANUEL	ROMERO	MIER	2	1	JEFE DE VENTAS	1	666
1072	FELIPA	GONZALEZ	RUIZ	15	2	PREPARADOR DE VENTAS DE ALM.	1	7483
3742	MARIA ELENA	PLATA	MELENDEZ	15	2	PREPARADOR DE VENTAS DE ALM.	1	26173
1854	JORGE	SANCHEZ	VALVERDE	20	2	AUXILIAR VENTAS	1	13350

Una reunión theta (θ-Join) de dos relaciones es equivalente a:

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

Donde la condición θ es libre.

División: R / S

Supongamos que tenemos dos relaciones A(x, y) y B (y) donde el dominio de "y" en A y B, es el mismo. El operador división A / B retorna todos los distintos valores de "x" tales que para todo valor "y" en B existe una tupla $\langle x, y \rangle$ en A.
Ejemplo:

ConfCli	
NumCli	IdConf
CMX001	1
CMX001	2
WMX001	1
WMX001	2
WMX001	3
WMX001	4
LIV001	1
IV001	3

Confirmación
IdConf
1
2
3

ConfCli/ Confirmación = WMX001

Cálculo relacional

El cálculo relacional está basado en una rama de la lógica matemática denominada cálculo de predicados. El concepto de cálculo relacional -es decir, un cálculo de predicados aplicado específicamente a las bases de datos relacionales- fue propuesto por primera vez por Codd quien también elaboró un lenguaje basado explícitamente en ese cálculo al que se llama Sublenguaje de datos ALPHA. (Date, 2001)

Las variables de alcance son una característica fundamental del cálculo que, resumiendo, son aquellas que "abarcan" a alguna relación especificada; es decir, una variable cuyos valores permitidos son tuplas de esa relación. Por lo tanto, si la variable de alcance X abarca a la relación *EmpAlm*, entonces, en cualquier momento dado, la expresión "X" denota alguna tupla "t" de *EmpAlm*. Por ejemplo, la consulta "Obtener el Id del empleado (NumEmp) cuya contraseña (Pass) es = '012950' " podría expresarse en QUEL[24] como sigue:

RANGE OF X IS EmpProd;
RETRIEVE (X.NumEmp) WHERE X. Pass = "012950";

Aquí, la variable de alcance es X y abarca cualquier relación que sea el valor actual de la variable *EmpProd* (la instrucción RANGE es una definición de esa variable de alcance). Por lo tanto, la instrucción RETRIEVE puede ser parafraseada como: "Para cada valor posible de la variable X, recupera el componente *EmpProd* # de ese valor si y sólo si el componente Contraseña (*PASS*) tiene el valor "012950".

De la siguiente tabla el valor obtenido sería = 001850

EmpProd						
NumEmp	Nombre	ApPat	ApMat	IdPuesto	Status	Pass
001069	FELIPA	GONZALEZ	RUIZ	15	1	007483
001693	EDUARDO	SANCHEZ	IGLESIAS	5	1	011851
001850	JORGE	JUAREZ	VALVERDE	10	1	012950
002694	VICTOR RAUL	JUAREZ	VILLALBA	11	1	018858
003003	SONIA	TERRON	PAZ	11	1	021021

Debido a su dependencia sobre las variables de alcance cuyos valores son tuplas (y para distinguirlo del cálculo de dominios, el cálculo relacional original se conoce ahora como cálculo de tuplas.

Cálculo de Tuplas

El *cálculo relacional de tuplas TRC* (en ingles Tuple Relational Calculus) se basa en la especificación de un número de variables de tuplas que se extienden sobre una relación (los valores son tuplas de la relación).

Una consulta formal de la *TRC* se expresa como {t | P (t)}, donde t es una variable tupla y P (t) es una expresión que incluya la variable *t.*

El resultado de la consulta se considera como un conjunto de tuplas t donde el predicado P es cierto para t.

Varias variables tupla aparecen en una fórmula. Una variable puede estar libre o unida a una fórmula. Una variable tupla libre es aquella que no se cuantifica por un ∃ o un ∀. Por ejemplo en la fórmula:

t ∈ DEPARTAMENTO ∧ ∃ e ∈ EMPLEADO (e . Nombre = Juan ∧ e. Deptid = t. Deptid)

[24] QUEL es un lenguaje de base de datos de acceso relacional, similar en muchos aspectos a SQL. Fue creado en la Universidad de California, Berkeley, basado en ALPHA.

La variable *t* es libre y la variable *e* es obligada.

Una fórmula de *TRC* se construye a partir de una estructura o fórmula atómica. Construyéndose a partir de formulas elementales concatenadas entre sí mediante un operador. Siguiendo las siguientes normas.

Si *P* es una fórmula, entonces también lo son ¬*P* y (*P*)

Si P_1 y P_2 son fórmulas, entonces también lo son $P_1 \wedge P_2$, $P_1 \vee P_2$ y $P_1 \Rightarrow P_2$

Si *P(t)* es una fórmula que contiene una variable *t* libre entonces $\exists t \in r(P(t))$ y $\forall t \in r(P(t))$ son también fórmulas.

Ejemplo de consultas básicas en el cálculo relacional de tuplas:

Considere las siguientes relaciones:

- EMPLEADO (#NumEmp, Nombre, FecNacimiento, Dirección, Salario, idDepto)
- DEPARTAMENTO (#idDepto, Nom_Depto, Oficina, Cod_Admi)
- PROYECTO (#idProy, Nombre, Presupuesto, idDepto)
- EMP_DEPENDIENTE (#NumEmp, Nombre_Dependiente, FecNacimiento, Relación)

Consulta 1: Dar número y la dirección de los empleados que se encuentran en el proyecto 1.

(<t. NumEmp, t. Dirección >/EMPLEADO (e) ∧ ∃v(PROYECTO(p) ∧ p.idDepto = e.idDepto ∧ p. idProy = 1)

Consulta 2: Buscar el nombre, la relación de todos los dependientes de empleados que trabaja para el Departamento de Recursos Humanos

(t . NumEmp, t .Nombre_Dependiente, t .Relación | t ∈ EMP_DEPENDIENTE ∧ ∃ e ∈ EMPLEADO (e.NumEmp = t.NumEmp ∧ ∃ d ∈ DEPARTAMENTO (d.idDepto = e.idDepto ∧ d . Nom_Depto ='Recursos Humanos')))

Consulta 3: Busca los nombres de los empleados que no tienen dependientes

{ t . Nombre | t ∈ EMPLEADO ∧ ¬ (∃ d ∈ EMP_DEPENDIENTE (d. NumEmp = t .NumEmp)) }

Cálculo de Dominios

El cálculo de dominios difiere del cálculo de tuplas en que sus variables de alcance abarcan dominios en lugar de relaciones. Desde un punto de vista práctico, la diferencia de sintaxis que resulta más obvia inmediatamente es que el cálculo de dominios maneja una forma adicional de <expresión lógica> a la cual nos referiremos como condición de pertenencia. (Date, 2001)

Una condición de pertenencia toma la forma R (par, par,...) donde R es un nombre de relación o tabla y cada par es de la forma A: v, donde A es un atributo de R y v puede ser el nombre de una variable de alcance del cálculo de dominios, o bien una invocación a selector (por lo regular una literal). La condición da como resultado "*verdadero*" si, y sólo:

Existe una tupla en cualquier relación que sea el valor actual de R y que tenga los valores especificados para los atributos señalados. Por ejemplo, la expresión VP (V#: V# ('V1'), P#: P# ('P1')) es una condición de pertenencia que da como resultado "*verdadero*" si, y sólo si, existe actualmente una tupla de envío con

el valor V1 en V# y el valor P1 en P#. En forma similar, la condición de pertenencia VP (V#:VX, P#:PX) da como resultado "*verdadero*" si, y sólo si, existe actualmente una tupla de envío con un valor de V# igual al valor actual de la variable de alcance VX (cualquiera que éste pueda ser) y un valor de P# igual al valor actual de la variable de alcance PX (cualquiera que éste pueda ser). (Date, 2001)

Para el resto de esta sección supondremos la existencia de variables de alcance para cálculo de dominios, como sigue:

Dominio:

v# p# NOMBRE COLOR PESO CANT CHAR INTEGER

Variables de alcance:

vx,VY, ...

PX, PY, ... NOMX,NOMY, ... COLORX,COLOP.Y, . . . PESOX,PESOY, ... CANTX,CANTY, ... CIUDADX,CIUDADY, .. STATUSX, STATUSY,..

Entonces, aquí tenemos algunos ejemplos de expresiones del cálculo de dominios:

- Vx VX WHERE V (V#:VX)
- VX WHERE V (V#:VX, CIUDAD:'Londres')
- (VX, CIUDADX) WHERE V (V#:VX, CIUDAD:CIUDADX)
  AND VP (V#:VX, P#:P#('P2'))
- (VX, PX) WHERE V (V#:VX, CIUDAD:CIUDADX)
  AND P (P#:PX, CIUDAD:CIUDADY)
  AND CIUDADX <> CIUDADY

Lenguaje SQL comercial

Puede considerarse que el lenguaje *SQL* es una de las razones más importantes del éxito de las *Bases de Datos Relacionales* en el mundo comercial. Como se convirtió en el lenguaje estándar para estas *Bases de Datos* permitió a los usuarios no tener que preocuparse por migrar sus aplicaciones de bases de datos desde otros tipos de sistemas -como los sistemas en red o los sistemas jerárquicos- a sistemas relacionales. Al seguir los mismos estándares de lenguaje, los *SGDB* relacionales permiten la conversión de un determinado producto SGBD relacional a otro de manera rápida y poco costosa. En la práctica, por supuesto, hay muchas diferencias entre los distintos paquetes comerciales de *SGBD* relacionales pero, si el usuario es diligente, la conversión resulta simple. Otra ventaja de tener este estándar es que, en el caso de que dos o más *SGBD* relacionales soporten el estándar *SQL*, los usuarios pueden escribir sentencias en un programa de aplicación de bases de datos que puede acceder a información almacenada en dichos SGBD, sin tener que cambiar el sub lenguaje (SQL) de la base de datos. (Elmasri & Navathe, 2002)

Las operaciones del algebra relacional se consideran demasiado técnicas para la mayoría de los usuarios de se SGBD relacionales. Una razón es que una consulta en algebra relacional se escribe en forma de secuencias de operaciones que, al ejecutarse, producen el resultado deseado. Por lo tanto, el usuario debe especificar cómo (es decir, en qué orden) se deben de ejecutar las operaciones de consulta. Por otro lado, el lenguaje SQL proporciona una interfaz de leguaje declarativo de alto nivel, de modo que el usuario solo tiene que especificar cuál es el resultado, esperado dejando que el SGBD se encargue de la optimización actual y de las decisiones sobre cómo se ejecutará la consulta. SQL incluye algunas características de álgebra relacional, pero está basado en gran parte en el cálculo relacional orientado a tuplas.

La sintaxis de SQL es más amigable que cualquiera de los dos lenguajes formales. (Elmasri & Navathe, 2002)

El nombre SQL se deriva de *Structured Query Language. Originalmente*, SQL se llamaba SEQUEL, (de *Structured English QUEry Language*) y fue diseñado e implementado por *IBM Research* como interfaz para un sistema experimental de bases de datos relacional llamado SYSTEM R. Ahora SQL es el lenguaje estándar de los SGBD relacionales. Un esfuerzo conjunto de ANSI (*American National Standads Institute*) e ISO (*International Standards Organization*) ha dado lugar a una versión estándar de SQL (ANSI 1986), llamada SQL-86 o SQL1. Posteriormente, se ha desarrollado un estándar revisado y más expandido llamado SQL2 (también llamado SQL-92). Ya existen planes para SQL3, que extenderá SQL con conceptos de orientación a objetos y otros conceptos recientes de bases de datos. (Elmasri & Navathe, 2002)
SQL es un lenguaje de base de datos global; cuenta con enunciados de definición, consulta y actualización de datos. Así pues, es tanto un DDL[25] como un DML[26]. Además, cuenta con mecanismos para definir vistas de la base de datos, para especificar seguridad y autorización por lo tanto es un lenguaje DCL[27], finalmente SQL es capaz de definir restricciones de integridad, y para especificar controles de transacciones por lo tanto es también un lenguaje TCL[28]. Tiene reglas para insertar sentencias de SQL en leguajes de programación de propósito general como *C* o *PASCAL*. (Elmasri & Navathe, 2002)

Las instrucciones SQL más importantes ordenadas por tipo de lenguaje son:

- DDL: CREATE, ALTER, DROP, TRUNCATE
- DML: SELECT, UPDATE, INSERT, DELETE
- DCL: GRANT, REVOKE
- TCL: BEGIN, COMMIT, ROLLBACK

[25] Lenguaje de definición de datos, en inglés *Data Definition Language*.
[26] Lenguaje de manipulación de datos, en inglés *Data Manipulation Language*.
[27] Lenguaje de Control de Datos, en ingles *Data Control Language*. Se utiliza para crear roles, permisos, y la integridad referencial, así que se utiliza para controlar el acceso a la base de datos, asegurando la misma.
[28] Lenguaje de control de transacciones en ingles *Transaction Control Languaje*. Sus declaraciones se utilizan para gestionar los cambios introducidos por las instrucciones DML.

Definición de datos (DDL)

Antes de entrar de manera específica en la definición de los datos, hay un par de puntos que señalar sobre el tema de las tablas de SQL en general: (Date, 2001)

- Primero, se permite que las tablas de SQL incluyan filas duplicadas. Por lo tanto, no necesariamente tienen una clave primaria (o, de manera más fundamental, claves candidatas).

- Segundo, se considera que las tablas de SQL tienen un ordenamiento de izquierda a derecha. Por ejemplo, en la tabla CLIENTES, la columna NUMCLI podría ser la primera columna, la columna NOMCLI podría ser la segunda, etcétera.

CREATE TABLE

Crea una nueva tabla dentro de la base de datos. (Observe que aquí la palabra reservada TABLE se refiere específicamente al nombre una tabla de la base; esto mismo es cierto para ALTER TABLE y DROP TABLE). (Date, 2001)

La sintaxis es la siguiente:[29]

```
CREATE TABLE <nombre de tabla base>
      (
            <Lista de elementos de la tabla base separados con comas >
      )
GO
```

Cada *<elemento de la tabla base>* es:

- una *<definición de columna>*,
- bien una *<restricción>.*

Las *<restricciones>* especifican ciertas restricciones de integridad que se aplican a las tablas base en cuestión.

Las *<definiciones de columna>* (debe existir por lo menos una) toman la siguiente forma general:

<nombre de columna> <nombre de tipo o de dominio> [<especificación predeterminada>]

Aquí, el *<nombre de tipo o de dominio>* es un nombre de tipo integrado o bien un nombre de dominio, y la *<especificación predeterminada>* opcional especifica un valor predeterminado para la columna, el cual sustituye a cualquier valor predeterminado especificado en el nivel del dominio (en caso de que sea aplicable).

Ejemplo:

```
CREATE TABLE [dbo].[CEDIS]
 (
       [NUMCLI]  [char](6)              NOT NULL,
       [IDCEDIS] [smallint]             NOT NULL,
       [DET]     [varchar](8)           NOT NULL,
       [NOMBRE]  [varchar](30)          NOT NULL,
       [TEL]     [varchar](20)          NULL,
       [ABREV]   [varchar](8)           NULL
 )
```

[29] Sintaxis completa CREATE TABLE en Anexo: I.VI Sintaxis completa instrucciones Microsoft SQL Server

ALTER TABLE
Altera cualquier tabla de la base existente en cualquier momento.

Las siguientes "alteraciones" son soportadas: (Date, 2001)

- Es posible agregar una nueva columna.
- Es posible definir un valor predeterminado para una columna existente (remplazando, si lo hay, al anterior).
- Es posible eliminar el valor predeterminado de una columna existente.
- Es posible eliminar una columna existente.
- Es posible especificar una nueva restricción de integridad.
- Es posible eliminar una restricción de integridad existente.

Sólo ofrecemos un ejemplo del primer caso:[30]

```
 ALTER TABLE [CEDIS] ADD RESPONSABLE VARCHAR (20) NULL
GO
```

Esta instrucción agrega la columna `RESPONSABLE` (del tipo VARCHAR (20)) a la tabla base de CEDIS. Todas las filas de esa tabla se amplían de seis a siete columnas; en todos los casos, el valor inicial de la quinta columna nueva es NULL.

DROP TABLE
Quita una definición de tabla todos los datos, índices, desencadenadores, restricciones y especificaciones de permisos de la tabla. Las vistas o procedimientos almacenados que hagan referencia a la tabla eliminada se deben quitar explícitamente con la instrucción DROP VIEW o DROP PROCEDURE.

Sintaxis
DROP TABLE nombre_tabla

Argumentos
nombre_tabla: Es el nombre de la tabla que se va a eliminar

Observaciones
No se puede utilizar DROP TABLE para quitar una tabla a la que se haga referencia con una restricción FOREIGN KEY. Primero se debe quitar la restricción FOREIGN KEY o la tabla de referencia.

El propietario de una tabla puede quitar la tabla de cualquier base de datos. Cuando se quita la tabla, las reglas o valores predeterminados de la misma pierden sus enlaces y se quitan automáticamente las restricciones o desencadenadores asociados con ella. Si vuelve a crear una tabla, debe volver a enlazar las reglas y valores predeterminados apropiados, volver a crear los desencadenadores y agregar todas las restricciones necesarias.

No puede utilizar la instrucción DROP TABLE sobre las tablas del sistema.

Si elimina todas las filas de una tabla (DELETE *tablename*) o utiliza la instrucción TRUNCATE TABLE, la tabla existe hasta que se quite. (Microsoft, 2012)

[30] Sintaxis completa ALTER TABLE en Anexo: VI Sintaxis completa instrucciones Microsoft SQL Server

Manipulación de datos (DML)

Una vez definida la base de datos, podemos ahora comenzar a operar en ella por medio de las operaciones SQL de manipulación SELECT, INSERT, UPDATE y DELETE.

SELECT

Es la sentencia que tiene SQL para recuperar información de una base de datos. La forma básica de la sentencia SELECT[31], en ocasiones denominada correspondencia o bloque SELECT-FROM-WHERE, consta de las tres clausulas SELECT, FROM y WHERE y tiene la siguiente forma: (Elmasri & Navathe, 2002)

SELECT <lista de atributos>
FROM <lista de tablas>
WHERE <condición>

Donde:

- <lista de atributos> es una lista de nombres de atributos cuyos valores van a ser recuperados por la consulta.
- <lista de tablas> en una lista de nombres de las relaciones necesarias para procesar la consulta.
- <condición> es una expresión condicional (booleana) que identifica las tuplas que van a ser recuperadas por la consulta.

Ilustraremos la sentencia SELECT básica con algunos ejemplos de consultas. Utilizando la tabla *EmpAlm* que se encuentra en la página 28.

Por ejemplo obtener los Números de empleado (*NumEmp*) , Nombres del empleado (*Nombre*), Apellidos Paternos (*ApPat*), Apellidos Maternos (*ApMat*) de los empleados cuyo Apellido Paterno sea '*Juárez*', su identificador de Puesto (*IdPuesto*) sea '*10'* y su estatus (*Status*) sea *'1'*

Consulta SELECT

```
SELECT NumEmp , Nombre, ApPat, ApMat
FROM   EmpAlm
WHERE  ApPat    = 'Juárez'
AND    IdPuesto = '10'
AND    Status   = '1'
```

En esta consulta sólo interviene la relación o tabla *EMPALM* en la clausula FROM. La consulta selecciona las tuplas de *EMPLEADOS* que satisfacen la condición de la cláusula WHERE, luego proyecta el resultado *NUMEMP, NOMBRE, APPAT y APMAT*. La clausula SELECT es similar a la siguiente expresión del algebra relacional, excepto que los duplicados, si los hay, no se eliminarían:

$$\sigma_{ApPat=JUAREZ,\,IdPuesto=10,\,Status=1}(\mathrm{EmpAlm})$$

INSERT

En su forma más simple, INSERT sirve para añadir una sola tupla a una relación. Debemos especificar el nombre de la relación y una lista de valores para la tupla. Los valores deberán enumerarse en el mismo orden en que se especificaron los atributos correspondientes en la instrucción *CREATE TABLE*. (Elmasri & Navathe, 2002)

[31] Hay muchas opciones y matices para la sentencia SELECT de SQL, para más información ver Anexo: VI Sintaxis completa instrucciones Microsoft SQL Server

Por ejemplo, si queremos añadir una nueva tupla a la relación *EMPALM* que se muestra en la página 28. Podemos usar:

```
INSERT INTO EmpAlm
VALUES ('000001','RENE','DOMINGUEZ','BORJAS', '1', 1, '00001')
```

Una segunda forma de sentencia INSERT permite al usuario especificar nombres de atributos explícitos que correspondan a los valores de la instrucción INSERT. Esto es útil si una relación tiene muchos atributos pero solo se van a asignar valores a una nueva tupla. Estos últimos atributos deben incluirse con la especificación de NOT NULL y sin valores por defecto; los atributos con valores NULL o DEFAULT se pueden omitir. Por ejemplo si queremos introducir una tupla para un nuevo EMPLEADO del cual solo conocemos los atributos Número de empleado (*NumEmp*), Nombre, Apellido Paterno (*ApPat*), Identificado de puesto (*IdPuesto*) y *Status* podemos usar:

```
INSERT INTO EmpAlm(NumEmp, Nombre, ApPat, IdPuesto, Status)
VALUES ('000002','RICARDO','MARTINEZ', '1', 1)
```

Los atributos no especificados reciben por omisión DEFAULT o NULL y los valores se enumeran en el mismo orden en que aparecen los atributos en la instrucción INSERT.[32]

UPDATE

La instrucción UPDATE sirve para modificar los valores de los atributos en una o más tuplas seleccionadas. Al igual que en la instrucción DELETE, una cláusula WHERE en la instrucción UPDATE selecciona de una sola relación las tuplas que se van a modificar. Sin embargo, la actualización de un valor de *clave primaria* puede propagarse a los valores de *clave externa* de tuplas de otras relaciones si se especifica acción de disparo referencial de las restricciones de integridad referencial del DDL. Una Clausula SET adicional especifica los atributos que se modificaran y sus nuevos valores.

Por ejemplo, para la relación *EMPALM* que se muestra en la página 28. Si queremos agregarle al empleado "*000002*" su apellido materno y su clave personal, usaremos

```
UPDATE  EmpAlm
SET     ApMat  = 'Larrauri',  Pass = 'Padi'
WHERE   NumEmp = '000002'
```

DELETE

La instrucción DELETE elimina tuplas de una relación. Cuenta con una cláusula WHERE, similar a la de las consultas SQL, para seleccionar las tuplas que van a ser eliminadas. Las tuplas se eliminan explícitamente de una sola tabla cada vez. Sin embargo, la eliminación puede propagarse a tuplas de otras relaciones si las acciones de disparo referencial se especifican en las restricciones de integridad referencial del *DDL*.

Dependiendo del número de tuplas seleccionadas por la condición de la cláusula WHERE, una sola instrucción DELETE puede eliminar cero, una o varias tuplas. La omisión de la cláusula WHERE, indica que se deben eliminar todas las tuplas de la relación; sin embargo la tabla permanecerá en la base de datos como una tabla vacía. (Elmasri & Navathe, 2002)

[32] Para ver la sintaxis complete así como la explicación de todos los atributos ver: http://technet.microsoft.com/es-es/library/ms174335.aspx

Por ejemplo si queremos eliminar tuplas de la relación *EMPALM* que se muestra en la página 28:

- `DELETE EmpAlm WHERE NumEmp = '000002'`
- `DELETE EmpAlm WHERE IdPuesto in (SELECT IdPuesto WHERE DescPuesto = 'JEFE DE ALMACEN DE PROD. TERM.')`
- `DELETE EmpAlm`

Lenguaje de Control de Datos (DCL)

Como mencionamos anteriormente el lenguaje de control de datos es la parte de *SQL* que se encarga de controlar la seguridad y los permisos de la base de datos.

Las directivas de DCL de *SQL* no forma parte de los lenguajes relacionales como tal y tienen una gran cantidad de variaciones dependiendo del SGBD. Sus principales características para la protección de los datos son:

- Manipulación de los mismos sin autorización
- Prevención de errores
- Prevención de daños

GRANT

Concede permisos sobre un elemento protegible a una entidad de seguridad. El concepto general es:

GRANT <algún permiso>
ON <algún objeto>
TO <algún usuario, inicio de sesión o grupo>. [33]

Donde:

- <algún permiso>: Todos los elementos protegibles de *SQL Server* tienen permisos asociados que se pueden conceder a una entidad de seguridad.[34]
- <algún objeto>: elementos protegibles por SQL. [35]
- <algún usuario, inicio de sesión o grupo>: Es el nombre de una entidad de seguridad. Las entidades de seguridad a las que se pueden conceder permisos para un elemento protegible varían según el elemento protegible.

REVOKE

Quita un permiso concedido o denegado previamente. El concepto general es:

REVOKE <algún permiso>
ON <algún objeto>
TO <algún usuario, inicio de sesión o grupo>. [36]

Donde:

- <algún permiso>: Todos los elementos protegibles de SQL Server tienen permisos asociados que se pueden conceder a una entidad de seguridad.
- <algún objeto>: elementos protegibles por SQL.

[33] Para ver la sintaxis complete así como todos los atributos ver: http://msdn.microsoft.com/es-mx/library/ms187965(v=sql.110).aspx
[34] Más información sobre permisos ver: http://msdn.microsoft.com/es-mx/library/ms191291.aspx
[35] Más información sobre elementos protegibles ver: http://msdn.microsoft.com/es-mx/library/ms187965(v=sql.110).aspx
[36] Para ver la sintaxis complete así como la explicación de todos los atributos ver: http://msdn.microsoft.com/es-es/library/ms187728.aspx

- <algún usuario, inicio de sesión o grupo>: Es el nombre de una entidad de seguridad. Las entidades de seguridad a las que se pueden conceder permisos para un elemento protegible varían según el elemento protegible.

Leguaje de Control de Transacciones (TCL)

El *Leguaje de Control de transacciones* permite confirmar y deshacer cambios efectuados por las sentencias *DML*.

Una transacción es una unidad única de trabajo. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos. Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos.

BEGIN TRANSACTION , COMMIT TRANSACTION y ROLLBACK TRANSACTION

- BEGIN TRANSACTION: Marca el punto de inicio de una transacción local explícita. Incrementa @@TRANCOUNT en 1.

- COMMIT TRANSACTION: Marca el final de una transacción correcta, implícita o explícita. Si @@TRANCOUNT es 1, COMMIT TRANSACTION hace que todas las modificaciones efectuadas sobre los datos desde el inicio de la transacción sean parte permanente de la base de datos, libera los recursos mantenidos por la transacción y reduce @@TRANCOUNT a 0. Si @@TRANCOUNT es mayor que 1, COMMIT TRANSACTION solo reduce @@TRANCOUNT en 1 y la transacción sigue activa.

- ROLLBACK TRANSACTION: Revierte una transacción explícita o implícita hasta el inicio de la transacción o hasta un punto de retorno dentro de la transacción. Puede usar ROLLBACK TRANSACTION para borrar todas las modificaciones de datos realizadas desde el inicio de la transacción o hasta un punto de retorno. También libera los recursos que mantiene la transacción.

Sintaxis:

BEGIN { TRAN | TRANSACTION } [{ nombre_tran | @variable_nombre_tran } [WITH MARK ['description']]

COMMIT { TRAN | TRANSACTION } [nombre_tran | @variable_nombre_tran]]

ROLLBACK { TRAN | TRANSACTION } [nombre_tran | @variable_nombre_tran | nombre_Punto_Ini | @nombre_variable_Punto_Ini]

Argumentos:

nombre_tran

Es el nombre asignado a la transacción. *nombre_tran* debe cumplir las reglas de los identificadores, pero no se admiten identificadores de más de 32 caracteres. Utilice nombres de transacciones solamente en la pareja más externa de instrucciones BEGIN…COMMIT o BEGIN…ROLLBACK anidadas.

@variable_nombre_tran

Se trata del nombre de una variable definida por el usuario que contiene un nombre de transacción válido. La variable debe declararse con un tipo de datos *char*, *varchar*, *nchar* o *nvarchar*. Si se pasan más de 32 caracteres a la variable, solo se utilizarán los primeros 32; el resto de caracteres se truncará.

WITH MARK ['description']

Especifica que la transacción está marcada en el registro. Description es un valor de tipo *string* que describe la marca. Un valor de description superior a 128 caracteres se trunca a 128 caracteres antes de almacenarse en la tabla *msdb.dbo.logmarkhistory*.

nombre_Punto_Ini

Es nombre del punto de inicio de una instrucción SAVE TRANSACTION. *nombre_Punto_Ini* debe ajustarse a las reglas de los identificadores. Utilice *nombre_Punto_Ini* cuando una operación de reversión condicional solo deba afectar a parte de la transacción.

@nombre_variable_Punto_Ini

Es el nombre de una variable definida por el usuario que contiene un nombre de punto de retorno válido. La variable debe declararse con un tipo de datos char, varchar, nchar o nvarchar.

Ejemplo:

```
BEGIN TRAN T1

      INSERT INTO EmpAlm (NumEmp, Nombre, ApPat, IdPuesto, Status)
      VALUES ('000002','RICARDO','MARTINEZ', '1', 1)

      IF @@ERROR <> 0
         BEGIN
              ROLLBACK TRAN T1
         END
COMMIT TRAN T1
```

2.1.5. *DEPENDENCIAS FUNCIONALES Y LA TEORÍA DE LA NORMALIZACIÓN*

El propósito de las dependencias funcionales y la teoría de la normalización es garantizar que el esquema relacional definido para una base de datos está bien construido. Un esquema relacional mal hecho puede causar anormalidades en el manejo de una base de datos.

Por ejemplo:

EmpAlmNvo					
NumEmp	Nombre	ApPat	ApMat	Puesto	Departamento
001693	EDUARDO	SANCHEZ	IGLESIAS	AUXILIAR VENTAS	VENTAS
001850	JORGE	JUAREZ	VALVERDE	JEFE APT	APT
002694	VICTOR RAUL	JUAREZ	VILLALBA	PREPARADOR APT	APT
003003	SONIA	TERRON	PAZ	PREPARADOR APT	APT
003725	JHONK KAREN	DELGADO	MUÑOZ	PREPARADOR APT	APT

Esta relación está mal construida: la información sobre *Puesto* y *Departamento* es redundante. Para actualizar el puesto "*PREPARADOR APT*", es esencial verificar que todas las direcciones "*PREPARADOR APT"* estén actualizadas. Esta solución no es razonable. La base de datos se convertirá rápidamente en inconsistente.

La solución es evitar la redundancia en la base de datos. Esto requiere que el esquema de la base de datos está bien construido. Existen dos métodos que ayudan a reducir la redundancia en una base de datos relacional:

- Estudio de las dependencias funcionales.
- La descomposición y la normalización de las relaciones.

Dependencias funcionales (DF)

Las DF proporcionan una base para enfrentar, de manera científica, diversos problemas prácticos ya que, por el rico conjunto de propiedades formales que poseen, son útiles para estructurar la información en la base de datos. Estas propiedades especiales hacen posible tratar los problemas en cuestión de una manera formal y rigurosa.

En esencia, una DF es un vínculo muchos a uno, que va de un conjunto de atributos a otro, dentro de una determinada relación o tabla. (Date, 2001)

Por ejemplo, en el caso de la relación anterior *EmpAlmNvo*, existe una dependencia funcional del conjunto de atributos *NumEmp* al conjunto de atributos *Puesto*.

Lo que esto significa es que dentro de cualquier relación que resulte ser un valor válido de la relación *EmpAlmNvo*:

- Para cualquier valor dado del atributo *NumEmp* sólo existe un valor correspondiente del atributo *Puesto*, pero
- Muchos valores distintos atributo *NumEmp* pueden tener (en general) el mismo valor correspondiente del atributo *Puesto*.

Formalmente:

Sea el esquema de relación *R(A, DF)* y sean *X* e *Y* dos subconjuntos de atributos de *A*. Se dice que existe una *DF* entre *X* e *Y*, de forma que *X* determina a *Y* ($\boldsymbol{X \rightarrow Y}$) si y sólo si se cumple que para cualesquiera dos tuplas de *R*, u y v tales que *u[X] = v[X],* entonces necesariamente *u[Y] = v[Y].* (Date, 2001)

Esto significa que a cada valor *x* del atributo *X*, le corresponde un único valor *y* del atributo *Y*.

Lo contrario, decir *X* no determina funcionalmente a *Y*:

$X \not\rightarrow Y$ o $X \rightarrow Y$

Un *determinante* o *implicante* es un conjunto de atributos del que depende funcionalmente otro conjunto de atributos al que llamamos *determinado* o *implicado*.

Ejemplo:

El número de empleado determina el nombre del mismo:

NumEmp → Nombre *(Nombre, Apellido Paterno, Apellido Materno)*

Esta propiedad se establece a partir de la intención de la relación y no del contenido (significa que es invariante en el tiempo y no se puede extraer a partir de ejemplos). Se trata de una propiedad que debe ser extraída a partir del conocimiento que se tiene del modelo.

Dependencia funcional plena o completa

En una dependencia funcional *X -> Y*, cuando *X* es un conjunto de atributos, decimos que la dependencia funcional es completa, si *Y* sólo depende de *X*, y no de ningún subconjunto de *X*.

La dependencia funcional se representa como ***X => Y***.

Por tanto, $X \Rightarrow Y$ sii $\neg \exists X' \subset X \mid X' \rightarrow Y$

• Ejemplo: en la relación

PAGO_NOMINA (Periodo, NumEmp, Sueldo)

La DF plena ***Periodo, NumEmp ⇒ Sueldo*** refleja el sueldo que obtiene un empleado en un periodo determinado.

Se denomina ***atributo extraño*** a aquellos atributos del determinante de una DF que hacen que ésta no sea plena. También se llaman ajenos. Por ejemplo:

La DF ***Periodo, NumEmp, Cod_Banco ⇒ Sueldo***

Es no plena y *Cód_Banco* es un atributo extraño, ya que el banco no determina el sueldo de un empleado.

Dependencias funcionales triviales y no triviales

Una DF $X \rightarrow Y$ es trivial si y sólo si Y (el dependiente) es un subconjunto de *X* (el determinante). (Date, 2001)

$Y \subseteq X$.

Ejemplo: las siguientes DF son triviales:

NumEmp, CURP → CURP

Una DF no trivial es aquella donde el dependiente no es un subconjunto del determinante.

En bases de datos estamos más interesados en las dependencias no triviales, ya que son las únicas que corresponden a restricciones de integridad ***"genuinas"***. (Date, 2001)

Dependencia funcional elemental

Si tenemos una dependencia completa *X => Y*, diremos que es una dependencia funcional elemental si Y es un atributo, y no un conjunto de ellos.

Es decir, una *DF* elemental es una *DF plena*, *no trivial* y en la que el implicado es un *atributo único*:

$$X \rightarrow Y \text{ es elemental } sii \ (\neg \exists Y' \subset Y) \wedge (Y \subseteq X) \wedge (\neg \exists X' \subset X | X' \rightarrow Y)$$

Únicamente las *DF* elementales son útiles para la normalización.

El resto de *DF* no interesa y no se tienen en cuenta. Estas son las dependencias que buscaremos en nuestras relaciones. Las dependencias funcionales elementales son un caso particular de las dependencias completas. (Date, 2001)

Consecuencia Lógica

El conocimiento de ciertas *DF* puede llevar a inferir la existencia de otras que no se encontraban en el conjunto inicial:

Dado un esquema de relación: *R (A, DF)* es posible deducir de *DF* nuevas dependencias funcionales que sean una consecuencia lógica del conjunto de partida.

Las nuevas dependencias *f* que se cumplen para cualquier extensión de *r* de *R* son consecuencia lógica de *DF* (vienen implicadas por *DF*). Se representan como:

DF |= f

Ejemplo:

Dado el esquema de relación PAGO_EMPLEADO ({NumEmp, CURP, Tipo_Contrato, Forma_Pago}, DF)

donde:

$$DF = \{ NumEmp \rightarrow CURP;\ CURP \rightarrow NumEmp;\ NumEmp, Tipo\_Contrato \rightarrow Forma\_Pago \}$$

se cumple que

$$DF \models CURP, Tipo\_Contrato \rightarrow Forma\_Pago$$

Transitiva y Directamente dependiente

Dado el esquema de relación *R (X, Y, Z)* una DF X – → Z (representada por una flecha discontinua) es transitivamente dependiente cuando *Z* no es inmediatamente dependiente de *X*, pero sí de un tercer conjunto de atributos *Y,* que a su vez depende de *X*.

Es decir, $X \rightarrow Y$, $Y \rightarrow Z$, $Y \not\rightarrow X$ por virtud de $X \rightarrow Y$ **e** $Y \rightarrow Z$.[37]

Una DF es transitiva estricta cuando además de las condiciones anteriores, también se cumple Que $Z \not\rightarrow Y$

Una DF $X \rightarrow Z$, es directamente dependiente cuando *Z* no es transitivamente dependiente de X.

Ejemplo:

[37] Notar que *X* e *Y* no tienen que ser equivalentes

Dada la relación EMPLEADO_DEPTO (NumEmp, Puesto, Cód_Departamento) en donde se tiene para cada empleado un número único (*NumEmp*), el puesto que ocupa en la Empresa y el departamento donde trabaja (suponemos que un empleado solamente tiene un puesto y que cada puesto depende de un único departamento) se tendrán las siguientes DF:

NumEmp → Puesto

Puesto → Cód_Departamento

Además, con ese puesto puede haber uno o más empleados:

Puesto/→ NumEmp

y por tanto, se cumple la DF transitiva

NumEmp –→ Cód_Departamento que también es estricta puesto que

Cód_Departamento /→ Puesto

Cierre de un Conjunto de Dependencias

El cierre de un conjunto de dependencias funcionales DF (que se denota DF+) es el conjunto de todas las dependencias que son consecuencia lógica de DF:

DF+ = { X → Y | DF |= X → Y }

DF será siempre un subconjunto del cierre *(DF ⊆ DF+).* Por lo tanto, las notaciones *R(A, DF) y R(A, DF+)* definen el mismo esquema de relación.

Estas definiciones no permiten el cálculo del cierre, siendo necesarias unas reglas de derivación que faciliten la implicación lógica de dependencias. Estas reglas de derivación, se conocen como *Axiomas de Armstrong*, y forman un conjunto completo y correcto de axiomas. (Date, 2001)

Reglas de Derivación ó axiomas de Armstrong

Dado un conjunto DF de dependencias funcionales, se dice que *f* se deriva de *DF*, lo que se representa por ***DF |— f***, si *f* se puede obtener por aplicación sucesiva de los *axiomas de Armstrong* a DF (o a un subconjunto de DF), es decir, si existe una secuencia de dependencias *f1, f2, … fn* tal que *fn = f,* donde cada *fi* es bien un elemento de DF o ha sido derivada a partir de las dependencias precedentes aplicando las reglas de derivación.

Aunque son conceptos distintos, se cumple siempre que si una dependencia *f* es una consecuencia lógica de un conjunto de dependencias, también será posible derivarla de dicho conjunto aplicando los *axiomas de Armstrong*, y viceversa; es decir (Elmasri & Navathe, 2002):

∀f | DF |–f se_implica_que ***DF |= f*** (propiedad de corrección) y ***∀f | DF |= f*** se_implica_que ***DF |–f*** (propiedad de plenitud)

Axiomas de Armstrong

Básicos:

- *A1: Reflexividad*
  Si Y ⊆ X, X → Y (X → Y es una DF trivial)

- *A2: Aumentatividad*
  Si X → Y y Z ⊆ W, entonces XW → YZ

- *A3: Transitividad*
  Si X → Y e Y → Z, entonces X → Z

Derivados:

- *D1: Proyectividad*
  Si X → Y, entonces X → Y' si Y' ⊂ Y

- *D2: Unión o aditividad*
  Si X → Y y X → Z, entonces X → YZ

- *D3: Pseudotransitividad*
  Si X → Y e YW → Z, entonces XW → Z

Ejemplo:

Dado el esquema de relación:

R(NumEmp, Puesto, Banco, Banco_Sucursal, Tipo_cuenta;

{NumEmp→Puesto, Banco→Banco_Sucursal, Banco_Sucursal→Tipo_cuenta})

Demostrar que:

NumEmp , Banco → NumEmp, Puesto, Banco, Banco_Sucursal, Banco_Sucursal, Tipo_cuenta

Se demuestra aplicando los axiomas de Armstrong de la siguiente forma:

1. ***NumEmp→Puesto*** (dada)
2. ***NumEmp, Banco → NumEmp, Puesto, Banco*** (aumentatividad de la anterior por *NumEmp, Banco*)
3. ***Banco→Banco_Sucursal*** (dada)
4. ***Banco_Sucursal→Tipo_cuenta*** (dada)
5. ***Banco →Tipo_cuenta*** (transitividad de 3 y 4)
6. ***Banco → Tipo_cuenta, Banco_Sucursal*** (unión de 3 y 5)
7. ***NumEmp, Puesto, Banco → NumEmp, Puesto, Banco, Banco_Sucursal, Tipo_cuenta*** (aumentatividad de 6 por *NumEmp, Puesto, Banco*)
8. ***NumEmp, Banco → NumEmp, Puesto, Banco, Banco_Sucursal, Banco_Sucursal, Tipo_cuenta*** (transitividad de 2 y 7)

Aunque los *axiomas de Armstrong* facilitan un procedimiento algorítmico para calcular el cierre DF+ de un conjunto de dependencias, su cálculo consume mucho tiempo, ya que, aunque el número inicial de dependencias sea pequeño, el número total de dependencias en el cierre es muy elevado. Sin embargo el *algoritmo de Armstrong*, así como los conceptos de DF, nos pueden ayudar para:

1. Determinar si una dependencia $X \rightarrow Y$ pertenece al cierre DF+
2. Encontrar un procedimiento eficiente no basado en el cierre de un conjunto de dependencias, para determinar la equivalencia entre dos conjuntos de dependencias.
3. Hallar un recubrimiento irredundante, necesario para abordar el tema de la normalización, tanto en los algoritmos de síntesis como de análisis.
4. Verificar si un determinante es clave de un esquema de relación.
5. Obtener todas las claves de un esquema relación.

Normalización

Además de las restricciones impuestas por las reglas generales del *Modelo Relacional*, y de las reglas específicas impuestas por el administrador de bases de datos, existe una serie de reglas que permite reforzar el *Modelo Relacional,* ayudando a mantener la integridad de los datos y evitando la redundancia. Esto es lo que se conoce como normalización.

La teoría de *Normalización* nos permite un diseño de base de datos riguroso y demuestra que existen métodos automáticos de diseño que permiten llegar a ciertas formas normales que garantizan un diseño de calidad. Es decir, siempre se puede asegurar el que una base de datos relacional esté al menos en forma normal de *Boyce Codd* y con esto habremos eliminado gran parte de los problemas de redundancias y de ambigüedad que pueden surgir durante el diseño.

La *Normalización* puede considerarse como un proceso durante el cual los esquemas de relación insatisfactorios se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables. Un objetivo del proceso de normalización original es garantizar que no ocurran las anomalías de actualización, así como la eliminación de información redundante en las tuplas.

Existen tres formas normales básicas, expuestas por Codd en la primera versión del modelo (Codd, 1972), conocidas como *1NF, 2NF y 3NF*, respectivamente, más otras tres que fueron añadidas con posterioridad (*BCNF, 4NF y PJ/NF ó 5NF*). En realidad, *BCNF* (la forma normal de *Boyce/Codd*) (Codd, 1974) no es más que un intento de tapar los huecos de *3NF*, y durante un tiempo fue llamada simplemente *3NF*. Las dos restantes, *4NF* y *PJ/NF*, no fueron definidas por Codd, sino por R. Fagin (Fagin, 1977 y 1979, respectivamente).

Primera Forma Normal (1FN)

Una relación R está en primera forma normal, si y sólo si todos los atributos contienen únicamente valores atómicos. (Date, 2001)

Esta condición es una restricción inherente al modelo relacional, ya que como hemos dicho anteriormente, en el cruce de una fila y una columna debe haber un único valor, no pueden existir atributos multivaluados. Como es una restricción inherente al modelo, está al menos en primera forma normal.

Atómica significa "i*ndivisible*", es decir, cada atributo debe contener un único valor del dominio. Los atributos, en cada tabla de una base de datos *1FN*, no pueden tener listas o arreglos de valores, ya sean del mismo dominio o de dominios diferentes.

Ejemplo:

Cada departamento de una empresa tiene un identificador único por departamento (*IdDepto*), tiene un *Nombre*, un solo jefe de departamento y varios empleados.

Tanto el jefe de departamento como los empleados sé que se pueden identificar por su Número de Empleado (*IdJefeDepto* y *NumEmp*).

Departamento			
IdDepto	Nombre	IdJefeDepto	NumEmp
0001	VENTAS	001690	001693
0002	APT	001850	002694, 003003, 003725
002694	RESGUARDO	000150	004728, 007775

La relación anterior no es 1FN, ya que en la columna de *NumEmp* sólo debe existir un valor del dominio.

Departamento					
IdDepto	Nombre	IdJefeDepto	NumEmp1	NumEmp2	NumEmp3
0001	VENTAS	001690	001693		
0002	APT	001850	002694	003003	003725
002694	RESGUARDO	000150	004728	007775	

La relación anterior tampoco se encuentra en *1FN*, ya que se tienen valores iguales *NumEmp* en distintos dominios.

Para que se encuentre en *1FN* debemos convertir ese atributo en uno *multivaluado*:

Departamento			
IdDepto	Nombre	IdJefeDepto	NumEmp
0001	VENTAS	001690	001693
0002	APT	001850	002694
0002	APT	001850	003003
0002	APT	001850	003725
002694	RESGUARDO	000150	004728
002694	RESGUARDO	000150	007775

Segunda Forma Normal (2FN)

Una relación se encuentra en segunda forma normal cuando está en *1FN* y todo atributo que no sea clave es dependiente irreduciblemente de la *clave primaria*. (Debe aportar información sobre la clave completa)

Esta regla significa que en una relación sólo se debe almacenar información sobre un tipo de entidad, y se traduce en que los atributos que no aporten información directa sobre la *clave principal* deben almacenarse en una relación separada.

Veamos cómo aplicar esta regla usando un ejemplo. En este caso trataremos de guardar datos relacionados con la administración de clientes con sucursales.

Planteemos, por ejemplo, este esquema de relación:

Cliente_Sucursales (NumCliente, Nom_Cliente, Dir_Cliente, NumSucursal, Nom_Sucursal, Dir_Sucursal, Centro_Distribución, Nombre_Centro)

Lo primero que tenemos que hacer es buscar las posibles *claves candidatas*. En este caso sólo existe una posibilidad:

(NumCliente, NumSucursal)

Recordemos que cualquier clave candidata debe identificar de forma unívoca una clave completa. En este caso, la clave completa son las sucursales por cliente, que se define por tres parámetros: el número de cliente (*NumCliente*), el número de sucursal (*NumSucursal*) y el centro de distribución (*Centro_Distribución*).

Es decir, dos sucursales son diferentes si cambian cualquiera de éstos parámetros. El mismo cliente puede tener varias sucursales. Lo que no es posible es que una misma sucursal pertenezca a dos diferentes clientes.

El siguiente paso consiste en buscar los atributos que no aporten información directa sobre la clave completa: las *sucursales por cliente con centro de distribución*. En este caso el *nombre* y la *dirección de la sucursal,* así como el *nombre del cliente* y *Nombre_Centro* no aportan información sobre la clave principal. En realidad, estos datos no son atributos de las *sucursales por cliente con centro de distribución*, sino de la propia s*ucursal*, del *cliente* y del *Centro de Distribución*. Realmente no existe ningún dato el cual requiera una clave que contenga el cliente, centro de distribución y sucursal.

Expresado en forma de dependencias se ve muy claro:

- (Centro_Distribución) -> Nombre_Centro
- (NumCliente) -> Nom_Cliente
- (NumSucursal) -> Centro_Distribución
- (NumSucursal) -> Nom_Sucursal
- (NumSucursal) -> Dir_Sucursal

El último paso consiste en extraer los atributos que no forman parte de la clave de otra relación.

- Cliente (NumCliente, Nom_Cliente)
- Sucursal (NumSucursal, Nom_Sucursal, Dir_Sucursal, Centro_Distribución)
- Centro_Distribución (Centro_Distribución, Nombre_Centro)

Tercera Forma Normal (3FN)

Una relación está en 3FN si y sólo si está en 2FN y los atributos que no son clave son dependientes en forma no transitiva de la clave primaria. Es decir los atributos son:

a. Mutuamente independientes, y

b. Dependientes por completo de la *clave primaria* (dan información sobre la *clave principal completa* y sólo sobre la *clave principal*).

Pero esto es una definición demasiado teórica. En la práctica significa que se debe eliminar cualquier relación que permita llegar a un mismo dato de dos o más formas diferentes.

Forma normal de Boyce-Codd (FNBC o BCFN)

La *Forma Normal de Boyce-Codd* (o FNBC) es una versión ligeramente más fuerte de la Tercera forma normal (*3FN*). La *forma normal de Boyce-Codd* requiere que no existan dependencias funcionales (*DF*) no triviales de los atributos que no sean un conjunto de la clave candidata. En una tabla en *3FN*, todos los atributos dependen de una clave, de la *clave completa* y de ninguna otra cosa excepto de la *clave*.

Se dice que una relación está en FNBC si y solo si está en 3FN y cada dependencia funcional no trivial tiene una única *clave candidata* como *determinante*.

Esto significa que no deben existir interrelaciones entre atributos fuera de las claves candidatas.

Por ejemplo

Pedido (NumPed, OrdCli, FecPed, NumCliente, Nom_Cliente, Total_Artículos)

Un conjunto de datos podría ser el siguiente:

Pedido					
NumPed	OrdCli	FecPed	NumCliente	Nom_Cliente	Total_Artículos
0001	123784	1/10/2010	CMX001	Comercial Mexicana	5,878,454
0002	75936415	12/10/2010	CMX001	Comercial Mexicana	472,111
0003	7421455	13/10/2010	CHE001	Chedrahui	892,544
0004	1256754	16/10/2010	SOR001	Soriana	874,545

Podemos ver que los atributos *No_cliente* y *Nom_cliente* sólo proporcionan información entre ellos mutuamente, pero ninguno de ellos es una clave candidata.

Intuitivamente ya habremos visto que esta estructura puede producir redundancia, sobre todo en el caso de clientes habituales, donde se repetirá la misma información cada vez que el mismo cliente haga un pedido.

Por lo tanto, si queremos que esta relación sea FNBC debemos separarla en dos relaciones distintas:

Pedidos (**NumPed**, OrdCli, FecPed, NumCliente, Total_Artículos)

Clientes (**NumCliente ,** Nom_Cliente)

Cuarta forma normal

Una relación está en *4NF* si y sólo si, en cada dependencia multivaluada *X ->-> Y* no trivial, *X* es clave candidata.

Tomemos por ejemplo la tabla de *Proveedor*, pero dejando sólo los atributos multivaluados:

Proveedor (nombre, teléfono, correo)

Proveedor		
Nombre	Teléfono	Correo
Empaques García	55 54 28 57	ventas@empgarcia.com.mx
Empaques García	56 83 74 15	ventas@empgarcia.com.mx
Plásticos Ideal	56 87 15 42	pedidos@ideal.com.mx
Refacciones industriales	59 87 45 35	ventas@refacindustriales.com.mx
Refacciones industriales	59 87 45 35	clientes@refacindustriales.com.mx

Lo primero que debemos hacer es buscar las claves y las dependencias. Recordemos que las claves candidatas deben identificar de forma unívoca cada tupla. De modo que estamos obligados a usar los tres atributos para formar la clave candidata.

Pero las dependencias que tenemos son:

nombre ->-> teléfono

nombre ->-> correo

Y *nombre* no es clave candidata de esta relación.

Resumiendo, debemos separar esta relación en varias (tantas como atributos multivaluados tenga).

Proveedor_Teléfonos (nombre, teléfono)

Proveedor_Correos (nombre, correo)

Ahora en las dos relaciones se cumple la *cuarta forma normal*.

Quinta forma normal (5FN)

La quinta forma normal (5FN), también conocida como forma normal de proyección-unión (PJ/NF), es un nivel de normalización de bases de datos designado para reducir redundancia en las bases de datos relacionales que guardan hechos multivaluados, aislando semánticamente relaciones múltiples relacionadas.

La *quinta forma normal* maneja con éxito los casos en que la información puede ser reconstruida a partir de pequeñas piezas de información que se puede mantener con menos redundancia. Las formas normales de *2FN, 3FN y 4FN* también sirven a este propósito, pero la *quinta forma normal* se generaliza en casos no previstos por las demás.

Una tabla se encuentra en *5FN* si:

- La tabla está en *4FN*.
- No existen relaciones de dependencias no triviales que no siguen los criterios de las claves. Una tabla que se encuentra en la *4FN* se dice que está en la *5FN* si, y sólo si, cada relación de dependencia (join) se encuentra definida por las claves candidatas.

Ejemplo:

Si cada cliente tiene un agente y un cliente solicita diferentes departamentos, y los agentes están encargados de diferentes departamentos, entonces puede ser que desee mantener un registro de cuál es el agente que vende productos para cuales empresas. Esta información puede ser guardada en una tabla con tres campos:

AGENTE	CLIENTE	DEPARTAMENTO
Carlos	Comercial	Caballeros
Carlos	Chedrahui	Damas

Esta forma es necesaria para un caso general. Por ejemplo, aunque el agente *Carlos* le vende a la *Comercial* y a *Chedraui*, no tiene que vender productos de *Caballero* a *Comercial* o de *Damas* a *Chedraui.* Así pues, necesitamos la combinación de tres campos para saber qué combinaciones son válidas y cuáles no lo son.

Pero supongamos que hay una nueva regla: si un *Agente* vende un determinado *Departamento*, y representa a un *Cliente*, es el responsable de vender ese Departamento a esa empresa.

AGENTE	CLIENTE	DEPARTAMENTO
Carlos	Comercial	Caballeros
Carlos	Comercial	Damas
Carlos	Walmart	Caballeros
Carlos	Walmart	Damas
Adriana	Chedrahui	Caballeros
Adriana	Chedrahui	Damas
Adriana	Chedrahui	Niños
Jorge	Walmart	Damas
Jorge	Walmart	Niños
Jorge	Comercial	Damas
Jorge	Comercial	Niños

En este caso, resulta que se puede reconstruir todos los hechos reales a partir de un formulario normalizado que consta de tres tipos de registros distintos, cada uno con dos campos:

AGENTE	CLIENTE
Carlos	Comercial
Carlos	Walmart
Adriana	Chedrahui
Jorge	Walmart
Jorge	Comercial

CLIENTE	DEPARTAMENTO
Comercial	Caballeros
Comercial	Damas
Comercial	Niños
Chedrahui	Caballeros
Chedrahui	Damas
Chedrahui	Niños
Walmart	Damas
Walmart	Niños

AGENTE	DEPARTAMENTO
Carlos	Caballeros
Carlos	Damas
Adriana	Caballeros
Adriana	Damas
Adriana	Niños
Jorge	Damas
Jorge	Niños

Estos tres tipos de registro están en quinta forma normal, mientras que la correspondiente de tres campos de registro mostrado anteriormente no lo está.

En términos generales, podemos decir que un registro se encuentra en la quinta forma normal, cuando su

contenido no puede ser reconstruido a partir de varios tipos de registros más pequeños, es decir, de cada uno de los tipos de registros que tienen menos campos que el registro original. El caso en que todos los registros más pequeños tienen la misma clave está excluido. Si un tipo de registro sólo se puede descomponer en pequeños registros que tienen la misma clave, entonces se considera que el registro esta en 5FN sin descomposición.

La quinta forma normal no se diferencia de la cuarta forma normal a menos que exista una restricción simétrica, como la regla de los *Agentes*, *Clientes* y *Departamentos*. En ausencia de tal restricción, un tipo de registro en forma normal cuarto está siempre en quinta forma normal.

Una ventaja de quinta forma normal es que ciertas redundancias pueden ser eliminadas. En el formulario normalizado, el hecho de que *Jorge* nada más venda *Damas* y *Niños* se registra una sola vez. En una forma no normalizada se tendría que registrar varias veces.

Debe observarse que, aunque la forma normalizada consiste en varios tipos de registro, puede haber un menor número de ocurrencias de registros totales. Esto no es evidente cuando sólo hay unos pocos registros. Sin embargo se hace más evidente en la medida que aumenta el tamaño, ya que el tamaño de los incrementos normalizados contenidos se realiza de forma aditiva, mientras que el tamaño de los incrementos de archivos no normalizados se realiza de manera multiplicativa .Por ejemplo, si se añade un nuevo agente que vende X *Departamentos* para los Clientes Y, hay que sumar X + Y nuevos registros a la forma normalizada, pero XY nuevos registros a la forma no normalizada.

Cabe señalar que los tres tipos de registro son necesarios en la forma normalizada con el fin de reconstruir la misma información. A partir de los tipos de registros que se muestran arriba nos enteramos que *Carlos* representa a *Comercial* y que *Comercial* solicita artículos del Departamento de Niños. Pero no podemos determinar si *Carlos* vende artículos de *n*iños a *Comercial* hasta que nos fijamos en el tipo de registro para determinar si *Carlos* vende artículos del Departamento de *Niños*.

Desnormalización

Tanto en esta forma normal, como en las próximas que veremos, es importante no llevar el proceso de normalización demasiado lejos. Se trata de facilitar el trabajo y evitar problemas de redundancia e integridad y no de complicarlo. Debemos considerar la necesidad o ventajas de aplicar la norma en cada caso, y no excedernos al aplicarlas demasiado al pie de la letra.

1. La normalización total significa muchas relaciones separadas lógicamente (y aquí damos por hecho que las relaciones en cuestión son específicamente relaciones base)

2. Muchas relaciones separadas lógicamente significan muchos archivos almacenados que están separados físicamente

3. Muchos archivos almacenados separados físicamente significan una gran cantidad de operaciones de E/S.

Desde luego, de manera estricta este argumento no es válido, ya que el modelo relacional no estipula en ninguna parte que las entidades deban asociarse una a una con los archivos almacenados.

Pero el argumento si es válido para los productos *SQL* actuales debido, precisamente, al grado inadecuado de separación, entre esos dos niveles, que encontramos en dichos productos. (Date, 2001)

Definición

Podemos definir la desnormalización como sigue. Sea Rl ..... Rn un conjunto de relaciones. Entonces desnormalizar esas relaciones significa remplazarlas por su junta (digamos) R, tal que para todos los valores posibles rl, ..., rn de Rl, ..., Rn al proyectar el valor r correspondiente de R sobre los atributos de Ri, se garantiza que, se obtiene de nuevo ri (i = 1,..., ri), El objetivo general consiste en aumentar la redundancia, asegurando que R esté a un nivel menor de normalización que las relaciones Rl, ..., Rn.

En forma más específica, el objetivo es reducir el número de juntas que deben ser realizadas en tiempo de ejecución haciendo (en efecto) algunas de esas juntas antes de tiempo, como parte del diseño o de la base de datos.

Algunos problemas

El concepto de desnormalización padece diversos problemas bien conocidos. Uno, obvio, es que una vez que comenzamos a realizar la desnormalización, no está claro cuándo debemos detenernos. Con la normalización hay razones claramente lógicas para continuar hasta alcanzar la forma normal más alta posible; ¿concluimos entonces que con la desnormalización debemos proseguir hasta alcanzar la forma normal más baja posible? Por supuesto que no, aunque no hay criterios lógicos para decidir exactamente dónde parar. En otras palabras, al seleccionar la desnormalización estamos retrocediendo de una posición que por lo menos tiene cierta ciencia sólida y una teoría lógica que la apoyan, a una que es de naturaleza puramente pragmática y subjetiva.

El segundo inconveniente, obvio también, es que hay redundancia y problemas de actualización, precisamente porque estamos tratando, una vez más, con relaciones que no están completamente normalizadas. Ya hemos expuesto ampliamente éstos aspectos. Sin embargo, lo que es menos obvio es que, también podría haber problemas de recuperación; es decir, la desnormalización, en realidad, puede hacer que ciertas consultas sean más difíciles de expresar.

El tercer problema, y el más importante, es que, cuando decimos que la desnormalización "es buena para el rendimiento", en realidad queremos decir que es buena para el rendimiento de aplicaciones específicas (esto se aplica a la desnormalización "propia" —es decir, a la desnormalización que se hace sólo al nivel físico—y al tipo de desnormalización que en ocasiones tiene que hacerse en los productos *SQL* actuales).

2.1.6. *LAS 12 REGLAS DE CODD*

En 1984 Codd publicó dos artículos *"Is Your DBMS Really Relational?"* y *"Does Your DBMS Run By the Rules"* que pretendían mantener la integridad del modelo relacional y dejar claro que la colocación de una interfaz de usuario relacional, en la parte superior de un sistema que utiliza algún otro modelo como su modelo de datos básico no es suficiente para que un SGBD sea realmente relacional. (Ricardo, 2004) El plantea que para que un SGBD se considere completamente relacional debe de seguir 12 reglas, más una regla fundamental llamada regla cero. En la práctica algunas de ellas son difíciles de realizar, y han causado mucha polémica a lo largo de los años.

Regla 0

Para que un sistema se denomine sistema de gestión de bases de datos relacionales, este sistema debe usar (exclusivamente) sus capacidades relacionales para gestionar la base de datos.

Regla 1: Regla de la información

Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas.

- Por tanto los metadatos (diccionario, catálogo) se representan exactamente igual que los datos de usuario.
- Y puede usarse el mismo lenguaje (ej. SQL) para acceder a los datos y a los metadatos (regla 4)
- Un valor posible es el valor nulo, con sus dos interpretaciones:
    - Valor desconocido (ej. dirección desconocida)
    - Valor no aplicable (ej. empleado soltero no tiene esposa).

Regla 2: Regla de acceso garantizado

Para todos y cada uno de los datos (valores atómicos) de una base de datos relacional se garantiza que son accesibles a nivel lógico utilizando una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

Cualquier dato almacenado en una *base de datos relacional* tiene que poder ser direccionado unívocamente. Para ello hay que indicar en qué tabla está, cuál es la columna y cuál es la fila (mediante la clave primaria).

- Por tanto se necesita el concepto de clave primaria, que no es soportado en muchas implementaciones. En éstos casos, para lograr un efecto similar se puede hacer lo siguiente:
    - Hacer que los atributos clave primaria no puedan ser nulos (NOT NULL).
    - Crear un índice único sobre la clave primaria.
    - No eliminar nunca el índice.

Regla 3: Tratamiento sistemático de valores nulos

Los valores nulos (que son distintos de la cadena vacía, blancos, 0, ...) se soportan en los SGBD totalmente relacionales para representar información desconocida o no aplicable de manera sistemática, independientemente del tipo de datos.

Se reconoce la necesidad de la existencia de valores nulos, para un tratamiento sistemático de los mismos.

- Hay problemas para soportar los valores nulos en las operaciones relacionales, especialmente en las operaciones lógicas.

Lógica trivaluada. Es una posible solución. Existen tres (no dos) valores de verdad: Verdadero, Falso y Desconocido (null). Se crean tablas de verdad para las operaciones lógicas:

$$Null \bigcap Null = Null$$
$$Null \bigcap Verdadero = Null$$
$$Null \bigcap Falso = Falso$$
$$Verdadero \bigcup Null = Verdadero$$

Un inconveniente es que de cara al usuario el manejo de los lenguajes relacionales se complica pues es más difícil de entender.

Regla 4: Catálogo dinámico en línea basado en el modelo relacional

La descripción de la base de datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, igual que lo aplican a los datos normales.

Es una consecuencia de la regla 1 que se destaca por su importancia. Los metadatos se almacenan usando el modelo relacional, con todas las consecuencias.

Regla 5: Sublenguaje de datos completo

Un sistema relacional debe soportar varios lenguajes y varios modos de uso de terminal (ej.: rellenar formularios, etc.). Sin embargo, debe existir al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida, como cadenas de caracteres y que sea completo, soportando:

- Definición de datos
- Definición de vistas
- Manipulación de datos (interactiva y por programa)
- Limitantes de integridad
- Limitantes de transacción (iniciar, realizar, deshacer) (Begin, commit, rollback).

Además de poder tener interfaces más amigables para hacer consultas, etc. siempre debe de haber una manera de hacerlo todo de manera textual, que es tanto como decir que pueda ser incorporada en un programa tradicional.

Un lenguaje que cumple esto en gran medida es SQL.

Regla 6: Actualización de vistas

Todas las vistas que son teóricamente actualizables se pueden actualizar por el sistema.

- El problema es determinar cuáles son las vistas teóricamente actualizables, ya que no está muy claro.
- Cada sistema puede hacer unas suposiciones particulares sobre las vistas que son actualizables.

Regla 7: Inserción, actualización y borrado de alto nivel

La capacidad de manejar una relación base o derivada como un solo operando se aplica no sólo a la recuperación de los datos (consultas), sino también a la inserción, actualización y borrado de datos.

Esto es, el lenguaje de manejo de datos también debe ser de alto nivel (de conjuntos). Algunas bases de datos inicialmente sólo podían modificar las tuplas de la base de datos de una en una (un registro de cada vez).

Regla 8: Independencia física de datos

Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cuando quiera que se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

El modelo relacional es un modelo lógico de datos, y oculta las características de su representación física.

Regla 9: Independencia lógica de datos

Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cuando quiera que se realicen cambios a las tablas base que preserven la información.

Cuando se modifica el esquema lógico preservando información (no valdría p.ej. eliminar un atributo) no es necesario modificar nada en niveles superiores.

Ejemplos de cambios que preservan la información:

- Añadir un atributo a una tabla base.
- Sustituir dos tablas base por la unión de las mismas. Usando vistas de la unión puedo recrear las tablas anteriores...

Regla 10: Independencia de integridad

Los limitantes de integridad específicos para una determinada base de datos relacional deben poder ser definidos en el lenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.

El objetivo de las bases de datos no es sólo almacenar los datos, sino también sus relaciones y evitar que estas (limitantes) se codifiquen en los programas. Por tanto en una *base de datos relacional* se deben poder definir limitantes de integridad.

Cada vez se van ampliando más los tipos de limitantes de integridad que se pueden utilizar en los SGBD, aunque hasta hace poco eran muy escasos.

Como parte de los limitantes inherentes al modelo relacional (forman parte de su definición) están:

- Una *base de datos relacional* tiene **integridad de entidad**. Es decir, toda tabla debe tener una clave primaria.
- Una *base de datos relacional* tiene **integridad referencial**. Es decir, para toda clave externa no nula debe existir en la relación donde es primaria.

Regla 11: Independencia de distribución

Una base de datos relacional tiene independencia de distribución.

Las mismas órdenes y programas se ejecutan igual en una BD centralizada que en una distribuida.

Las *bases de datos relacionales* son fácilmente distribuibles:

- Se parten las tablas en fragmentos que se distribuyen.
- Cuando se necesitan las tablas completas se recombinan usando operaciones relacionales con los fragmentos.
- Sin embargo se complica más la gestión interna de la integridad, etc.
- Esta regla es responsable de tres tipos de **transparencia de distribución**:
    - Transparencia de localización. El usuario tiene la impresión de que trabaja con una BD local. (aspecto de la regla de independencia física)
    - Transparencia de fragmentación. El usuario no se da cuenta de que la relación con que trabaja está fragmentada. (aspecto de la regla de independencia lógica de datos).
    - Transparencia de replicación. El usuario no se da cuenta de que pueden existir copias (réplicas) de una misma relación en diferentes lugares.

Regla 12: Regla de la no subversión

Si un sistema relacional tiene un lenguaje de bajo nivel (un registro de cada vez), ese bajo nivel no puede ser usado para saltarse (subvertir) las reglas de integridad y los limitantes expresados en los lenguajes relacionales de más alto nivel (una relación (conjunto de registros) de cada vez).

Algunos problemas no se pueden solucionar directamente con el lenguaje de alto nivel.

2.2. SERVICIOS WEB

La programación orientada a componentes se ha vuelto muy popular y, dado que las aplicaciones se han vuelto más sofisticadas, la necesidad de aprovechar componentes distribuidos en las máquinas remotas también ha crecido.

Un ejemplo de aplicaciones basadas en componentes puede observarse en algunas soluciones de comercio electrónico cuando un cliente, cuya aplicación reside en un conjunto de servidores Web, debe intercambiar información comercial -como facturación, *Avisos Anticipados de Embarque,* etcétera- con un ERP residente en un hardware y un sistema operativo diferente.

Se han generado diferentes alternativas para dar solución a este tipo de problema. Por ejemplo, *Microsoft Distributed Object Model* (DCOM) que es una infraestructura que permite a una aplicación invocar un objeto COM (*Component Object Model*) instalado en otro servidor. No obstante, esta solución no ganado suficiente aceptación en plataformas no *Windows*, por lo que rara vez se utiliza para facilitar la comunicación entre ordenadores *Windows* y no *Windows*. Además DCOM y las tecnologías relacionadas como CORBA y *Java* RMI se limitan a las aplicaciones y componentes instalados en el centro de datos corporativo. Los clientes que se comunican con el servidor a través de Internet se enfrentan a numerosos obstáculos potenciales al comunicarse con el servidor; uno es el bloqueo de puertos, y el otro, es que no pueden manejar las interrupciones de red con gracia debido a que Internet no está bajo su control directo. Esto implica que no se puede hacer ninguna suposición sobre la calidad o fiabilidad de la conexión y que, si hay una interrupción en la red, la comunicación entre el cliente y el servidor puede fallar. Los protocolos propietarios, como los utilizados por DCOM, CORBA y *Java* RMI, no son prácticos para escenarios de Internet.

Se hizo evidente que se necesitaba un nuevo enfoque que cumpliera con los siguientes requerimientos: (Short, 2002)

- ***Interoperabilidad:*** El servicio remoto debe ser capaz de ser utilizado por los clientes en otras plataformas.

- ***Amigable con Internet:*** La solución debe apoyar a los clientes que utilizan el servicio remoto a través de Internet.

- ***Comunicación solida:*** No debe haber ninguna ambigüedad sobre el tipo de datos que se envían y reciben de un servicio remoto. Además, los tipos de datos definidos en el servicio remoto deben corresponder razonablemente bien a los tipos de datos de la mayoría de los lenguajes de programación procedimentales.

- ***Capacidad para aprovechar los estándares de Internet existentes:*** La implementación del servicio remoto debe aprovechar los estándares existentes de Internet tanto como sea posible y evitar reinventar soluciones a los problemas que ya han sido resueltos. Una solución construida sobre un estándar de Internet ampliamente adoptado puede aprovechar conjuntos de herramientas y productos existentes creados para dicha tecnología.

- ***Soporte para cualquier lenguaje:*** La solución no debe ligarse a un lenguaje de programación particular. *Java* RMI, por ejemplo, está ligada completamente a lenguaje *Java*. Sería muy difícil invocar funcionalidad de un objeto *Java* remoto desde *Visual Basic* o *PERL*. Un cliente debe ser

capaz de implementar un nuevo servicio Web existente independientemente del lenguaje de programación en el que se halla escrito.

- ***Soporte para cualquier infraestructura de componente distribuido:*** La solución no debe estar fuertemente ligada a una infraestructura de componentes en particular. De hecho, no debería ser necesario comprar, instalar o mantener una infraestructura de objetos distribuidos, sino solo construir un nuevo servicio remoto utilizando un servicio existente. Los protocolos subyacentes deberían proporcionar un nivel base de comunicación entre la infraestructura de los objetos distribuidos existentes tales como DCOM y CORBA.

Bloques de construcción básicos necesarios para facilitar la comunicación remota

El siguiente gráfico muestra los bloques de construcción básicos necesarios para facilitar la comunicación entre dos aplicaciones remotas mediante servicios Web.

FIG. 19 BLOQUES DE CONSTRUCCIÓN BÁSICOS DE UN SERVICIO WEB

- ***Descubrimiento:*** La aplicación cliente, que necesita acceder a la funcionalidad que expone un *Servicio Web*, necesita una forma de resolver la ubicación de servicio remoto. Se logra mediante un proceso llamado, *descubrimiento* o en ingles *Discovery*. El *descubrimiento* se puede proporcionar mediante un directorio centralizado así como por otros métodos ad hoc. En DCOM, el servicio de descubrimiento lo proporciona el Administrador de control de servicios (SCM, *Services Control Manager*).

- ***Descripción:*** Una vez que se ha resuelto el extremo de un *servicio Web* dado, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La *descripción* de un *servicio Web* implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobré el *servicio Web* incluyendo ejemplo de uso. Un componente DCOM expone meta datos estructurados sobre sus interfaces mediante una biblioteca de tipo (*typelib*). Los meta datos dentro de una *typelib* de componente se guardan en un

formato binario propietario a los que se accede mediante una interfaz de programación de aplicación *(API)* propietaria.

- ***Formato del mensaje***: Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje.
  El uso de un mecanismo estándar para codificar los datos asegura que los datos que codifica el cliente serán interpretados correctamente por el servidor. En *DCOM* los mensajes que se envían entre un cliente y un servidor tienen un formato definido por el protocolo *DCOM Object RPC* (ORPC).

- ***Codificación:*** Los datos que se trasmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensaje. DCOM utiliza un esquema de codificación binaria para serializar los datos de los parámetros que se intercambian entre el cliente y el servidor.

- ***Transporte:*** Una vez se ha dado formato al mensaje y se han serializado los datos en el cuerpo del mensaje se debe transferir entre el cliente y el servidor utilizando algún protocolo de transporte. DCOM dispone de varios protocolos propietarios como *TCP, SPX, NetBEUI y NetBIOS* sobre I*PX*.[38] (Short, 2002)

Vamos a discutir algunas de las decisiones de diseño detrás de éstos bloques de construcción para los *servicios Web*.

Selección de protocolos de transporte

Como mencioné anteriormente, las tecnologías como DCOM, CORBA y *Java* RMI son poco adecuadas para la comunicación entre el cliente y el servidor, a través de Internet. Tanto *Protocolos de transferencia de hipertexto* (HTTP) como *Simple Mail Transfer Protocol* (SMTP) son protocolos de Internet probados. HTTP define un patrón de mensajes de solicitud / respuesta para la presentación de una solicitud y la obtención de una respuesta asociada. SMTP define un protocolo enrutable de mensajería para la comunicación asíncrona. Vamos a examinar por qué HTTP y SMTP son protocolos muy adecuados para la Internet.

Las aplicaciones web que se basan en HTTP inherentemente se encuentran sin estado. No se basan en una conexión continua entre el cliente y el servidor. Esto hace que el protocolo HTTP sea ideal para configuraciones de alta disponibilidad, tales como firewalls. Si el servidor original del cliente que maneja la solicitud no está disponible, las solicitudes posteriores pueden ser automáticamente enrutadas a otro servidor sin que el cliente se percate.

En el caso de SMTP, casi todas las empresas tienen una infraestructura que admite este protocolo. Además SMTP es adecuado para la comunicación asíncrona. Si el servicio se interrumpe, la infraestructura de correo electrónico maneja automáticamente reintentos.

A diferencia de HTTP, puede pasar los mensajes SMTP a un servidor local de correo que se intentará entregar el mensaje de correo electrónico en su nombre.

La otra ventaja importante de HTTP y SMTP es su omnipresencia. Tecnologías como la *Traducción de Dirección de Red (*NAT por sus siglas en ingles *Network Address Translation*) y los servidores *Proxy* proporcionan un medio para acceder a Internet a través de HTTP de dentro de redes LAN corporativas

[38] Ver Anexo III Ejemplos SOAP y WSDL http://www.w3.org/2001/03/14-annotated-WSDL-examples.html

de otra manera aislada. Los administradores de redes a menudo exponen los servidores SMTP que reside dentro del firewall. Los mensajes a este servidor, son enviados a su destino final a través de Internet.

La otra ventaja importante de HTTP y SMTP es su omnipresencia. Tecnologías como la *Traducción de Dirección de Red (*NAT por sus siglas en ingles *Network Address Translation*) y los servidores *Proxy* proporcionan un medio para acceder a Internet a través de HTTP desde adentro de redes LAN corporativas de otra manera aislada. Los administradores de redes a menudo exponen los servidores SMTP a accesos remotos mediante internet aunque dichos servidores residen dentro de un Firewall de sitio en el que se encuentran. Los mensajes de correo electrónico son enviados de las computadoras que se encuentran dentro de la red hacia el servidor de correos, y a través de este son transmitidos a su destino final mediante Internet.

La mayoría de los paquetes de software ERP no son capaces de manejar grandes volúmenes de órdenes que procedan del comercio electrónico, y pueden ser potencialmente expulsados de la aplicación por un Firewall. Además, no es imperativo que las órdenes se desplieguen en el sistema ERP en tiempo real. SMTP puede ir ordenando las órdenes en cola de manera que puedan ser procesadas en serie por el sistema ERP. (Short, 2002)

Elección de un esquema de codificación

HTTP y SMTP proporcionan un medio de envío de datos entre el cliente y el servidor. Sin embargo, no especifica cómo deben ser codificados los datos dentro del cuerpo del mensaje.

Microsoft necesitaba una norma que permitiera codificar los datos intercambiados entre el cliente y el servidor independientemente de la plataforma.

Debido a que el objetivo era aprovechar Internet, los protocolos XML (en ingles, *e**X**tensible **M**arkup **L**anguage*) fue la elección natural. XML ofrece muchas ventajas, incluyendo soporte multi-plataforma, un sistema común, y soporte para los caracteres estándar de la industria.

XML evita problemas de codificación binaria, ya que utiliza un esquema de codificación basado en texto que aprovecha los conjuntos de caracteres estándar. Además, algunos protocolos de transporte, tales como SMTP, contienen sólo mensajes basados en texto.

Métodos de codificación binaria, tales como los utilizados por DCOM y CORBA, son engorrosos y requieren una infraestructura de apoyo para abstraer al programador de los detalles. XML es menos pesado y más fácil de manejar, ya que puede ser creado y consumido utilizando estándares de análisis de texto.

Además, una variedad de analizadores XML están disponibles para simplificar aún más la creación y el consumo de los documentos XML en prácticamente todas las plataformas modernas. Permitiéndole un alcance increíble, porque prácticamente cualquier cliente en cualquier plataforma puede comunicarse con el *servicio Web.* (Short, 2002)

Elección de un convenio de formato

A menudo es necesario incluir metadatos adicionales con el cuerpo del mensaje. Para ejemplo, es posible que desee incluir información sobre el tipo de servicios que un sitio de *servicios Web* debe proporcionar a fin de cumplir con su solicitud, tales como inscripción en una transacción o información de enrutamiento. XML no proporciona mecanismo alguno para diferenciar el cuerpo del mensaje a partir de sus datos asociados.

Los protocolos de transporte tales como HTTP proporcionan un mecanismo extensible para los datos de cabecera, pero algunos datos relacionados con el mensaje podrían no ser especificados para el protocolo de transporte. Para ejemplo, el cliente puede enviar un mensaje que debe ser remitido a varios destinos, posiblemente a través de diferentes protocolos de transporte. Si la información de enrutamiento se colocara en la cabecera del HTTP, tendría que ser traducidos antes de ser enviado siguiente protocolo de transporte, como SMTP. Debido a que la información de enrutamiento es específica para el mensaje y no el protocolo de transporte, debería ser una parte del mensaje.

Simple Object Access Protocol (SOAP) proporciona un medio independiente al protocolo que permite la asociación de cabecera con el cuerpo del mensaje.

Todos los mensajes SOAP deben definir una envolvente que contenga la carga útil del mensaje y un encabezado con los metadatos asociados.

SOAP no impone ninguna restricción sobre el formato del cuerpo del mensaje. Esta es una preocupación potencial porque sin una forma coherente de codificar los datos, es difícil desarrollar un conjunto de herramientas que permitan abstraer los datos de los protocolos subyacentes. Por lo tanto se necesitaba una forma estándar de dar formato a un mensaje mediante una llamada de procedimiento remoto (*RPC*) que permitiera codificar su lista de parámetros. Esto es exactamente lo que proporciona la especificación de la Sección 7 de SOAP. En la que se describe una convención de nomenclatura y el estilo de codificación estándar para mensajes orientados a procedimientos.

Debido a que *SOAP* proporciona un formato estándar para serializar los datos en un mensaje *XML*, plataformas como *ASP.Net* y *Remoting* puede abstraer los detalles para uno. (Short, 2002)

Selección de los mecanismos descripción

SOAP proporciona una forma estándar para dar formato a los mensajes intercambiados entre el *servicio Web* y el cliente. Sin embargo, el cliente necesita información adicional para poder analizar adecuadamente la serialización de la solicitud e interpretar la respuesta. El esquema de *XML* es un medio que permite crear esquemas que pueden ser utilizados para describir el contenido de un mensaje.

El Esquema de *XML* proporciona un conjunto básico de tipos de datos incorporados que se pueden utilizar para describir el contenido de un mensaje. También se pueden crear tipos propios de datos. Por ejemplo, el comerciante de un banco puede crear un tipo de datos complejo para describir el contenido y la estructura del cuerpo de un mensaje que se utiliza para enviar una solicitud de pago a una tarjeta de crédito.

Un esquema contiene una serie de definiciones de tipos de datos y de elementos. Un *servicio web* utiliza el esquema no sólo para comunicar el tipo de datos que se espera se encuentre dentro de un mensaje sino también para validar los mensajes entrantes y salientes.

Un esquema por sí solo no proporciona suficiente información para describir con eficacia un *servicio Web*. El esquema no describe los patrones de mensajes entre el cliente y el servidor.

Por ejemplo, un cliente tiene que saber si va a esperar una respuesta del sistema ERP cuando una orden se contabiliza en este. Debe también conocer cuál es el protocolo de transporte que el *Servicio Web* espera para recibir las solicitudes. Por último, el cliente necesita saber la dirección donde el *Servicio Web* puede ser alcanzado.

Esta información es proporcionada por WSDL (*Web Services Description Language*). WSDL es un documento XML que describe completamente un *servicio Web* en particular describiendo los requisitos

del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Herramientas tales como *ASP.Net, WSDL.exe, Soapsuds.exe* y *Remoting* puede interpretar WSDL y automáticamente construir canales de comunicación para el desarrollador. (Short, 2002)

Selección de mecanismos de detección

Después de desarrollado y documentado un *servicio Web* la atención debe dirigirse a la búsqueda de clientes potenciales. Cuando estos clientes se encuentran en la Internet, el *servicio Web* debe publicitarse de manera efectiva y formal. El *Universal Description, Discovery and Integration* (UDDI) posibilita esta acción al ofrecer un directorio, centralizado y estándar, de la industria que puede ser utilizado para anunciar y localizar *servicios Web*. Este sistema permite buscar *servicios Web* de una serie de criterios de búsqueda que incluyen nombre de la empresa, categoría y tipo de servicio.

También se puede hacer publicidad de servicios Web a través de *DISCO* que es un formato para documentos propietarios en *XML,* definido por Microsoft, y que permite a los sitios Web anunciar los servicios que ofrecen. *DISCO* define un protocolo sencillo para facilitar un estilo de hipervínculo para la localización de recursos. El principal consumidor de *DISCO* es *Microsoft Visual Studio. NET*.

2.3. TRAZABILIDAD

Según GS1:[39]

La trazabilidad es la capacidad de seguir una unidad de producto a lo largo de la cadena de suministros. Son aquellos procedimientos preestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto o lote de productos a lo largo de la cadena de suministros en un momento dado, a través de unas herramientas determinadas.

El término trazabilidad es definido por la Organización Internacional para la Estandarización (ISO):[40]

La propiedad del resultado de una medida o del valor de un estándar donde éste pueda estar relacionado con referencias especificadas, usualmente estándares nacionales o internacionales, a través de una cadena continúa de comparaciones todas con incertidumbres especificadas.

Podemos inferir que la trazabilidad requiere un método que permita relacionar el producto con todas las variables involucradas en el proceso de su producción y comercialización, al igual que la participación de los actores en dicha cadena de abastecimiento. Ello implica la identificación de todas las configuraciones de empaquetado de producto y transporte en todas las fases de la cadena de suministros.

Para poder seguir un producto deben aplicarse y registrarse números de identificación, lotes, etcétera, en toda la cadena de suministros de forma que se garantice una relación entre ellos y los datos relativos a la trazabilidad de ese producto. El procedimiento debe permitir conocer desde el origen de las materias primas involucradas en un determinado producto hasta la unidad de venta, pasando por los procesos intermedios de la cadena de suministros.

Es responsabilidad de cada empresa conocer que productos han sido expedidos al siguiente agente de la cadena de suministros y los procesos realizados. Las herramientas de codificación estándar GS1 facilitan la trazabilidad de los productos mediante sistemas automáticos de lectura de código de barras, pero es responsabilidad de cada una de las empresas adaptar sus procesos para facilitar este procedimiento.

Base de un sistema de trazabilidad

Desde el punto de vista de gestión de la información, la trazabilidad consiste en asociar sistemáticamente un flujo de información a un flujo físico de mercancías de manera que se pueda tener un seguimiento de los lotes o grupos de productos específicos en un instante determinado.

Con el fin de asegurar la continuidad de un flujo de información, cada actor debe comunicar al actor siguiente en la cadena de abastecimiento los identificadores de los lotes o grupos de productos trazados que permitirán a este último aplicar los principios de base de trazabilidad. A esta identificación se añade información complementaria disponible en cada eslabón de la cadena. (GS1, 2011)

Niveles de un sistema de trazabilidad: (GS1, 2011)

Un modelo de trazabilidad se encuentra dividido en tres partes fundamentales que se citan a continuación.

[39] GS1 (*Global System, Global Standard y Global Solution)* es una organización privada global sin fines de lucro dedicada a la elaboración y aplicación de normas mundiales y soluciones para mejorar la eficiencia y visibilidad de las cadenas de abastecimiento, la oferta y la demanda a nivel mundial y en todos los sectores. En el año 2005 la asociación EAN (European Article Number) se ha fusionado con la UCC (Uniform Code Council) para formar una nueva y única organización mundial identificada como GS1. El sistema de normas GS1 es el más ampliamente utilizado en la cadena de suministro en el mundo.

[40] ISO 30:1992, 3.8

1. La información que se añada en una etiqueta de códigos de barras directamente sobre la mercancía y que viaja físicamente con ella. Parte de esta información irá en código de barras, para permitir su lectura automática por un lector de código de barras, y otra parte irá en caracteres humanamente legibles para poder hacer un control visual si fuera necesario.
2. La información que se trasmite vía electrónica entre un agente de la cadena de suministros y el siguiente.
3. La información que debe de ser almacenada en la base de datos de cada actor de la cadena de suministros para ser rescatada en caso de necesidad.

Dado que la trazabilidad no es un proyecto individual de una sola empresa sino que relaciona varias empresas a lo largo de la cadena de suministros, el sistema de transmisión de información entre los distintos agentes de la cadena debe ser único. Los estándares GS1 de identificación de mercancía y de transición electrónica de datos, facilitan esa misión.

Partes de un sistema de trazabilidad:

El concepto de trazabilidad se divide en dos partes bien diferenciadas que nos permiten entender cómo se mueve un producto a través de su cadena de suministro o de su rama logística,

La *Trazabilidad Interna* o de *seguimiento del producto (tracking):* Es la capacidad de seguir el curso de una unidad, grupo o lote de productos a través de la cadena de abastecimiento, así como todos los indicios que hacen o pueden hacer variar el producto para el consumidor final.

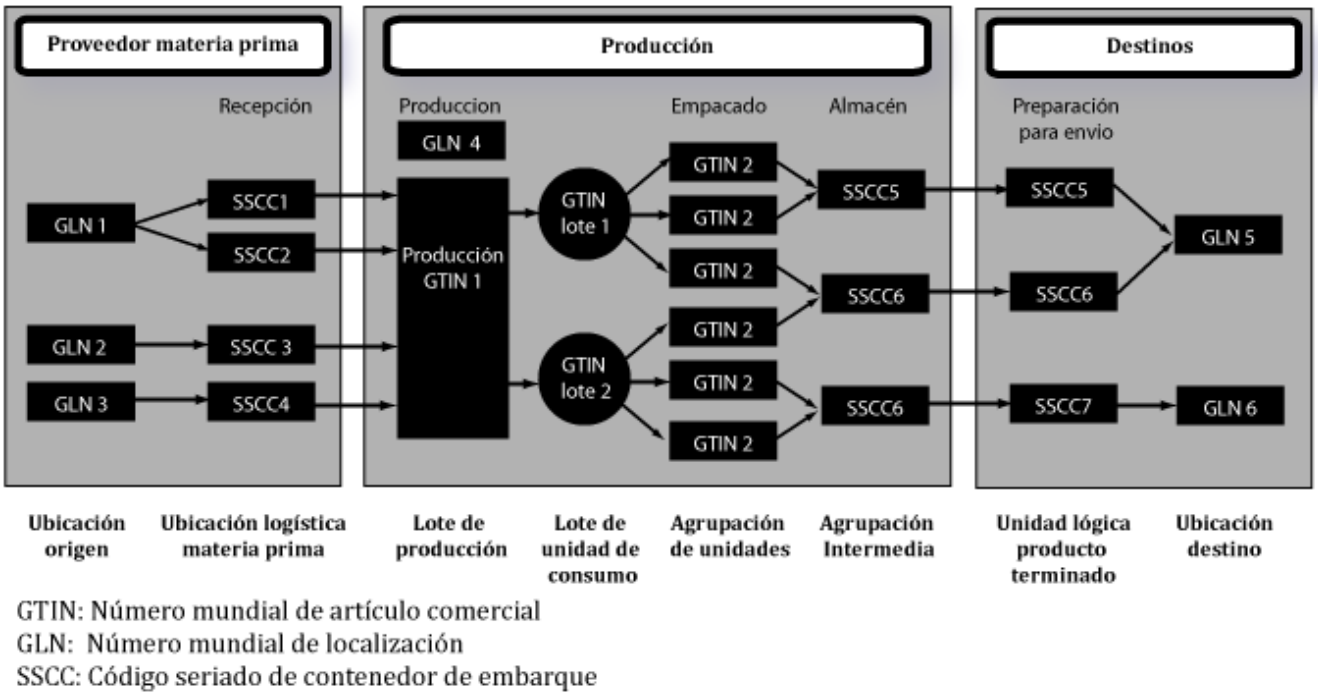


FIG. 20 MODELO DE TRAZABILIDAD INTERNA

La *Trazabilidad Externa* o *Rastreo del producto (tracking):* Es la capacidad de identificar el origen de una unidad en particular, de un grupo o lote de productos ubicados dentro de la cadena de abastecimiento. Externalizando los datos de la traza interna y añadirle algunos indicios más si fuera necesario.

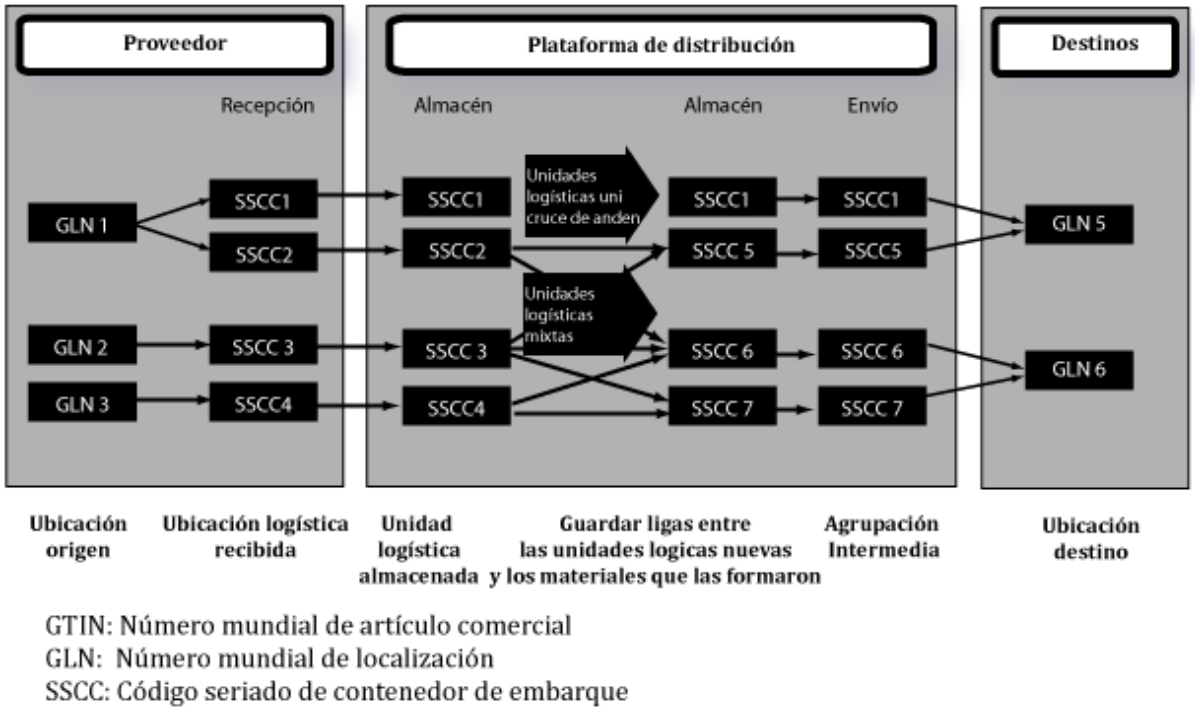


FIG. 21 MODELO DE TRAZABILIDAD EXTERNA

Como consecuencia vemos que para obtener la trazabilidad de un producto, hay que ir registrando los indicios que va dejando el producto mientras se mueve por la cadena, ya sea en el sentido normal o en el sentido inverso (como la logística inversa). (GS1, 2011)

Ventajas de la trazabilidad

La trazabilidad se aplica y se justifica por los beneficios que aporta a los negocios: mayor eficiencia en los procesos productivos, menores costos ante fallas, mejor servicio a clientes, etcétera.

Cuando un sistema de trazabilidad está basado en una infraestructura apoyada en las tecnologías de la información y las comunicaciones (TIC), la trazabilidad brinda importantes utilidades a los diferentes actores de una cadena de valor: gestión eficiente de la logística y del suministro, y aumento de la productividad.

2.3.1. *SISTEMA DE TRAZABILIDAD*

Un sistema de trazabilidad es un conjunto de disciplinas de diferente naturaleza que, coordinadas entre sí, nos permiten obtener el seguimiento de los productos a lo largo de cualquier cadena del tipo que sea. Por lo tanto un sistema de trazabilidad deberá de estar compuesto por:

Sistemas de identificación

- Un sistema de identificación del producto unitario.
- Un sistema de identificación los embalajes o cajas.
- Un sistema de identificación de bultos o pallets.

Sistemas para la captura de datos

- Para las materias primas.
- Para la captura de datos en planta.
- Para la captura de datos en almacén.

Software para la gestión de datos

- Capaz de imprimir etiquetas.
- Capaz de almacenar los datos capturados.
- Capaz de intercambiar datos con los sistemas de gestión empresariales.

2.3.2. *SOFTWARE DE TRAZABILIDAD*

El Software de trazabilidad registra la traza de los productos a lo largo de la cadena de suministro interna o externa, los empaqueta en un formato legible y los prepara para poder ser gestionados por el propio software o como respuesta a una solicitud de servicio. El software de trazabilidad generalmente utiliza ó genera identificadores para RFID y/o códigos de barras, dependiendo si necesitamos que la información vaya en un sentido o en ambos sentidos, y del factor de automatización que deseemos para la captura de datos.

Los sistemas para el intercambio de información basados en peticiones de servicios como *SOAP, XML, HTTP y WSDL* han simplificado las comunicaciones entre diferentes plataformas permitiendo tener un sistema central de gestión y almacenamiento de la información. Por ello, la capa principal y más importante del software de trazabilidad ha de ser cómo captar los datos y transmitirlos al sistema de gestión, en lugar de mantener bases de datos replicadas sin tener disponibilidad en línea de la información, Delegando la gestión de los códigos, proveedores y otro tipo de informaciones al sistema de gestión central, que se comunica con el software de trazabilidad a través del lenguaje SOA, con un fichero plano que tiene una estructura estandarizada globalmente por W3C.

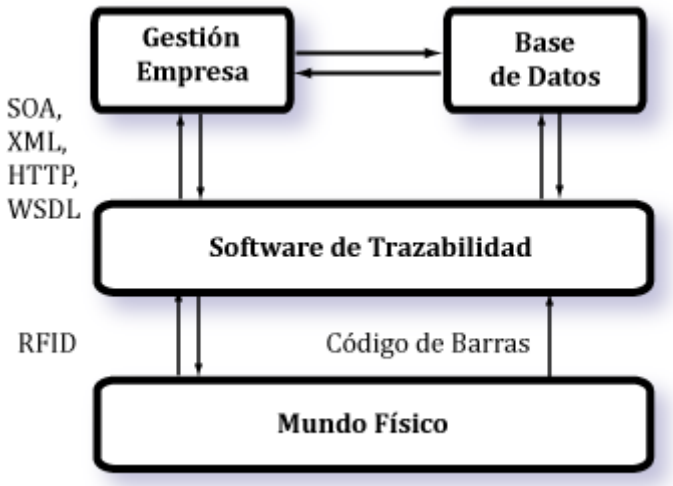


FIG. 22 DIAGRAMA SOFTWARE DE TRAZABILIDAD

3. IMPLEMENTACIÓN

3.1. HIPÓTESIS

Esta tesis establece como hipótesis que es factible diseñar e implantar un sistema de trazabilidad y distribución de mercancía, que permita reducir costos, tiempos de entrega, mejorar el intercambio comercial entre los clientes y la empresa, así como la competitividad de la misma. Con esta filosofía de *Sistemas Orientado a Servicios* resolveremos la problemática detallada en el capítulo 1.

La investigación aplicada propone nuevos procesos o un nuevo sistema, para dar respuesta a cada una de las directivas del enunciado en el problema original. Esperamos que el escenario del problema se vea afectado positivamente de la siguiente manera:

- Se reestructurará parte del modelo comercial de la empresa, uniendo a este un nuevo modelo de empaquetado y distribución de mercancía.
- El sistema reducirá los tiempos de preparado de mercancía, facturación, envío de documentación y embarque en más de un 50%
- El sistema tendrá el propósito de evitar errores involuntarios por parte del personal, generando así modelos a prueba de error.
- El sistema contendrá métodos de validación de la información para que, en caso de que la interfaz detecte en alguna consulta un conjunto de datos no válidos, automáticamente deseche la consulta e informe al distribuidor sobre la causa por la que se desechó.
- El sistema generará la información en línea que soliciten los diferentes clientes, dependiendo de los requerimientos en el momento en que esta información sea solicitada, evitando enviar información extemporánea.
- El sistema tendrá el propósito de reducir la recaptura de información, con el objetivo de evitar errores.
- El sistema será programado utilizando estándares y tecnología de punta. Los *servicios Web* están siendo adoptados rápidamente por empresas y fabricantes como el estándar en Arquitecturas Orientadas a Servicio, lo que hace que todos los fabricantes especialicen sus herramientas para ésta tarea.
- Se reducirán las pedidas de mercancía, mejorando el manejo del inventario.
- El sistema utilizará sistemas operativos, motores de bases de datos y lenguajes de programación, que sean compatibles con los lineamientos de la empresa.
- Se vigilará que la interfaz sea la mejor opción en tiempo de programación y adaptación; el tiempo deberá ser reducido en comparación con otras opciones o metodologías.
- La interfaz aportará un nuevo volumen de datos, que permitirán generar y desarrollar nuevas prácticas de negocios.

3.2. HERRAMIENTAS DE SOFTWARE Y HARDWARE

3.2.1. *SOFTWARE*

Uno de los lineamientos de la empresa es que todo el sistema se desarrollara en *Visual Basic, Microsoft SQL Server,* por lo que no tuve opción de escoger otro *SGBD*, o leguaje de programación.

Microsoft SQL Server 2008

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son *T-SQL y ANSI SQL. Microsoft SQL* Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son *Oracle, PostgreSQL o MySQL.*

Características generales:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.
- Para el desarrollo de aplicaciones más complejas (tres o más capas), *Microsoft SQL Server* incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas *.NET*, pero el servidor sólo está disponible para Sistemas Operativos Windows. [41]

Transact SQL

T-SQL (Transact-SQL) es el principal medio de programación y administración de *SQL Server.* Expone las palabras clave para las operaciones que pueden realizarse en *SQL Server*, incluyendo creación y modificación de esquemas de la base de datos, introducción y edición de datos, así como supervisión y gestión del propio servidor. Las aplicaciones cliente, ya sea que consuman datos o administren el servidor, aprovechan la funcionalidad de *SQL Server* mediante el envío de consultas de *T-SQL* y declaraciones que son procesadas por el servidor; los resultados (o errores) regresan a la aplicación cliente. *SQL Server* permite que los resultados sean administrados mediante *T-SQL* Para esto, expone tablas de sólo lectura con estadísticas del servidor. La funcionalidad para la administración se expone a través de procedimientos almacenados, definidos por el sistema, que se pueden invocar desde las consultas de *T-SQL* para realizar la operación de administración. También es posible crear servidores vinculados (Linked Servers) mediante *T-SQL.* Los servidores vinculados permiten el funcionamiento entre múltiples servidores con una consulta.

Cliente Nativo de SQL

Es la biblioteca de acceso a datos para los clientes de *Microsoft SQL Server* versión 2005 en adelante. Implementa nativamente soporte para las características de *SQL Server*, incluyendo la ejecución de la secuencia de datos tabular, soporte para bases de datos en espejo de *SQL Server,* soporte completo para todos los tipos de datos compatibles con *SQL Server*, conjuntos de operaciones asíncronas, notificaciones de consulta, soporte para cifrado, y recepción de varios conjuntos de resultados en una sola sesión de

[41] http://www.microsoft.com/spain/sql/2008/overview.aspx

base de datos. *Cliente Nativo de SQL* se utiliza como extensión de *SQL Server* plug-ins para otras tecnologías de acceso de datos, incluyendo ADO u OLE DB. *Cliente Nativo de SQL* puede también usarse directamente, pasando por alto las capas de acceso de datos genéricos.

Microsoft Visual Basic 2010 .Net

Microsoft Visual Basic es hoy en día el lenguaje más popular de programación en el mundo. La palabra Basic hace referencia al lenguaje *BASIC* (*Beginenner All-Purpose Symbolic Instruction Code*), un lenguaje utilizado por más programadores que ningún otro lenguaje utilizado en la historia de la informática. *Visual Basic* ha evolucionado a partir del lenguaje *BASIC* original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están relacionadas a la interfaz gráfica de *Windows*. (Ceballos, 2010)

Microsoft liberó *Visual Basic 1.0* en 1991. *Visual Basic 1.0* revolucionó la forma de desarrollo de software para *Windows*; desmitificó el proceso de desarrollo de aplicaciones con interfaz gráfica de usuario y abrió este tipo de programación a las masas. En sus posteriores versiones, *Visual Basic* ha continuado proporcionando nuevas características que facilitaron la creación de aplicaciones para *Windows* cada vez más potentes, la *versión 3.0* introdujo el control de datos para facilitar el acceso a bases de datos, la *versión 4.0* mejoró y potenció este acceso con los objetos **DAO**. Con la aparición de *Windows 95 Microsoft* liberó *Visual Basic 4.0*, que abrió las puertas a las aplicaciones de 32 bits y a la creación de las **DLL**. La *versión 5.0* mejoró la productividad con la incorporación de la ayuda inteligente y la introducción de los controles **ActiveX**. Finalmente la *versión 6.0* introdujo a la programación en Internet con las aplicaciones **DHTML** y el objeto ***WebClass*.**

Ahora disponemos de *Visual Basic .NET* (*Visual Basic 2005*, *Visual Basic 2008*, y *Visual Basic 2010*) que cambia la manera de programar de las versiones iniciales. Ahora se requiere una programación orientada a objetos, lo que obliga al desarrollador a programar de una forma ordenada, con unas reglas metodológicas de programación analógicas a otros leguajes de programación orientados a objetos como C++ o Java.

La palabra *NET* hace referencia al ámbito donde operarán nuestras aplicaciones (Network), *Visual Basic .NET*, ahora *Visual Basic 2010* proporciona la tecnología necesaria para saltar de aplicaciones cliente servidor tradicionales a la siguiente generación de aplicaciones escalables para la WEB, *Servicios WEB*, ADO.NET, ASP.NET y el .NET Framework. (Ceballos, 2010)

.NET conduce a la tercera generación de Internet. La primera generación consistió en trabajar con información estática que podía ser consultada a través de exploradores. La segunda generación se ha basado en que las aplicaciones pudieran interaccionar con las personas; sirva como ejemplo los famosos carritos de compras. La tercera generación de aplicaciones se caracterizará por aplicaciones que puedan interaccionar con otras aplicaciones.

Precisamente el principio de .NET es que los sitios WEB aislados de hoy en día, y los diferentes dispositivos, trabajen conectados a través de Internet para ofrecer soluciones más ricas. Esto se conseguirá gracias a la aceptación de los estándares abiertos basados en XML (Extensible Markup Language). De esta manera Internet se convierte en una fuente de servicios y no sólo de datos.

Visual Basic .NET cuenta con las siguientes ventajas y desventajas:

Ventajas:

- Posee una curva de aprendizaje muy rápida.
- Integra el diseño e implementación de formularios de *Windows.*
- Permite usar con facilidad la plataforma de los sistemas *Windows,* dado que tiene acceso prácticamente total a la API de *Windows* e incluye librerías actuales.
- Es uno de los lenguajes de uso más extendido, por lo que resulta fácil encontrar información, documentación y fuentes de apoyo para los proyectos.
- Fácilmente extensible mediante librerías DLL y componentes ActiveX de otros lenguajes.
- Posibilita añadir soporte para ejecución de scripts, *VBScript o JScript,* en las aplicaciones mediante *Microsoft Script Control.*
- Tiene acceso a la API multimedia de DirectX (versiones 7 y 8). También está disponible, de forma no oficial, un componente para trabajar con OpenGL 1.1.

Desventajas:

- Problema de versiones asociado con varias librerías runtime DLL, conocido como DLL Hell
- Dependencia de complejas y frágiles entradas de registro COM.
- Costo

3.2.2. *HARDWARE*

El hardware que se adquirió para implementar el sistema es el siguiente:

Cantidad	Hardware	Utilidad	Costo Unitario	Costo
1	Servidor Proliant Serie DL160 G6	Aloja base de datos, aplicación y conecta PC´s	$35,000.00	$35,000.00
35	Computadoras personales		$5,000.00	$175,000.00
15	Impresoras Zebra S4M	Imprimen la etiquetas para pedidos y cajas	$25,000.00	$375,000.00
20	Motorola Symbol LS1203 Láser	Lector de códigos de barras	$1,750.00	$35,000.00
		Total		$620,000.00

Servidor HP PROLIANT Serie DL160 G6

El *HP ProLiant DL160 G6* es un servidor para bastidor ultra denso, de alto rendimiento, bajo coste, diseñado para entornos informáticos de alto rendimiento, e implantaciones de servidores de servicio web y de gran cantidad de memoria.[42]

FIG. 23 HP PROLIANT SERIE DL160 G6

Impresora Zebra S4M

La impresora ofrece terminal de serie estándar, conexión paralelo y USB, así como interno opcional Ethernet 10/100 y IEEE 802.11b / g de red inalámbrica para la integración y la monitorización remota, además de que se puede pedir con diferentes lenguajes de programación. Ofrece facilidad de uso, una construcción de metal, y una variedad de opciones de conectividad en una impresora térmica que toma un completo de 8 "rollo de etiquetas de Zebra para menos cambios de los medios de comunicación. La S4M posee las características necesarias para soportar muchas aplicaciones. Con controles fáciles de usar y de diseño comprobado de carga lateral. [43]

FIG. 24 IMPRESORA ZEBRA S4M

[42] Ver Anexo: VII Características técnicas HP Proliant
[43] Ver Anexo: VIII Características técnicas impresora Zebra S4M

Motorola Symbol LS1203 Láser / Con cable / De mano

El lector de mano *Symbol LS1203* está diseñado para satisfacer las necesidades y ajustarse a los presupuestos de tiendas pequeñas, al ofrecer una lectura láser de alta calidad, diseño ergonómico, facilidad de manejo y durabilidad. Para pequeñas tiendas independientes, el lector Symbol LS1203 ofrece las características, la funcionalidad y la confiabilidad necesarias para mejorar la eficacia de funcionamiento desde el paso por caja hasta almacén.

Se minimiza la introducción manual de datos a través del teclado y se automatizan los procesos de inventario en papel.[44]

FIG. 25 MOTOROLA SYMBOL LS1203

[44] Ver Anexo IX Características técnicas lector de códigos de barras Motorola Symbol

3.3. SECUENCIA PROPUESTA PARA EL MODELO COMERCIAL Y SU INTEGRACIÓN CON EL NUEVO MODELO DE DISTRIBUCIÓN

Analizar el modelo comercial, diseñar un nuevo modelo de distribución y trazabilidad que permitiera integrar ambos a una filosofía orientada a servicios, fue un proceso arduo que duró varios meses. Durante dicho proceso se analizó el modelo comercial anterior en su totalidad y se recopiló información de todos los departamentos involucrados (Ventas, A.P.T, Envíos, Facturación y Sistemas) con la intención, establecida por la Dirección[45] , de simplificarlo y de eliminar tiempos muertos para evitar así un trabajo innecesario.

En razón de lo extenso de este trabajo solo mostraré el modelo resultante, de forma superficial, mediante un diagrama que permita entender la nueva estructura propuesta. Las repercusiones para la empresa de este nuevo modelo son considerables. Se dividió en dos el *Departamento de Envíos* creándose el Departamento de Resguardo y se implementaron nuevas metodologías de trabajo en todos los departamentos involucrados.

El siguiente diagrama (fig. 26) muestra la secuencia propuesta que recorren, tanto la mercancía como la información, por los diversos departamentos involucrados (Ventas, A.P.T, Envíos, Facturación, Sistemas y Resguardo). En la parte izquierda hay un encabezado indicando los procesos que se están realizando y que se explicaran en los siguientes párrafos. Este diagrama es similar a la Fig. 2 Diagrama secuencial de la salida y facturación de pedidos de la pág. 5; al compararlos se puede tener una idea más clara de los cambios realizados en la empresa.

[45] Ver Sección 1.1.5 Propuesta a la dirección

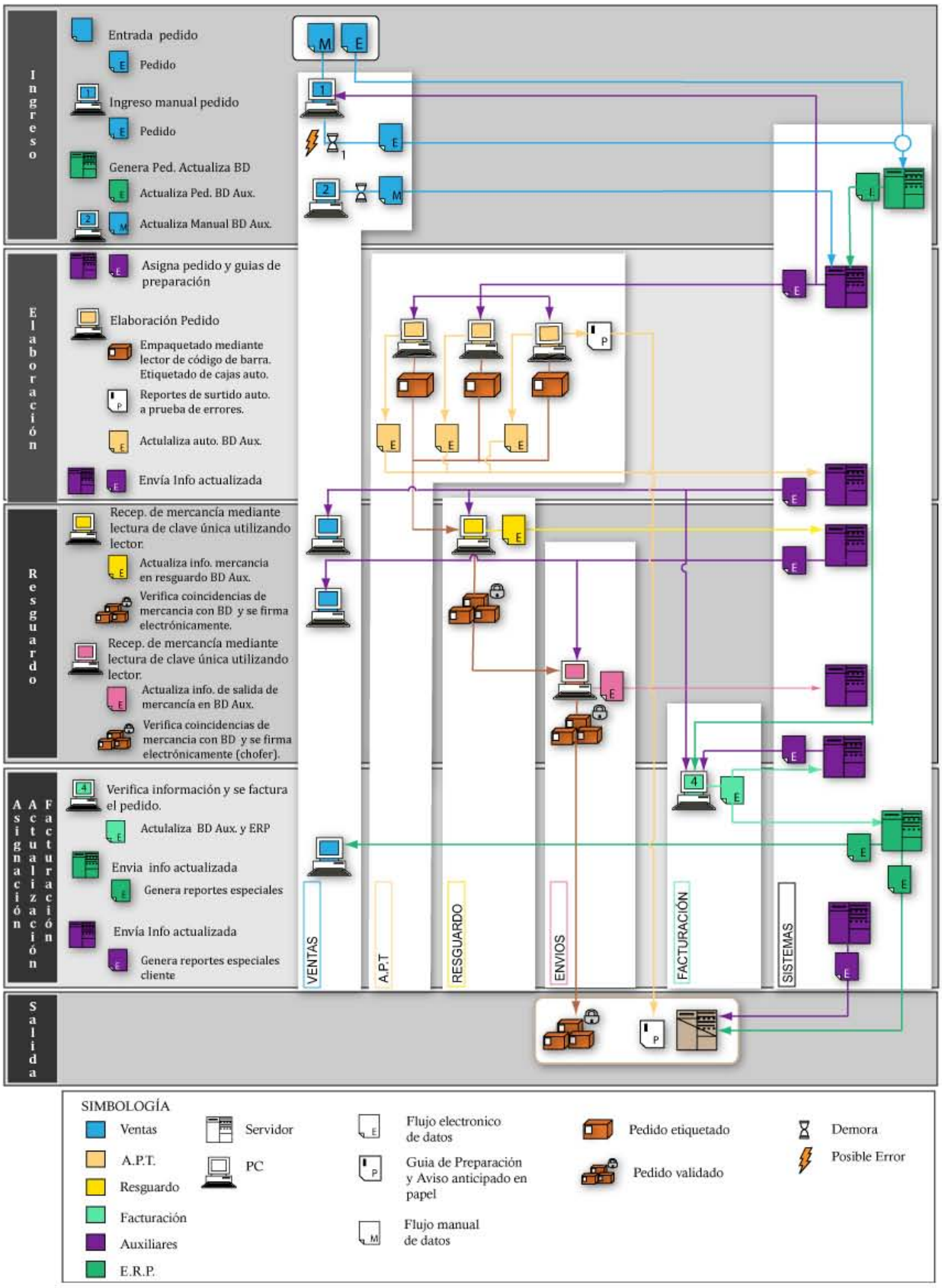


FIG. 26 DIAGRAMA SECUENCIAL DE LA SOLUCIÓN PROPUESTA

Ingreso del pedido al sistema

En el nuevo modelo los pedidos son ingresados al sistema en el Departamento de Ventas al *ERP* (Ver *Entrada Pedido)* como en el modelo previo; el ingreso puede hacerse de forma manual (Ver *Ingreso Manual Pedido)* o de forma automática mediante el *ERP*. Cuando la captura de pedidos se elabora manualmente se produce una demora y se puede generar un error de captura (Ver *Error 1*). Este tipo de errores no se pueden solucionar en el nuevo sistema pues son los valores de entrada y la forma en que se ingresan no depende de éste. Para resolver este problema se propuso que, administrativamente, el envío de pedidos se generara de manera electrónica.

Una vez ingresado el pedido en el *ERP*, este genera un archivo de extensión PRN que se guarda en una carpeta compartida[46], de la cual se obtiene la información referente al cliente, los artículos y las cantidades solicitadas en el pedido a surtir. (Ver *Genera Ped. Actualiza BD*)

La elaboración de un pedido puede requerir información adicional que necesita ser ingresada a la base de datos del sistema (*Auxiliares*). Esta información adicional hace referencia a algunas características especiales que no se encuentren almacenadas en la base del *ERP*. Dentro de ese tipo de información se encuentran códigos especiales por artículo (SKU) que pueden ser requeridos en la etiqueta[47]. (Ver *Actualiza Manual BD Aux.*)

Elaboración o surtido de un pedido

Una vez ingresado el pedido, este podrá visualizarse en el sistema de forma automática e instantánea. Esto permite que el jefe de A.P.T. genere guías de preparación inmediatamente después de que el pedido ha sido ingresado en el sistema. Estas guías dividen equitativamente la carga de trabajo entre los equipos responsables de preparar el pedido. (Ver *Asigna Pedido*)

Las guías de preparación indican por equipo las sucursales a surtir y la mercancía total. Los empleados de cada equipo distribuyen la mercancía en la mesa de trabajo correspondiente de manera global acomodándola por tipo (Damas, Caballeros, Niñas, Niños, Bebes). (*Ver Asigna Pedido*) Se selecciona el pedidoy las respectivas sucursales que se van a surtir[48]. Utilizando el lector de códigos de barras se empieza a escanear la mercancía y a acomodar en cajas[49], la aplicación verifica que la cantidad de artículos surtidos, concuerde con la cantidad pedida. Así mismo asigna la contraseña del empleado a cada caja que elaboró firmándola electrónicamente. La base de datos se va actualizando cada vez que se termina de surtir una sucursal, permitiendo tener un seguimiento de los pedidos de forma instantánea. (Ver *Elaboración Pedido*)

Una vez surtidas todas las sucursales el encargado del pedido imprimirá reportes de surtido y documentación adicional del pedido. Cada uno de éstos reportes se imprimen de forma automática (Ver *Reportes de surtido auto. a prueba de errores),* en este proceso se modifica la base de datos, actualizando los totales y modificando campos que indican al Departamento de facturación, ventas y resguardo que el pedido ha sido completamente empacado. (Ver *Envía información actualizada*)

[46] Dentro de una filosofía orientada a servicios, lo ideal sería implementar un servicio web que permitiera obtener la información directamente del ERP. La empresa que diseñó el ERP había indicado que el servicio web estaría disponible en una actualización de versión para 2010, la cual nunca llegó. Por lo cual esta parte del modelo no se puede considerar orientada a servicios, hasta que la actualización se encuentre disponible y se hagan las modificaciones pertinentes al sistema.
[47] Este tipo de información no requiere ser actualizada constantemente, por lo que no implica un trabajo excesivo.
[48] Ver Sección 3.6.2 Detalle del programa de surtido en APT
[49] Diseñé un procedimiento en el cual un empleado debe de tomar la mercancía, escanearla y guardarla. Así como la forma que debe el empleado visualizar la pantalla para evitar un posible error. Ver Sección 3.6.1 Metodología para el preparado de un pedido

Resguardo del pedido

El Departamento de Resguardo escanea las cajas mediante un código único por caja, firmando electrónicamente cada una de ellas con una clave en código de barras que es responsabilidad del encargado de resguardo (Ver *Recepción de mercancía mediante lectura de clave única utilizando lector*).

Una vez que todas las cajas han sido escaneadas el pedido cambia de estatus y el sistema indica que este se encuentra en Resguardo, listo para ser enviado. Lo ideal es que inmediatamente después de que el pedido llega a Resguardo se le dirija al Área de Envíos, sin embargo, dependiendo de la logística de los departamentos tanto de Ventas como Envíos, un pedido puede tardar varios días en ser remitidos.

Una vez que las cajas se encuentran en Resguardo ya no pueden regresar a A.P.T.

El proceso de cancelación tanto de pedidos como de cajas es responsabilidad del programa *ERP*.

Asignación, facturación y actualización de información del pedido

Una vez que todo el pedido se encuentra en Resguardo, los empleados del Departamento de Facturación puede asignar[50] el pedido de forma automática a la base de datos y espera a que se autorice su facturación (Ver *Verifica información y se factura el pedido*).

Algunos clientes establecen que las entregas de pedidos se hagan mediante citas, una vez que toda la mercancía se encuentra empaquetada. Otros clientes envían la fecha de entrega al mismo tiempo que el pedido. Si el tipo de pedido se encuentra dentro del primer caso el Departamento de Ventas puede empezar a asignar citas de entrega para la recepción de mercancía.

En esta etapa se generan y envían reportes especiales de distintas tipos y formas dependiendo del cliente. Usualmente *avisos anticipados de embarque ASN.* (Ver *Genera reportes especiales cliente*).

Salida de la mercancía

En el siguiente paso el Jefe de Envíos solicita el pedido a Reguardo, de acuerdo al plan de logística previamente establecido, y nuevamente se escanean las cajas que serán enviadas. Se genera una orden de salida en la presencia del responsable de distribuir el pedido, normalmente el chofer de la unidad, que tiene que verificar y firmar la orden de salida haciéndose responsable de las cajas que debe surtir.

El sistema verifica que las cantidades enviadas concuerden con la factura, de no ser así, no se autoriza la salida del pedido.

Adicionalmente la aplicación puede enviar información adicional, de forma automática y a través de la red, dependiendo del cliente.

[50] Proceso anterior a la facturación de un pedido, donde se ingresan todos los valores surtidos.

3.4. FASES DEL DESARROLLO

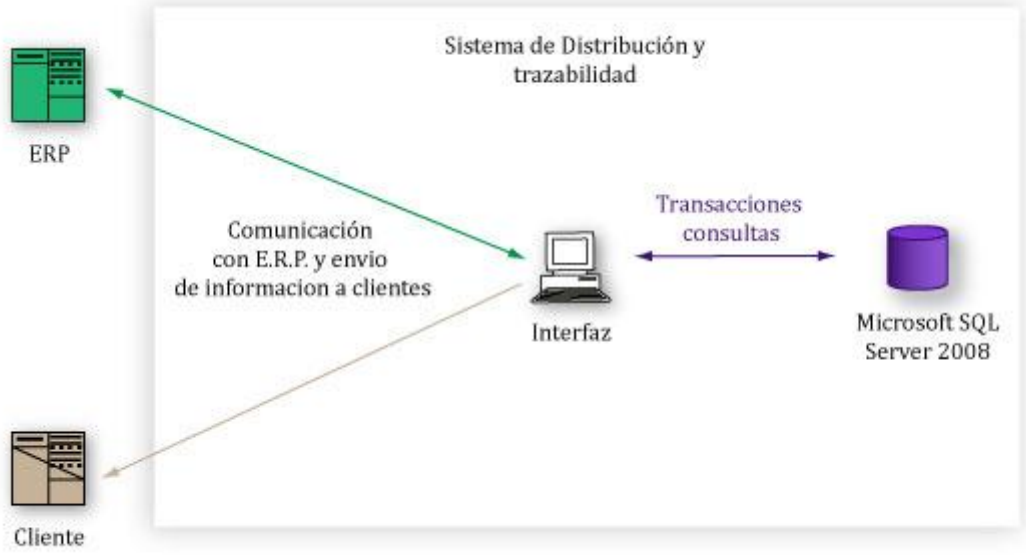


FIG. 27 COMPONENTES DE LA SOLUCIÓN PROPUESTA

Para analizar ésta solución, las próximas secciones darán detalles técnicos sobre:

- Base de datos. Ver Sección *3.5 Diseño, creación, optimización de la base de datos y evaluación de consultas.*
- Diseño de las interfaces. Ver Sección 3.6
- Comunicación con ERP y envío de información a clientes.

3.5. DISEÑO, CREACIÓN, OPTIMIZACIÓN DE LA BASE DE DATOS Y EVALUACIÓN DE CONSULTAS.

3.5.1. *ANTECEDENTES*

Para ajustar el sistema a las necesidades del cliente se ha procedido a una optimización constante que ha implicado el replanteamiento de los lineamientos iniciales y la re elaboración de los modelos en un par de ocasiones. Esto ha sido así pues el cliente desconoce los alcances del proyecto y sus necesidades futuras. También influye que no se pueden hacer análisis fidedignos de carga de trabajo en la base de datos ya que no se cuenta con datos reales.

Como vimos en el capítulo 1, al iniciarse el proyecto se desarrolló un prototipo, siguiendo la metodología por etapas, al que después se le hicieron adecuaciones. Entre la concepción y la liberación de la primera parte del modelo pasaron, aproximadamente, siete meses. Se desarrolló una base de datos con los lineamientos iniciales y se generó una interfaz en *Visual Basic 6.* La estructura de la base de datos se fue construyendo a partir de los resultados obtenidos de esa base de datos inicial. El esquema que se muestra a continuación (fig. 28) presenta el modelo *Entidad Relación* que se utilizó en el prototipo y el programa prototipo donde se hicieron las primeras pruebas.

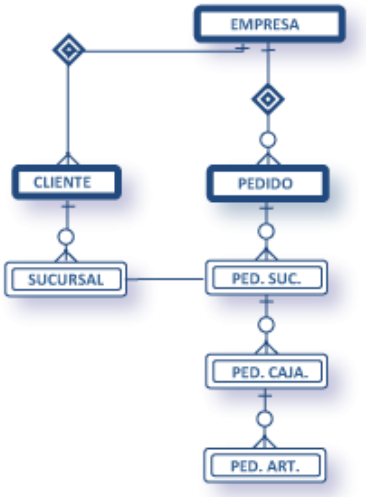


FIG. 28 MODELO ENTIDAD RELACIÓN PROTOTIPO

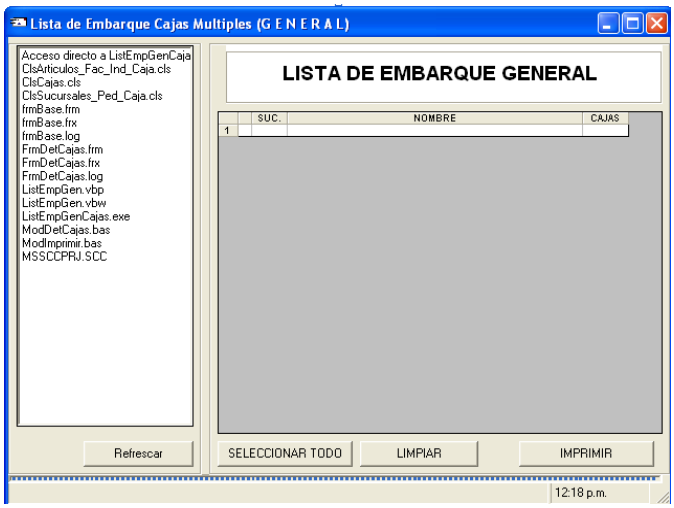


FIG. 29 INTERFAZ PROTOTIPO

No haré un análisis detallado de los cambios que se realizaron hasta llegar a la primera versión funcional del sistema que nos ocupa pues alargaría inútilmente este trabajo. Solo mostraré los lineamientos, modelos y bases de datos de dicha versión funcional y haré mención de algunas modificaciones que forman parte de la justificación de ciertas decisiones relevantes para el diseño de la base de datos. Entre éstos se encuentran, por ejemplo, la desnormalización, elaboración de disparadores o procedimientos almacenados.

Sin embargo, considero que es importante mencionar que un modelo inicial debe contemplar la mayoría de los requerimientos de la aplicación y ser lo suficientemente flexible para adaptar posibles cambios a la base.

En la siguiente sección: 3.5.2 *Diseño de la base de datos*, mostraré los requerimientos para la creación de la base de datos y como se fue conformando. Se encuentra dividida en cuatro temas los cuales son:

- Lineamientos del cliente,
- Estructura y fases de elaboración de un pedido
- Modelo Entidad Relación,
- Modelo Relacional de la base de datos.

Posteriormente en la sección 3.5.3 Creación de la base de datos. Se tocarán los siguientes temas:

- Criterios de normalización, redundancia y rendimiento. Donde se analizan los criterios para la creación de la base de datos
- Tablas ó Relaciones. Se dará una explicación detallada de las principales tablas de la base.

Finalmente en las secciones 3.5.5 Consultas a la base de datos y Administración y optimización del rendimiento de la base de datos, se mostrará el funcionamiento de algunas consultas así como ciertas evaluaciones del desempeño de la base de datos que contribuyeron a su forma final.

3.5.2. *DISEÑO DE LA BASE DE DATOS*

Lineamientos del cliente

La base de datos propuesta se formuló a partir de los siguientes lineamientos finales del cliente.

1. La base de datos debe contener una entidad empresa que esté compuesta por varios departamentos.
2. Cada departamento está compuesto por empleados. Dentro de cada área o departamento laboran diferentes tipos de empleados, con un responsable. Sin embargo, al ser un sistema de trazabilidad solamente se implementarán los departamento y empleados relacionados a al nuevo sistema.[51]
3. La empresa tiene cero o más clientes.
4. Cada cliente tiene una o más tiendas o sucursales.
5. Un cliente puede o no tener uno o más centros de distribución.
6. La empresa tiene diferentes tipos de artículos que se clasifican por departamentos (Bebes, Niños, Niñas, Damas o Caballeros) Se puede generar una confusión entre los departamentos de la empresa y los departamentos de los productos.
7. La empresa puede fabricar uno o más artículos, cada artículo tiene un código único.[52]
8. La empresa provee a los clientes mediante pedidos.
9. Cada pedido puede estar consolidado por uno o más pedidos hijos.
10. Un cliente puede o no tener un código especial (*SKU*) para cada artículo[53].
11. Un cliente puede o no tener un código especial para cada departamento de un artículo.
12. Cada pedido global está compuesto por una o más tiendas o sucursales a distribuir.
13. A cada sucursal se le surte uno o más artículos que se encuentran empaquetados en una o más cajas.
14. Dependiendo de la sucursal las cajas son enviadas, ya sea a un centro de distribución, a la sucursal, a la tienda o pueden ser recogidas dentro de la empresa.
15. Los pedidos pueden elaborarse sin interrupciones o por etapas, surtiendo solamente un porcentaje del pedido total, dependiendo del cliente o los requerimientos de la empresa.
16. Los pedidos sólo pueden ser preparados por empleados de almacén, deben quedar registrados en la base de datos, indicando qué empleado preparó qué caja, así como la hora y el responsable del pedido.
17. Las cajas una vez preparadas pasan por una aduana a Resguardo, donde deben ser registradas por un empleado encargado.

[51] El control de los departamentos y usuarios a detalle de la aplicación, se manejan mediante el ERP.
[52] El ERP no tiene una herramienta que permita consultar los artículos de la base de datos interna, así que también fue necesario crear dicha entidad.
[53] Inicialmente bajo el concepto de sistema orientado a servicios se buscó evitar duplicidad de datos utilizando los pedidos del ERP a través de un *servicio Web*, sin embargo, varios clientes comenzaron a consolidar pedidos lo que significa que un pedido padre puede tener varias órdenes o pedidos hijos, el ERP no permite este tipo de pedidos así que se tuvo que generar las entidades correspondientes.

18. Un pedido puede ser preparado por varios empleados del almacén pero solamente un empleado puede preparar una sucursal.

19. Una vez preparado el pedido, no se deben modificar las cantidades.

20. Un pedido desde que es generado hasta que es entregado al cliente pasa por varios estados: sin surtir, en preparación, en resguardo, facturado, en envíos y enviado. Para que un pedido pase completamente de un estado a otro, es necesario que todas las cajas de dicho pedido se encuentren en ese estado.

21. El cambio de estatus en el pedido, sólo lo puede hacer el empleado responsable; al cambiar el estatus de su departamento específico debe de quedar registrada la hora y el empleado que hizo dicho cambio de estatus tanto en la caja, como en la sucursal y en el pedido global. Una vez que la caja o el pedido cambian de estatus no pueden regresar a un estado anterior.

Estructura y fases de elaboración de un pedido

La parte medular del sistema propuesto es el Pedido, por lo tanto para poder diseñar la base de datos es necesario entender cómo se encuentra compuesto, cuales son los procesos y los lineamientos que debe cumplir. Temas que se revisaran en la siguiente sección.

Un pedido tiene tres principales fases de elaboración,

- **Ingreso:** Es la fase inicial, es donde se insertan todos los valores necesarios para procesar un pedido. Se crean:

    1. *el Encabezado* asignándole a éste datos generales (orden del cliente, número del pedido, fecha de vencimiento, etcétera),

    2. *sucursales* que están solicitando la mercancía,

    3. artículos solicitados por sucursal, y

    4. *códigos especiales* que algunos clientes particulares requieren.[54]

- **Surtido:** Al encabezado se le adiciona información general, como el empleado responsable del preparado del pedido, total de artículos surtidos, fecha de inicio de la elaboración. Total de cajas surtidas. Para integrar los valores surtidos a la base de datos se va a requerir que dentro de la estructura *Sucursales* se agreguen tres nuevos elementos.

    1. las *cajas por sucursal* que fueron elaboradas,

    2. los artículos surtidos por caja.[55]

    3. *control de los empleados que están elaborando el pedido.* Como varios empleados pueden trabajar en un pedido al mismo tiempo es necesario controlarlos y controlar el trabajo que están efectuando.

    En esta fase la estructura del pedido adquiere su forma final, que va a conservar en la base de datos.

[54] En caso de no estar ingresados en la base de datos de antemano
[55] Los artículos solicitados difiere del detalle de los artículos surtidos ya que los solicitados dependen de la sucursal y en el caso de los artículos surtidos dependen de la caja donde se encuentran que a su vez está vinculada a la sucursal.

- **Trazabilidad:** Esta fase es de seguimiento o trazabilidad de la mercancía, en ella es necesario conservar la estructura del pedido, por lo tanto es preciso impedir ciertas modificaciones a la base de datos como:

    - la información general del pedido o,
    - la eliminación de una caja o artículo.

  Dependiendo del departamento en el que se encuentra un pedido, se asigna un identificador o bandera dependiendo de la sección o departamento en el que se encuentra. Un pedido no puede pasar de sección si no han pasado todas las cajas de una sección inicial a la que le sigue.

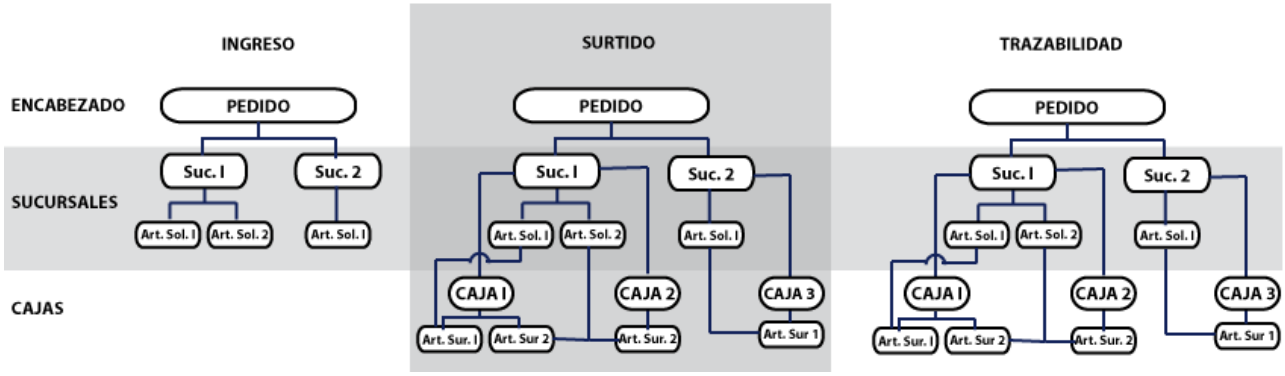


FIG. 30 ESTRUCTURA Y FASES DE ELABORACIÓN

Casos especiales de pedidos

La mayoría de los pedidos siguen la estructura y fase de elaboración que se mostró en la sección anterior, sin embargo, existen unas clases de pedidos que además de las características básicas que expliqué anteriormente tienen una serie de características especiales que citaré a continuación.

Pedidos Consolidados

Las políticas comerciales de algunos clientes demandan que varios pedidos de diferentes departamentos[56] puedan ser consolidados en un mismo pedido "*padre*". Cuando se presentan este tipo de casos una misma caja puede tener artículos solicitados en diferentes pedidos.

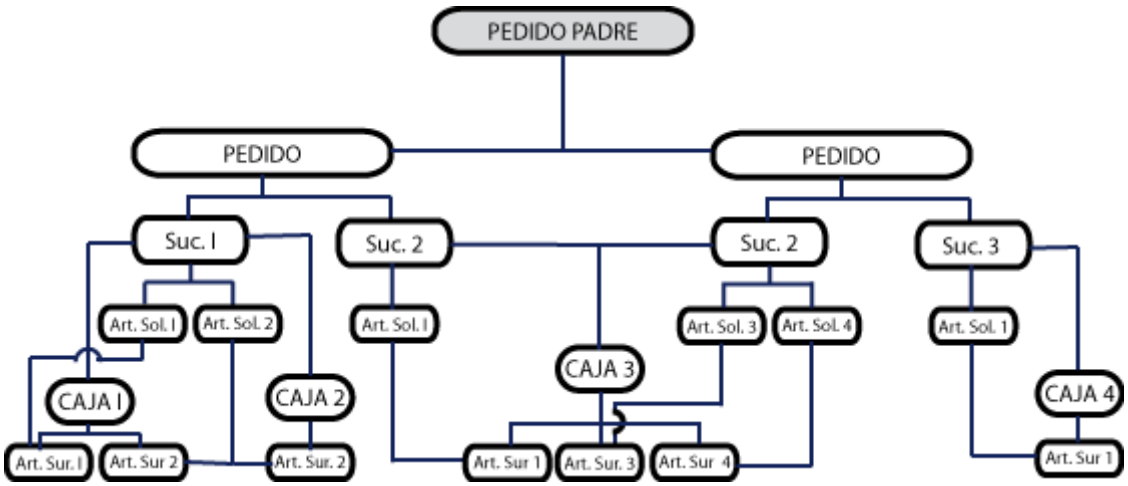


FIG. 31 PEDIDOS CONSOLIDADOS

[56] Departamentos de artículos (Niñas, Niños, Bebes, Dama y Caballero)

Pedidos de múltiples entrega de mercancía segmentadas por fecha

Un mismo pedido puede ser surtido en varias entregas segmentadas, para lo cual se necesita agregar un nuevo criterio de clasificación denominado etapa. En cada etapa se requiere información general, como las fechas de entrega, así como la cantidad de piezas surtidas y cajas, por etapa, también se requiere un identificador que indique qué cajas y artículos fueron enviados en una respectiva etapa.

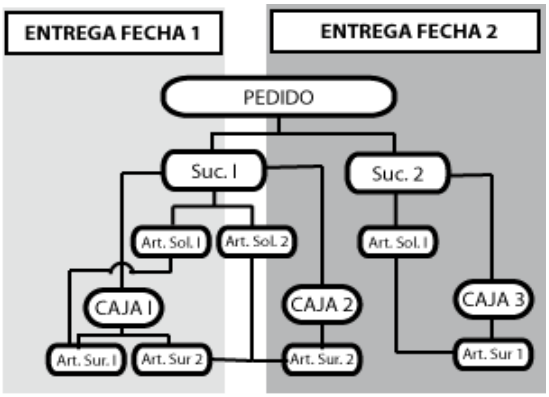


FIG. 32 PEDIDO POR ETAPAS

Modelo Entidad Relación

Mediante los lineamientos generales del cliente y comprendiendo, la estructura y funcionamiento requerido de un pedido, generé el siguiente modelo entidad relación.

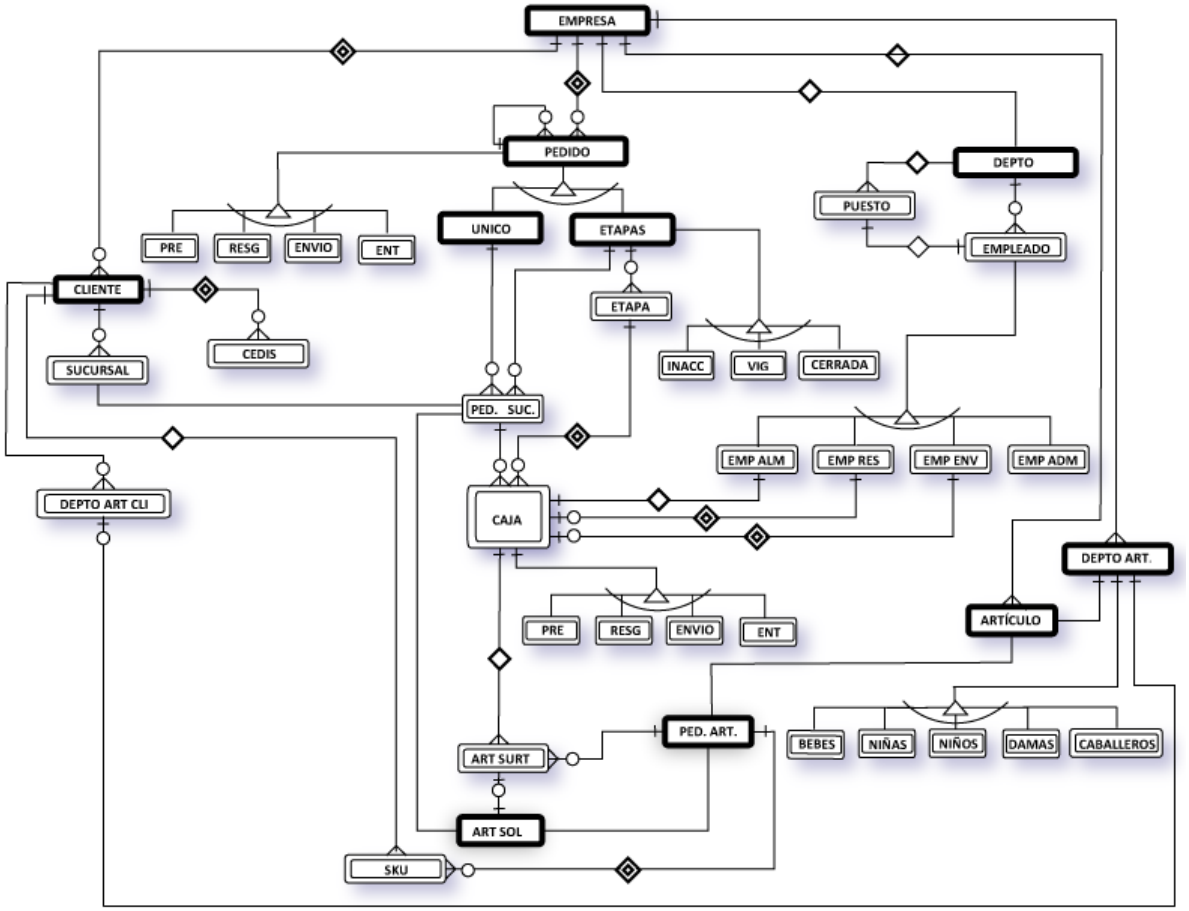


FIG. 33 MODELO ENTIDAD RELACIÓN PROPUESTO

Modelo Relacional de la base de datos

Elaborado el *Modelo Entidad Relación*, pasamos a la elaboración del *Modelo Relacional*. El modelo está compuesto por 23 relaciones.

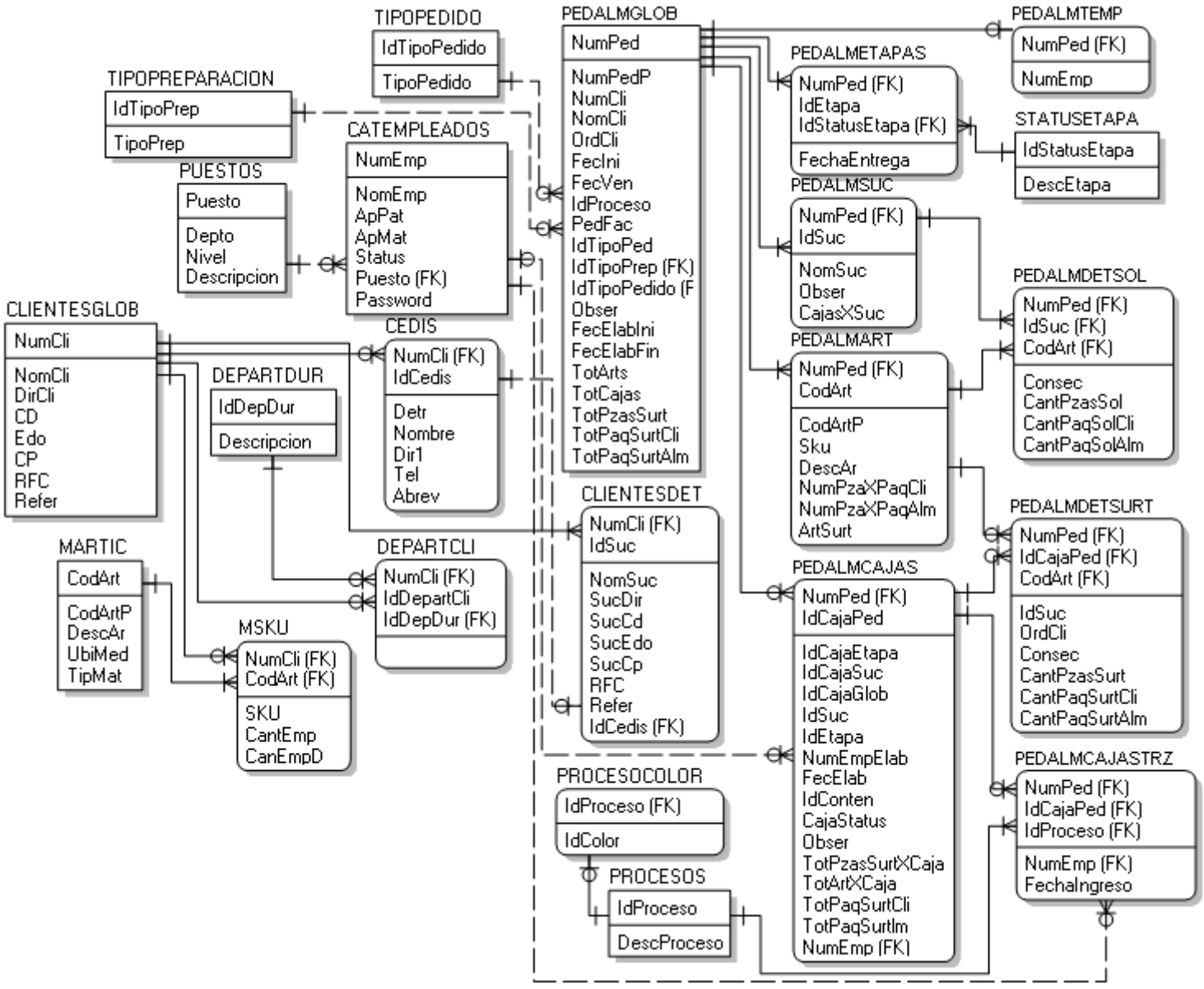


FIG. 34 MODELO RELACIONAL PROPUESTO

Dichas relaciones se pueden clasificar en tres grupos.

1. Empresa y sus requerimientos para la elaboración de pedidos;
2. Cliente y sus requerimientos para la elaboración de pedidos, y
3. Pedido y su trazabilidad.

La cantidad de información que contiene cada grupo así como la regularidad con la que se hacen cambios va aumentando del grupo *Empresa*, a *Clientes*, al grupo de *Pedidos*. Los dos primeros son relativamente

pequeños y sirven de soporte para el tercer grupo donde se encuentra el 99% de la información de la base de datos.[57]

Empresa y sus requerimientos

Consta de ocho relaciones, en este grupo se encuentra información referente a los artículos que elabora la empresa (MARTIC), los grupos de artículos que fabrica (DEPARTDUR), los empleados que laboran en ella (CATEMPLEADOS), sus puestos (PUÉSTOS), los diferentes tipos de preparación de pedidos (TIPOPREPARACIÓN), los tipos de pedidos (TIPOPEDIDO), y los procesos por los cuales tiene que pasar un pedido (PROCESOS, PROCESOCOLOR).

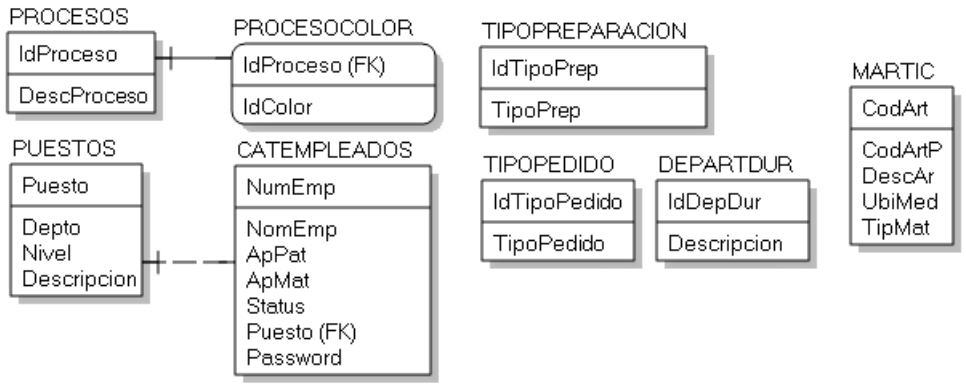


FIG. 35 EMPRESA Y SUS REQUERIMIENTOS

Clientes y sus requerimientos

Se encuentra formado por cinco relaciones, Información general del cliente (CLIENTESGLOB); Detalle de sus sucursales, tienda o domicilio particular (CLIENTESDET); Centros de distribución (CEDIS) y Códigos especiales por Artículo (MSKU).

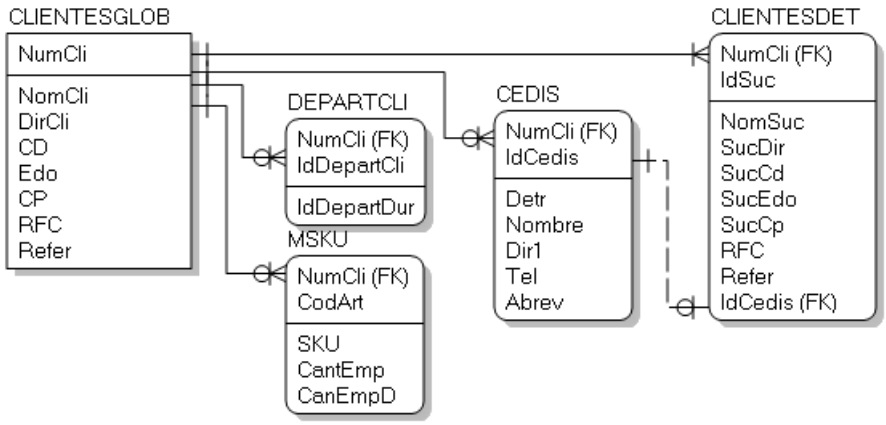


FIG. 36 RELACIONES CLIENTES Y SUS REQUERIMIENTOS

[57] Ver Anexos IV Cantidad de renglones por tabla (Dic. 2011)

Pedidos y trazabilidad

Por último, el grupo de Pedidos y trazabilidad este grupo está formado por diez relaciones, como ya mencioné anteriormente es el grupo que almacena la mayor cantidad de información de la base de datos. Contiene información general del pedido (PEDALMGLOB); Detalle del pedido por sucursal (PEDALMSUC); caja (PEDALMCAJAS); información general sobre los artículos solicitados (PEDALMART); información detallada por sucursal de los artículos solicitados (PEDALMDETSOL); detalle de los artículos surtidos (PEDALMDETSURT), Bloque o etapas en las que se ha enviado la mercancía (PEDALMETAPAS, STATUSETAPA), (PEDALMCAJASTRZ) tabla que indica la trayectoria de la caja a través de los distintos departamentos o secciones por donde pasa, y finalmente (PEDALMTEMP), tabla temporal que indica los empleados que están laborando en un pedido particular.

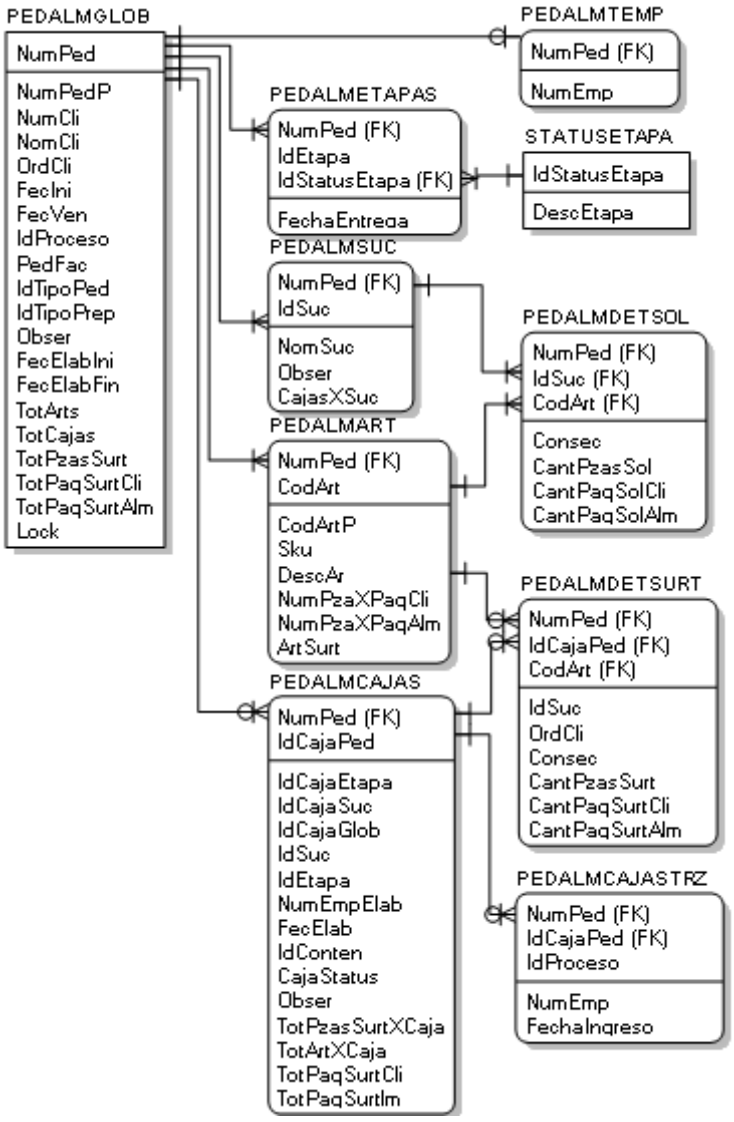


FIG. 37 RELACIONES PEDIDOS Y TRAZABILIDAD

3.5.3. *CREACIÓN DE LA BASE DE DATOS*

Antecedentes

En las siguientes secciones, voy a centrarme en el grupo 3 de relaciones *Pedidos y Trazabilidad*, el cual contiene la mayor cantidad de información y la mayor carga de trabajo ya que implica la elaboración de las tablas Disparadores y *Constraints*. [58]

Antes de elaborar el diseño final de la base de datos es importante identificar posibles problemas de rendimiento, tanto en su diseño como en su operación, detectando causas que originen:

- Bloqueos.
- Contención de recursos del sistema.
- Un conjunto determinado de consultas o procedimientos almacenados con largos tiempos de ejecución.

La estructura final de las tablas que se muestra en *Tablas ó Relaciones,* pág. 89, es el resultado de la implementación de modelos vistos anteriormente, y del análisis del comportamiento de los datos en la base. En

Criterios de normalización, redundancia y rendimiento, pág. 93, se pretende explicar los criterios que se utilizaron para la creación y modificación tanto de las relaciones, procedimientos almacenados y disparadores, antes de la creación de la base de datos, haciendo un análisis del tipo de manejo de la información que se iba a utilizar. Una vez que se contaba con una metodología de trabajo en el almacén y suficientes datos almacenados para poder identificar fallas potenciales se hicieron análisis más detallados utilizando algunas herramientas de *SQL* que permiten analizar el flujo de la información, y se verá más a detalle en la sección 3.5.5 *Administración y optimización del rendimiento de la base de datos.*

Sin embargo, para poder entender el comportamiento de la base de datos final, hay que partir de una estructura simple que surge del modelo relacional anterior, formado por las siguientes tablas o relaciones.

TABLA	DESCRIPCIÓN
PEDALMGLOB	Tabla que contiene los datos globales del pedido
PEDALMETAPAS	Tabla que contiene los datos por etapa
PEDALMSUC	Tabla que contiene los datos por sucursal
PEDALMCAJAS	Tabla que contiene los datos por caja
PEDALMARTÍCULOS	Tabla que contiene los datos generales por articulo solicitado
PEDALMDETSOL	Detalle de los artículos solicitados por sucursal
PEDALMDETSURT	Detalle de los artículos surtidos por caja

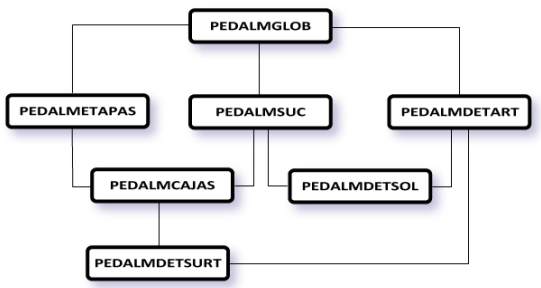


FIG. 38 ESTRUCTURA BÁSICA DE LAS TABLAS O RELACIONES

[58] En el Anexo X Creación de la base de datos

Criterios de normalización, redundancia y rendimiento

Conocer el comportamiento que va a tener la base de datos, y posteriormente hacer análisis de rendimiento para evaluar su desempeño es muy importante. A continuación se muestran dos diferentes tipos de base de datos y las consideraciones especiales que se deben de tener en cada una de ellas.

Base de datos de proceso de transacciones en línea ó OLTP (Online Transaction Processing)

Este tipo de bases de datos se utilizan para administrar datos que cambian con frecuencia. Las características de este tipo de bases de datos es que cuentan normalmente con muchos usuarios que realizan transacciones al mismo tiempo que cambian datos en tiempo real. Aunque las solicitudes de datos realizadas individualmente por los usuarios tienden a hacer referencia a pocos registros, muchas de estas peticiones se producen al mismo tiempo.

Las principales preocupaciones en este tipo de base de datos son la *simultaneidad* y la *atomicidad*.

Los controles de *simultaneidad* de un sistema de base de datos aseguran que dos usuarios no puedan cambiar los mismos datos o que un usuario no pueda cambiar un dato mientras otro usuario lo esté utilizando. Por ejemplo, si un empleado empieza a preparar un pedido para una sucursal, no es posible que otro empleado comience a preparar la misma sucursal.

La *atomicidad* garantiza que todos los pasos de una transacción se realicen correctamente como un grupo. Si se produce un error en algún paso, no se realizarán los pasos restantes. Por ejemplo, es importante que las cantidades de artículos que tiene una sucursal concuerden con el total de los valores de las cajas surtidas para dicha sucursal, o que, si se inserta un artículo inexistente no se modifiquen los totales del pedido.

Las consideraciones que se deben de tomar en cuenta para una base de datos *OLTP* [59]

- *Transacciones cortas* para reducir los bloqueos de larga duración y mejorar la simultaneidad.
- *Alto grado de normalización de la base de datos,* Reducir la información redundante con el fin de acelerar las actualizaciones y por tanto, mejorar la simultaneidad. La reducción de los datos también agiliza la realización de copias de seguridad, ya que es menor la cantidad de datos que es necesario copiar.
- Pocos (o ningún) *datos históricos* o agregados
- *Dosificar los índices.* Deben actualizarse cada vez que se agrega o se modifica una fila. Para evitar una indización excesiva de tablas con un gran número de actualizaciones se limita el alcance de los índices.

Bases de datos de ayuda a la toma de decisiones (DSS, Decision Support System).

Este tipo de bases de datos son óptimas para las consultas de datos que no impliquen cambios en los mismos. Normalmente las bases de datos de este tipo almacenan información de otras bases de datos, y no requieren modificaciones. Las tablas se indizan a menudo, y los datos sin formato suelen ser previamente procesados y organizados para los distintos tipos de consultas que se realizan. Dado que los usuarios no cambian los datos, la simultaneidad y la atomicidad no suponen motivos de preocupación; los datos sólo se cambian mediante actualizaciones masivas que se realizan periódicamente en horas de escasa actividad de la base de datos y fuera del horario laboral.

[59] Proceso de transacciones en línea frente a la ayuda a la toma de decisiones http://msdn.microsoft.com/es-es/library/ms187669(v=SQL.100).aspx

Las bases de datos de sistemas de ayuda a la toma de decisiones (*DSS, Decision Support System*) deben diseñarse para promover lo siguiente:

- Una *intensa indización*. Los sistemas de ayuda a la toma de decisiones exigen pocos requisitos para la actualización, pero suelen contener un gran volumen de datos. Utilice un gran número de índices para mejorar el rendimiento de las consultas.
- La *eliminación de la normalización* de la base de datos.
- *La introducción de datos resumidos* o previamente agregados para satisfacer los requisitos de consulta más comunes y mejorar los tiempos de respuesta de las consultas.
- La utilización de un *esquema de estrella o un esquema radial ramificado* para organizar los datos dentro de la base de datos.

*La base de datos del sistema (denominada Auxiliares) se comporta de las dos formas anteriores***.**

Tiene un comportamiento de tipo *OLTP* cuando se está preparando un pedido, ya que dicho pedido puede estar siendo elaborado por varias personas al mismo tiempo. Lo que implica varias inserciones casi simultáneas a la base de datos sobre las mismas tablas.

Por otro lado es necesario tener los subtotales de piezas surtidas por artículo, caja, sucursal y pedido, para las guías de empaque, de embarque y de facturación del pedido. Donde se requiere un comportamiento de bases de datos para la toma de decisiones, una base de datos tipo *DSS*.

Esto plantea una serie de problemas a tomar en cuenta. Si utilizamos un criterio para la elaboración de la base de datos de tipo *OLTP*, con una base de datos altamente normalizada, la solución sería hacer una inserción en la base de datos por artículos por caja. Los totales y subtotales se harían mediante sumatorias anidadas.

A continuación se muestran las tablas que se requieren para la elaboración de un pedido con un criterio tipo *OLTP.*

PEDALMGLOB	
NUMPEDP	NUMERO DEL PEDIDO PADRE
NUMPED	***NUMERO DEL PEDIDO***
NumCli	NUMERO DEL CLIENTE
NomCli	NOMBRE DEL CLIENTE
OrdCli	ORDEN DEL CLIENTE
FecIni	FECHA DE CARGA
FecVen	FECHA DE VENCIMIENTO DEL PEDIDO
idProceso	INDICADOR DE QUE PARTE DEL PROCESO DE ENTREGA SE ENCUENTRA
PedFac	INDICADOR SI EL PEDIDO SE ENCUENTRA FACTURADO
IdTipoPed	IDENTIFICADOR DEL TIPO DE PEDIDO, ESTE PUEDE SER CONSOLIDADO O INDIVIDUAL
IdTipoPrep	IDENTIFICADOR DEL TIPO DE PREPARADO PUEDE SER EN PARTES O EN UN SOLO MOV.
Obser	OBSERVACIONES

PEDALMSUC	
NumPed	***NUMERO DEL PEDIDO***
IdSuc	***IDENTIFICADOR DE LA SUCURSAL***
NomSuc	NOMBRE DE LA SUCURSAL
Obser	OBSERVACIONES

PEDALMCAJA	
IdCajaGlob	***IDENTIFICADOR ÚNICO DE LA CAJA***
IdCajaPed	IDENTIFICADOR DE LA CAJA POR PEDIDO
IdCajaEtapa	IDENTIFICADOR DE LA CAJA POR ETAPA
IdCajaSuc	IDENTIFICADOR DE LA CAJA POR SUCURSAL
NumPed	NUMERO DE PEDIDO
IdSuc	IDENTIFICADOR DE LA SUCURSAL
IdEtapa	IDENTIFICADOR DE LA ETAPA
NumEmpElab	NUMERO DEL EMPLEADO QUE ELABORÓ LA CAJA
FecElab	FECHA Y HORA DE ELABORACIÓN
idConten	IDENTIFICADOR DEL CONTENEDOR
CajaStatus	INDICADOR DE QUE PARTE DEL PROCESO DE ENTREGA SE ENCUENTRA LA CAJA
Obser	OBSERVACIONES

PEDALMDET	
NumPed	***NUMERO DE PEDIDO***
IdSuc	**IDENTIFICADOR DE SUCURSAL**
IdCajaPed	*IDENTIFICADOR CAJA*
OrdCli	ORDEN CLIENTE
Consec	CONSECUTIVO
CodArt	*CODIGO UPC*
CantPzasSurt	**CANTIDAD DE PIEZAS SURTIDAS**
CantPaqSurtCli	**CANTIDAD DE PAQUETES SURTIDOS EMPAQUE CLIENTE**
CantPaqSurtAlm	**CANTIDAD DE PAQUETES SURTIDOS EMPAQUE ALMACEN**

Esta manera de almacenar datos en la base es muy eficiente ya que son transacciones cortas que reducen los bloqueos de larga duración, mejoran la simultaneidad y reducen la información redundante. Sin embargo, en la consulta de dichos pedidos se generan transacciones con largos tiempos de ejecución pues, para hacer reportes completos de los artículos surtidos o para la elaboración de una factura, es necesario saber cuántas piezas por artículo se surtieron por sucursal y por pedido, y hacer las sumatorias genera un retardos considerables por máquina, ocasionando bloqueos cada vez que se quisiera imprimir o consultar los totales.[60]

Otra solución es crear subtotales en cada tabla -de cajas, sucursales y pedidos- y que cada vez que se insertara información, se hicieran modificaciones directamente en cada una de ellas. Esta parte reduce mucho el tiempo para la consulta de los pedidos, pero trae consigo una serie de problemas a la hora de insertar totales acumulados. Puede generar inconsistencias al hacer la suma de los totales de artículos, ó, para hacer consistentes los valores insertados en los totales imposibilitar que se hagan inserciones a la base de datos hasta haber concluido el conteo de subtotales, mediante un procedimiento almacenado que

[60] Las cantidades de piezas y paquetes(CantPzasSurt, CantPaqSurtCli y CantPaqSurtAlm) únicamente se encuentran en la tabla PEDALMDET.

haga la suma de los subtotales por artículo, caja, sucursal y pedido. Cuando se hacen una gran cantidad de inserciones de este tipo a la base de datos, se crean bloqueos ya que detener consultas posteriores puede ocasionar que el *tiempo límite de respuesta* de dichas consultas se exceda.

Para agilizar la inserción de datos y para facilitar la impresión de documentación, sin que hubiera bloqueos o tiempos muy largos de ejecución, sin comprometer la integridad de los valores de los totales, propuse lo siguiente.

Generar campos de totales de artículos en la tabla PEDALMGLOB *TotArts, TotCajas, TotPzasSurt, TotPaqSurtCli, TotPaqSurtAlm*. Estos campos **no** pueden ser modificados mediante inserción directa a la base de datos. Al ser generado un registro para un pedido, los valores totales adquieren valores de **0**. Los valores totales solamente se pueden actualizar de forma automática en una sola ocasión al finalizar el pedido mediante un disparador o *trigger*, al modificar el campo *LOCK*. Cuando se modifica este campo el trigger hace una suma de los totales por caja y por artículos, de las tablas PEDALMCAJAS y PEDALMART.

Por otro lado solamente se puede insertar artículos surtidos por caja, pero **no** se pueden modificar, o eliminar individualmente. Al ser insertados los artículos junto con la caja donde se encuentran se activa un disparador[61] que actualiza los totales de artículos surtidos en la tabla PEDALMART y los totales por caja en la tabla PEDALMCAJAS, impidiendo que los totales puedan ser modificados de otra manera. La única forma de borrar un artículo de la base de datos es eliminando la caja donde se encuentra. Esto genera un bloque sólido sin que se comprometa la integridad de la base de datos pero permitiendo consultar totales de forma más efectiva. Las sumatorias de piezas y artículos se realizan solamente en una ocasión al finalizar de surtir el pedido.[62] Además la sumatoria de los artículos se realiza a partir de los totales por caja en vez de sumando cada uno de las piezas de artículos surtidos reduciendo considerablemente el tiempo de consulta. [63]

[61] Ver Anexo XI Consideraciones acerca de operaciones con varias filas para triggers DML

[62] Antes de finalizar el pedido no es necesario verificar los totales de artículos surtidos
[63] La lista de Triggers o disparadores se encuentran en el Anexo XII Disparadores o triggers

Tablas ó Relaciones

Las tablas que se muestran a continuación son el resultado de los modelos, así como de los criterios de normalización y del análisis de los flujos de datos.

Encabezado del pedido (PEDALMGLOB).

Cada pedido cuenta con un número que se va generando secuencialmente conforme son creados por el ERP. Todos los pedidos tienen seis caracteres. Existen dos tipos de pedidos, los pedidos cliente que se identifican por que inician con dos caracteres "*PC*" y posteriormente cuentan con un número formado por cuatro dígitos. Los pedidos normales que inician con una letra "*P*" y posteriormente cinco dígitos. La estructura de los dos tipos de pedidos es idéntica. La llave primaria para el encabezado es el número de Pedido (*NUMPED*), este identificador es único por pedido, los valores de la tabla se encuentran ordenados por este campo lo que facilita su búsqueda. También este campo sirve como llave foránea para todas las tablas relacionadas al pedido.

Así mismo, PEDALMGLOB (fig.39) cuenta con cuatro llaves foráneas relacionadas con las tablas PROCESOS, TIPOPEDIDO, TIPOPREPARADO, y CLIENTESGLOB, impidiendo insertar un número de cliente, formas de preparado, tipos de preparado, secciones o procesos inexistentes.

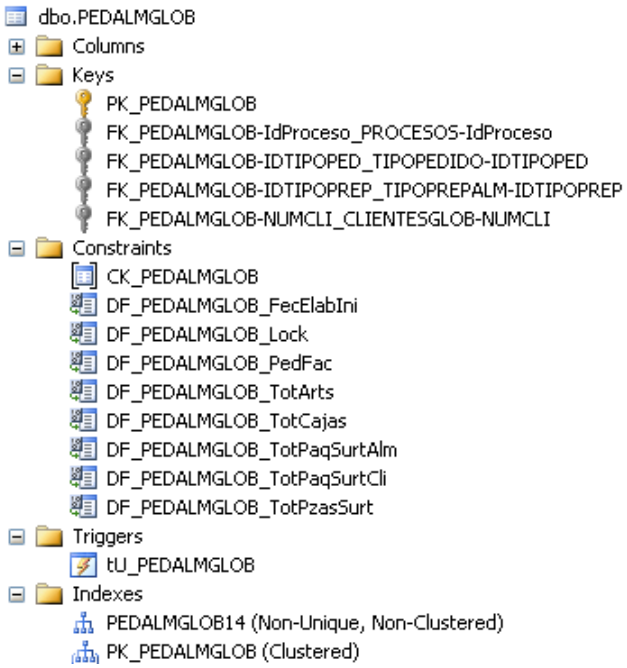


FIG. 39 ESTRUCTURA PEDALMGLOB

Cuando se inserta un Pedido a la tabla PEDALMGLOB, automáticamente agrega valores como es la fecha de elaboración, tomando la fecha y hora del servidor, asignándole cero productos surtidos y habilitando la bandera que permite modificar el pedido.

```
 CONSTRAINT [DF_PEDALMGLOB_FecElabIni] DEFAULT (getdate()) FOR FecElabIni],
CONSTRAINT [DF_PEDALMGLOB_TotArts] DEFAULT (0) FOR [TotArts],
CONSTRAINT [DF_PEDALMGLOB_TotCajas] DEFAULT (0) FOR [TotCajas],
CONSTRAINT [DF_PEDALMGLOB_TotPzasSurt] DEFAULT (0) FOR [TotPzasSurt],
CONSTRAINT [DF_PEDALMGLOB_TotPaqSurtCli] DEFAULT (0) FOR [TotPaqSurtCli],
CONSTRAINT [DF_PEDALMGLOB_TotPaqSurtAlm] DEFAULT (0) FOR [TotPaqSurtAlm],
CONSTRAINT [DF_PEDALMGLOB_PedFac] DEFAULT (0) FOR [PedFac],
```

```
CONSTRAINT [DF_PEDALMGLOB_Lock] DEFAULT (0) FOR [Lock]
```

En PEDALMGLOB también se verifica que tanto el pedido, el número de cliente, como en nombre contengan valores correctos.

```
CONSTRAINT[CK_PEDALMGLOB]CHECK((len([NOMCLI])>(0)AND[NUMCLI]like'[A-Z][A-Z]%[0-9][0-
9]' AND [NUMPED] like '[P]%'))
```

Un detalle a mencionar es *NUMPEDP* campo donde se almacena el número de pedido padre para aquellos pedidos que están relacionados con uno. Los pedidos consolidados son casos excepcionales donde un cliente desea que varios pedidos sean consolidados en una misma entrega y se elaboren en una misma factura. El *ERP* no está diseñado para este tipo de pedidos, por lo que un archivo consolidado solo se puede generar con un rango de pedidos individuales. Con la finalidad de facilitar el trabajo con las interfaces, y generar una estructura similar a la generada por el *ERP*, se agregó un campo a la tabla PEDALMGLOB, NUMPEDP, evitando una mayor normalización de la base de datos. Una vez que el ERP haya modificado su estructura de pedidos, está planeada la creación de la tabla PEDALMGLOBCON donde se puedan relacionar los pedidos hijos a un pedido padre, mejorando la normalización de la base de datos.

PEDALMGLOB	DESCRIPCIÓN
NUMPEDP	NUMERO DEL PEDIDO PADRE
NUMPED	**NUMERO DEL PEDIDO**
NumCli	NUMERO DEL CLIENTE
NomCli	NOMBRE DEL CLIENTE
OrdCli	ORDEN DEL CLIENTE
FecIni	FECHA DE CARGA
FecElabIni	FECHA Y HORA DE INICIO DE ELABORACIÓN DEL PEDIDO EN A.P.T
FecElabFin	FECHA Y HORA DE FIN DE ELABORACIÓN DEL PEDIDO EN A.P.T
FecVen	FECHA DE VENCIMIENTO DEL PEDIDO
TotArts	TOTAL DE ARTÍCULOS SURTIDOS
TotCajas	TOTAL DE CAJAS ELABORADAS
TotPzasSurt	TOTAL DE PIEZAS SURTIDAS
TotPaqSurtCli	TOTAL DE PAQUETES SURTIDOS (PIEZAS POR EMPAQUE SOLICITADO POR CLIENTE)
TotPaqSurtAlm	TOTAL DE PAQUETES SURTIDOS (EMPAQUES APT)
idProceso	INDICADOR DE QUE PARTE DEL PROCESO DE ENTREGA SE ENCUENTRA
PedFac	INDICADOR SI EL PEDIDO SE ENCUENTRA FACTURADO
IdTipoPed	IDENTIFICADOR DEL TIPO DE PEDIDO, ESTE PUEDE SER CONSOLIDADO O INDIVIDUAL
IdTipoPrep	IDENTIFICADOR DEL TIPO DE PREPARADO PUEDE SER EN PARTES O EN UN SOLO MOV.
Lock	IMPIDE CUALQUIER MODIFICACIÓN A LA TABLA
Obser	OBSERVACIONES

Sucursales (PEALMSUC)

La tabla PEALMSUC almacena las sucursales o tiendas a las que se enviará la mercancía. En caso de ser venta directa o tienda única se agrega la dirección del comprador.

La llave primaria para el encabezado es el número de Pedido (*NUMPED*) y el identificador por sucursal (*IDSUC*), los valores de la tabla se encuentran ordenados por estos campos lo que facilita su búsqueda.

Así mismo, PEDALMSUC cuenta con una llave foránea relacionadas con la tabla PEDALMGLOB, impidiendo insertar un pedido inexistente, y permitiendo vincular más rápidamente una sucursal con el encabezado del pedido.

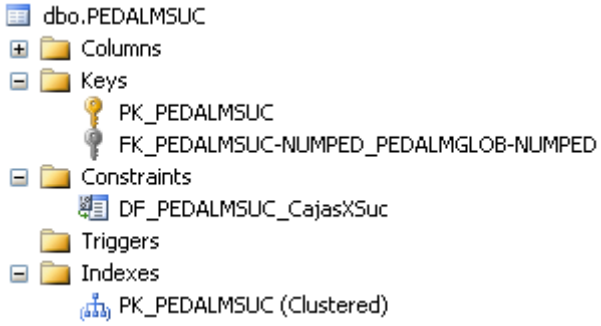


FIG. 40 ESTRUCTURA PEDALMSUC

Al insertar una sucursal se asigna por defecto un cero en la cantidad de cajas surtidas.

```
CONSTRAINT [DF_PEDALMSUC_CajasXSuc]  DEFAULT (0) FOR [CajasXSuc]
```

PEDALMSUC	DESCRIPCIÓN
NumPed	**NUMERO DEL PEDIDO**
IdSuc	**IDENTIFICADOR DE LA SUCURSAL**
NomSuc	NOMBRE DE LA SUCURSAL
Obser	OBSERVACIONES
CajasXSuc	TOTAL DE CAJAS SURTIDAS POR SUCURSAL

Cajas (PEDALMCAJAS)

A la tabla encargada de almacenar los valores generales por caja la denominé PEDALMCAJAS (fig. 41). Genera de manera automática un identificador único ***IdCajaGlob*** en cada caja**,** el cual permite rastrearla en cualquier momento. Sin embargo un identificador de este tipo no es muy efectivo, ni para las consultas ni para relacionarse con las demás tablas, por lo que cuenta con dos llaves adicionales (NUMPED, IDCAJAPED) y (NUMPED, IDSUC, IDCAJASUC). Estas llaves facilitan la identificación de la caja en el pedido, así como con la sucursal. La tabla se encuentra ordenada en base a la última llave, ya que es la manera más eficiente de agrupar los valores almacenando los datos con el mismo orden que tienen las demás tablas de los pedidos [64]. Al igual que las tablas anteriores asigna valores nulos a los totales.

Cuenta también con un estatus que permite identificar en que parte del proceso de trazabilidad se encuentra. Otro valor muy importante de esta tabla es el empleado que elaboró la caja.

[64] En la sección 3.5.5 *Administración y optimización del rendimiento de la base de datos* se verán detalles relacionados a las consultas y sus tiempos de respuesta.

PEDALMCAJAS	DESCRIPCIÓN
IdCajaGlob	**IDENTIFICADOR ÚNICO DE LA CAJA**
IdCajaPed	IDENTIFICADOR DE LA CAJA POR PEDIDO
IdCajaEtapa	IDENTIFICADOR DE LA CAJA POR ETAPA
IdCajaSuc	IDENTIFICADOR DE LA CAJA POR SUCURSAL
NumPed	NUMERO DE PEDIDO
IdSuc	IDENTIFICADOR DE LA SUCURSAL
IdEtapa	IDENTIFICADOR DE LA ETAPA
NumEmpElab	NUMERO DEL EMPLEADO QUE ELABORÓ LA CAJA
FecElab	FECHA Y HORA DE ELABORACIÓN
TotPzasSurtXCaja	TOTAL DE PIEZAS SURTIDAS EN CAJA
TotArtXCaja	TOTAL DE ARTÍCULOS EN CAJA
TotPaqSurtCli	TOTAL DE PAQUETES SURTIDOS (PIEZAS POR EMPAQUE SOLICITADO POR CLIENTE)
TotPaqSurtAlm	TOTAL DE PAQUETES SURTIDOS (PIEZAS POR EMPAQUE APT)
idConten	IDENTIFICADOR DEL CONTENEDOR
CajaStatus	INDICADOR DE QUE PARTE DEL PROCESO DE ENTREGA SE ENCUENTRA LA CAJA
Obser	OBSERVACIONES

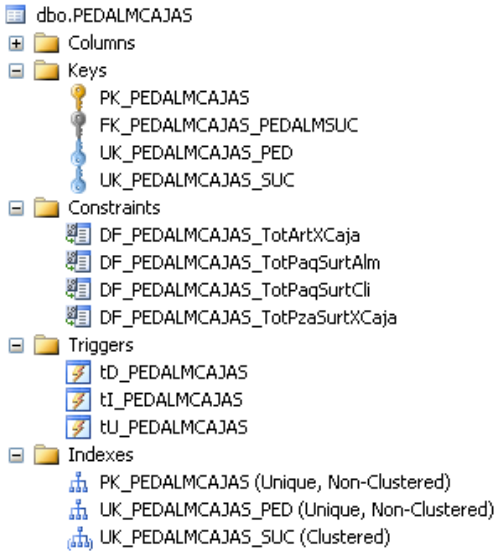


FIG. 41 ESTRUCTURA PEDALMCAJAS

Trazabilidad

A la tabla de trazabilidad la denominé PEDALMCAJASTRZ (fig. 42). En esta tabla se conserva un registro de todos los procesos, departamentos, o secciones por los que pasa una caja. Tiene su principal uso una vez terminada la elaboración del pedido y permite obtener un seguimiento preciso del flujo de la caja así como todos los empleados que han estado en contacto con ella.

PEDALMCAJASTRZ	DESCRIPCIÓN
Numped	**NUMERO DE PEDIDO**
IdCajaPed	**IDENTIFICADOR DE LA CAJA POR PEDIDO**
idProceso	**IDENTIFICADOR DE PROCESO O DEPTO. POR EL QUE ESTA PASANDO LA CAJA**
NumEmp	NUMERO DE EMPLEADO RESPONSABLE DE LA CAJA EN EL PROCESO
FechaIngreso	FECHA DE INGRESO A LA SECCIÓN, DEPTO, O PROCESO

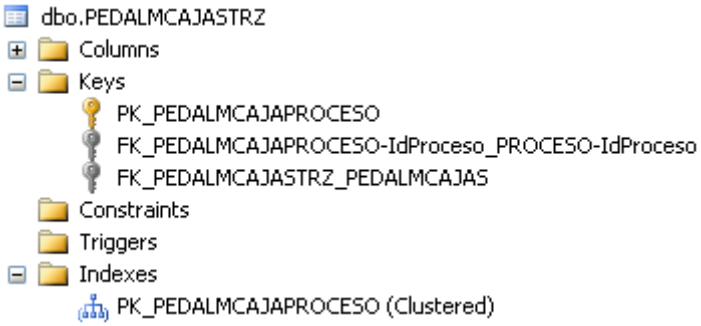


FIG. 42 ESTRUCTURA PEDALMCAJASTRZ

La llave primaria está conformada por el número del pedido, el identificado de la caja por pedido y el proceso o sección en la que se encuentra la caja (NUMPED, IDCAJAPED, IDPROCESO).

Maneja dos llaves foráneas mediante las cuales impide que se inserte o modifique un proceso que no existe en la tabla PROCESO, o que se inserte una caja que no se encuentra en la tabla PEDALMCAJAS.

Artículos

Utilizando la tercera forma normal dividí los artículos en tres tablas. En la primera tabla se encuentran los valores únicos del artículo en el pedido (PEDALMART). Una segunda tabla agrupa los artículos solicitados con sus respectivas sucursales o tiendas a las que serán enviados (PEDALMDETSOL). Por último, la tercera, los artículos surtidos agrupados por caja (PEDALMDETSURT).

a) PEDALMART (fig. 43) Esta tabla contiene todos los artículos solicitados por el cliente. En ella se almacenan los datos generales de cada artículo por pedido; éstos pueden ser los códigos especiales que requiere el cliente, la descripción del artículo, el número de piezas que contienen los paquetes, etcétera. Cabe recalcar que la descripción del artículo, el número de piezas por paquete, pueden variar con el tiempo de un pedido a otro, por lo cual no se toman directamente los valores de la tabla general de artículos (MARTIC) de la empresa o la tabla (MSKU) con códigos especiales por cliente. La llave primaria de esta tabla se encuentra compuesta por el número del pedido (NumPed) y el código del artículo UPC (CodArt).

PEDALMART	DESCRIPCIÓN
NumPed	**NÚMERO DE PEDIDO**
CodArt	CODIGO DEL ARTÍCULO UPC
CodArtP	CODIGO DEL ARTICULO ERP
Sku	SKU
DescAr	DESCRIPCIÓN DEL ARTÍCULO
NumPzaXPaqCli	CANTIDAD DE PIEZAS POR PAQUETE CLIENTE
NumPzaXPaqAlm	CANTIDAD DE PIEZAS POR PAQUETE ALMACEN
ArtSurt	ARTÍCULOS SURTIDOS

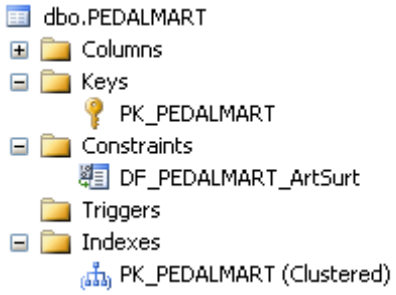


FIG. 43 ESTRUCTURA PEDALART

b) PEDALMDETSOL (fig. 44), es la tabla que contiene los valores solicitados por sucursal.

PEDALMDETSOL	DESCRIPCIÓN
NumPed	**NUMERO DE PEDIDO**
IdSuc	**IDENTIFICADOR DE SUCURSAL**
Consec	CONSECUTIVO ELABORACIÓN
CodArt	CODIGO UPC
CantPzasSol	CANTIDAD DE PIEZAS SOLICITADAS
CantPaqSolCli	CANTIDAD DE PAQUETES SOLICITADOS CLIENTE
CantPaqSolAlm	CANTIDAD DE PAQUETES SOLICITADOS ALMACEN

La llave primaria para el encabezado es el número de Pedido (NUMPED), el identificador por sucursal (IDSUC) y el código del artículo (CODART); los valores de la tabla se encuentran ordenados por estos campos lo que facilita las consultas.

Tiene llaves foráneas que impiden insertar artículos que no se encuentren en la tabla PEDALMART, así como artículos en sucursales que no se encuentran en el pedido.

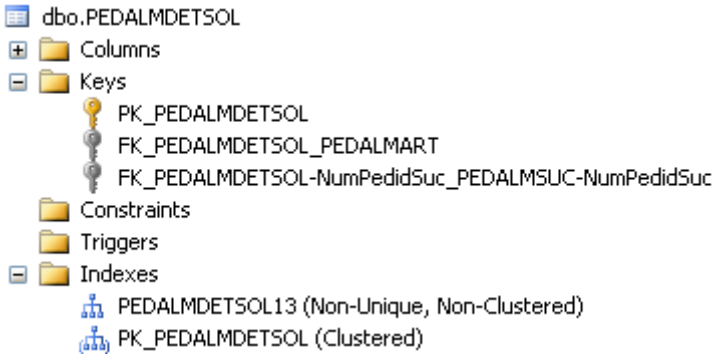


FIG. 44 ESTRUCTURA PEDALMSUC

c) PEDALMDETSURT (fig. 45), en esta tabla se encuentran los valores surtidos por caja. Su llave primaria está compuesta por el número del pedido, identificador de caja por pedido y el código del artículo (NUMPED, IDCAJAPED, CODART). Sin embargo no es este el orden en que se encuentran almacenados en la tabla. También cuenta con otra llave formada por el número del pedido (NUMPED), la sucursal (IDSUC), el identificador por caja (IDCAJAPED) y el código del artículo (CODART). Los valores de la tabla se encuentran ordenados de esta manera lo que facilita las consultas cuando se encuentran vinculadas con las demás tablas. En el caso de los pedidos consolidados la orden del cliente es la que identifica el origen del artículo.

PEDALMDETSURT	DESCRIPCIÓN
NumPed	**NUMERO DE PEDIDO**
IdSuc	**IDENTIFICADOR DE SUCURSAL**
IdCajaPed	IDENTIFICADOR CAJA
OrdCli	ORDEN CLIENTE
Consec	CONSECUTIVO
CodArt	CODIGO UPC
CantPzasSurt	CANTIDAD DE PIEZAS SURTIDAS
CantPaqSurtCli	CANTIDAD DE PAQUETES SURTIDOS EMPAQUE CLIENTE
CantPaqSurtAlm	CANTIDAD DE PAQUETES SURTIDOS EMPAQUE ALMACEN

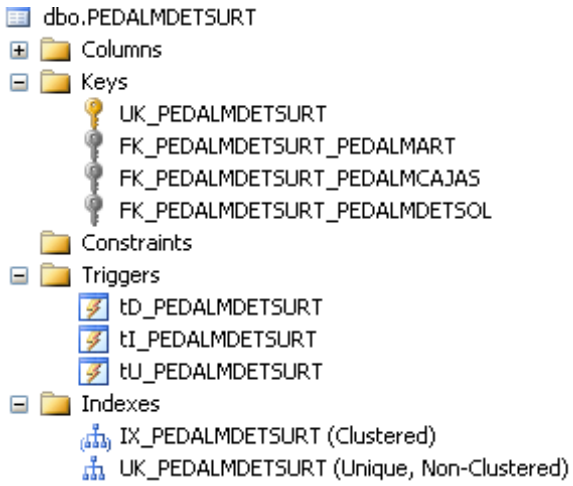


FIG. 45 ESTRUCTURA PEDALMDETSURT

Temporales

PEDALMTEMP. Esta tabla permite identificar a los empleados se encuentran elaborando un mismo pedido. Es un factor de seguridad que impide que se bloque un pedido o se impriman reportes generales cuando algún empleado se encuentra laborándolo. Cuenta con una llave primaria formada por el número de pedido y el número de empleado (NUMPED, NUMEMP). Contiene muy pocos registros, el número máximo posible es 200, ya que una vez que un empleado sale del pedido el registro es eliminado automáticamente.

PEDALMTEMPDESCRIPCIÓN	
NumPed	**NÚMERO DE PEDIDO**
NumEmp	**NÚMERO DE EMPLEADO**

Etapas

PEDALMETAPAS (fig. 46) Esta tabla se utiliza cuando un pedido cuenta con diferentes entregas de mercancías, estas entregas se hacen en fechas distintas. Al igual que los pedidos consolidados es un caso especial, donde se requiere tener un encabezado que especifique la fecha de entrega y el estado de la etapa. Los estados de una etapa pueden ser preparada, cerrada, surtida y una sola etapa vigente. Maneja por lo tanto dos llaves foráneas una impide que se inserte o modifique un estatus de etapa que no existe en la tabla STATUSETAPA, o que se inserte una etapa en un pedido que no se encuentra dado de alta en la tabla PEDALMGLOB.

PEDALMETAPASDESCRIPCIÓN	
NumPed	**NÚMERO DE PEDIDO**
IdEtapa	**IDENTIFICADOR DE LA ETAPA**
FechaEntrega	FECHA Y HORA DE ENTREGA
IdStatusEtapa	ESTATUS ETAPA

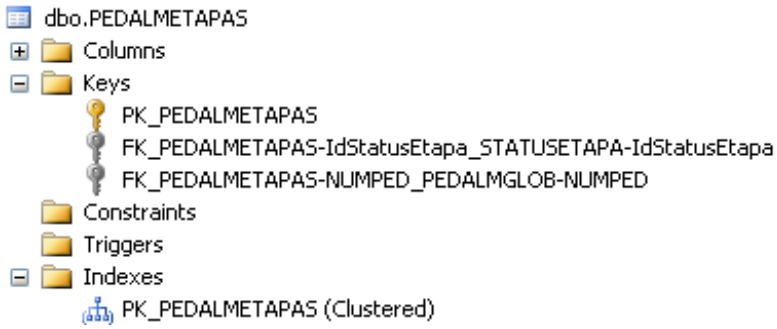


FIG. 46 ESTRUCTURA PEDALMETAPAS

3.5.4. *TRANSACCIONES A LA BASE DE DATOS*

A continuación se muestran consultas a la base de datos, mostrando las restricciones de la base.

Encabezado del pedido

En la siguiente consulta se hace una inserción a la base de datos de un pedido con valores totales y fechas de vencimiento y el campo LOCK activado indicando que el pedido no se puede elaborar

Consulta:

```
INSERT INTO PEDALMGLOB
(NUMPED,NumCli,NomCli,OrdCli,FecIni,FecElabIni,FecElabFin,FecVen,TotArts,TotCajas,
TotPzasSurt,TotPaqSurtCli,TotPaqSurtAlm,idProceso,PedFac,IdTipoPed,IdTipoPrep,Lock,Obser)
VALUES ('P55168','CHE001','TIENDAS CHEDRAUI, S.A. DE C.V.','7500131501',
'2010/10/26','2010/10/27','2010/10/27','2010/10/27',10,10,10,10,10,1,0,1,1,1,'')
```

Mensaje base de datos:

```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tI_PEDALMGLOB, Line 22
Al insertar pedido no se pueden agregar totales o bloquear pedido.
49PROG(sa): Msg 3609, Level 16, State 1, Line 3
The transaction ended in the trigger. The batch has been aborted.
```

La inserción no se efectúa debido a que el disparador *tI_PEDALMGLOB* [65] impide la inserción de totales fechas de Elaboración, ya que dichos campos se generan posteriormente de forma automática. También impide que en un pedido recién hecho se inserte una bandera que lo bloquee ya que impediría la elaboración posterior de este.

Consulta:

```
SELECT * FROM PEDALMGLOB WHERE NUMPED = 'P55168'
```

Resultado:

```
49PROG(sa): (0 row(s) affected)
```

La siguiente inserción a la base contiene todos los valores requeridos para iniciar un pedido

65 Ver Anexo XII Disparadores o triggers

Consulta:

```
INSERT INTO PEDALMGLOB (NUMPED, NumCli, NomCli, OrdCli, FecIni, FecVen, idProceso, IdTipoPed,
IdTipoPrep, Obser)
VALUES
        (
                /*NUMPED*/    'P55168',
                /*NumCli*/    'CHE001',
                /*NomCli*/    'TIENDAS CHEDRAUI, S.A. DE C.V.',
                /*OrdCli*/    '7500131501',
                /*FecIni*/    '2011/10/1',
                /*FecVen*/    '2011/12/31',
                /*idProceso*/ 1,
                /*IdTipoPed*/ 1,
                /*IdTipoPrep*/1,
                /*Obser*/     ''
        )
```

Mensaje base de datos:

```
49PROG(sa): (1 row(s) affected)
```

Se obtiene una inserción exitosa a la base de datos. Con valores iniciales en cero, con la fecha de elaboración que le asigna el servidor.[66]

Consulta:

```
SELECT * FROM PEDALMGLOB WHERE NUMPED = 'P55168'
```

Resultado:

NUMPEDP	NUMPED	NumCli	NomCli	OrdCli	FecIni	FecElabIni	FecElabFin	FecVen	TotArts	TotCajas	TotPzasSurt	TotPaqSurtCli	TotPaqSurtAlm	idProceso	PedFac	idTipoPed	idTipoPrep	Lock	Obser
NULL	P55168	CHE001	TIENDAS CHEDRAUI, S.A. DE C.V.	7500131501	2011-10-01	2011-11-30 12:22:00.000	NULL	2011-12-31	0	0	0	0	0	1	0	1	1	0	

General artículos solicitados

Una vez ingresado el pedido es necesario insertar el detalle general de artículos a surtir.

Si por alguna razón se inserta el detalle de artículos por sucursal, antes de insertar el detalle general por artículo.

Consulta:

```
INSERT INTO PEDALMDETSOL VALUES('P55168','112',1,'755860779039',12,2,2)
```

Resultado:

```
49PROG(sa): Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_PEDALMDETSOL_PEDALMART". The
conflict occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMART".
49PROG(sa):
The statement has been terminated.
```

La base de datos identifica que no se han dado de alta los artículos en la tabla PEDALMART [67]

Si por lo contrario se inserta un artículo a surtir en un pedido inexistente.

66 Ver Anexo X Creación de la base de datos

67 Ver anexo creación de tablas PEDALMDETSOL, llave foránea FK_PEDALMDETSOL_PEDALMART

Consulta:

```
INSERT INTO PEDALMART  VALUES ('P55999','755860134012','TA01001BCO4-6','1043943','TOB CRISTAL NYL BCO 4-6',6,6,0)
```

Resultado:

```
49PROG(sa): Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_PEDALMART_PEDALMGLOB". The conflict occurred in
database "ProscaiAuxP_Triggers", table "dbo.PEDALMGLOB", column 'NUMPED'.
49PROG(sa):
The statement has been terminated.
```

La base de datos identifica que el pedido no está dado de alta e impide la inserción en la base de datos.[68]

Si el pedido se encuentra dado de alta, se puede hacer la inserción de artículos

Consulta:

```
INSERT INTO PEDALMART  VALUES ('P55168','755860134012','TA01001BCO4-6','1043943','TOB CRISTAL NYL BCO 4-6',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860137013','TA01001BCO7-9','1043946','TOB CRISTAL NYL BCO 7-9',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860138010','TA01001BCO108','1043947','TOB CRISTAL NYL BCO 10-18',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860150005','TB01002SUR0-0','1323453','TOB FLORENCIA SUR 0-0',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860150012','TB01002BCO0-0','116004','TOB FLORENCIA BCO 0-0',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860151002','TB01002SUR1-3','1323457','TOB FLORENCIA SUR 1-3',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860151019','TB01002BCO1-3','116002','TOB FLORENCIA BCO 1-3',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860154010','TA01002BCO4-6','1200219','TOB FLORENCIA BCO 4-6',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860157011','TA01002BCO7-9','1200220','TOB FLORENCIA BCO 7-9',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860170003','TB01018SUR0-0','116008','TOB MULTICOLOR SUR 0-0',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860171000','TB01018SUR1-3','116009','TOB MULTICOLOR SUR 1-3',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860180019','TB01014BCO0-0','115998','TOB FANTASY C/APLIC BCO 0-0',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860214011','TA02050BCO4-6','115076','MED SURTIDA NYL BCO 4-6',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860217012','TA02050BCO7-9','115077','MED SURTIDA NYL BCO 7-9',6,6,0)
INSERT INTO PEDALMART  VALUES ('P55168','755860218019','TA02050BCO108','115078','MED SURTIDA NYL BCO 10-18',6,6,0)
```

Resultado:

```
49PROG(sa): (15 row(s) affected)
```

Sucursales

Si se intenta insertar un artículo solicitado por sucursal en una sucursal que es inexistente en el pedido, la base de datos lo impide mediante la llave foránea FK_PEDALMDETSOL-NumPedidSuc_PEDALMSUC-NumPedidSuc[69]

Consulta:

```
INSERT INTO PEDALMDETSOL VALUES('P55168','112',1,'755860779039',12,2,2)
```

Resultado:

```
The statement has been terminated.
49PROG(sa): Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_PEDALMDETSOL-NumPedidSuc_PEDALMSUC-
NumPedidSuc". The conflict occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMSUC".
49PROG(sa):
The statement has been terminated.
```

Inserción de una sucursal a un pedido inexistente.
Consulta:

```
INSERT INTO PEDALMSUC  (NumPed, idSuc, NomSuc, Obser)
VALUES   ( /*NUMPED  */ 'P55999',
           /*SUC     */ '112',
           /*NOMSUC  */ 'CANCUN V11',
           /*OBSER   */ ''
         )
```

Resultado:

```
49PROG(sa): Msg 547, Level 16, State 0, Line 1
```

[68] Ver Anexo creación de base de datos llave foránea FK_PEDALMART_PEDALMGLOB
[69] Ver Anexo creación de base de datos llave foránea FK_PEDALMDETSOL-NumPedidSuc_PEDALMSUC-NumPedidSuc

```
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_PEDALMSUC-NUMPED_PEDALMGLOB-
NUMPED". The conflict occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMGLOB", column
'NUMPED'.
49PROG(sa):
The statement has been terminated.
```

La base de datos lo impide mediante la llave foránea FK_PEDALMSUC-NUMPED_PEDALMGLOB-NUMPED[70]

Inserción valida de dos sucursales a un pedido existente.

Consulta:

```
INSERT INTO PEDALMSUC  (NumPed, idSuc, NomSuc, Obser)
VALUES   ( /*NUMPED  */ 'P55168',
           /*SUC     */ '112',
           /*NOMSUC  */ 'CANCUN V11',
           /*OBSER   */ ''
         )

49PROG(sa): (1 row(s) affected)

INSERT INTO PEDALMSUC  (NumPed, idSuc, NomSuc, Obser)
VALUES   ( /*NUMPED  */ 'P55168',
           /*SUC     */ '105',
           /*NOMSUC  */ 'OAXACA',
           /*OBSER   */ ''
        )

 49PROG(sa): (1 row(s) affected)
```

Resultado:

```
SELECT * FROM  PEDALMSUC  WHERE NUMPED  = 'P55168'
```

NumPed	idSuc	NomSuc	Obser	CajasXSuc
P55168	112	CANCUN V11		0
P55168	105	OAXACA		0

Ambas sucursales son dadas de alta en la base de datos.

Artículos solicitados por sucursal

Una vez ingresadas las sucursales se pueden dar de alta los artículos por sucursal solicitados en el pedido

Consulta:

```
INSERT INTO PEDALMDETSOL VALUES('P55168','112',1,'755860779039',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',2,'755860793004',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',3,'755860840418',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',4,'755860719011',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',5,'755860719042',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',6,'755860719035',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',7,'755860719097',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',8,'755860719059',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',9,'755860668005',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',10,'755860667008',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',11,'755860664007',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',12,'755860874307',36,6,6)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',13,'755860821905',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',14,'755860151002',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','112',15,'755860150005',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','105',1,'755860779039',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','105',2,'755860793004',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','105',3,'755860840418',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','105',4,'755860719011',12,2,2)
INSERT INTO PEDALMDETSOL VALUES('P55168','105',5,'755860719042',12,2,2)
```

Resultado:

```
49PROG(sa): (20 row(s) affected)
```

[70] Ver Anexo creación de base de datos llave foránea FK_PEDALMSUC-NUMPED_PEDALMGLOB-NUMPED

Se ingresaron los artículos por sucursal a la base de datos.

NumPed	IdSuc	Consec	CodArt	CantPzasSol	CantPaqSolCli	CantPaqSolAlm
P55168	105	1	755860134012	12	2	2
P55168	105	2	755860138010	12	2	2
P55168	105	3	755860151002	12	2	2
P55168	105	4	755860151019	12	2	2
P55168	105	5	755860157011	12	2	2
P55168	112	1	755860134012	12	2	2
P55168	112	2	755860137013	12	2	2
P55168	112	3	755860138010	12	2	2
P55168	112	4	755860150005	12	2	2
P55168	112	5	755860150012	12	2	2
P55168	112	6	755860151002	12	2	2
P55168	112	7	755860151019	12	2	2
P55168	112	8	755860154010	12	2	2
P55168	112	9	755860157011	12	2	2
P55168	112	10	755860170003	12	2	2
P55168	112	11	755860171000	12	2	2
P55168	112	12	755860180019	36	6	6
P55168	112	13	755860214011	12	2	2
P55168	112	14	755860217012	12	2	2
P55168	112	15	755860218019	12	2	2

Finalizando este proceso el pedido se encuentra listo para ser preparado terminándose la primera fase del pedido

Una vez insertados los valores iniciales de pedido se puede proseguir a la segunda fase de la preparación del pedido.

Preparado del pedido cajas

Si se inserta un artículo sin un identificador por caja.
Consulta:

```
INSERT INTO PEDALMDETSURT VALUES ('P55168','112',1,'7500131501',1,'755860134012',6,1,1)
```

Resultado:

```
49PROG(sa): Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_PEDALMDETSURT_PEDALMCAJAS". The
conflict occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMCAJAS".
49PROG(sa):
The statement has been terminated.
```

No se permite insertar un artículo a la base de datos sin que este artículo se pueda relacionar con una caja, gracias a la llave foránea FK_PEDALMDETSURT_PEDALMCAJAS [71].

Al igual que en los totales del encabezado por pedido no se pueden hacer ni inserciones, ni modificaciones en los totales de la caja.

Consulta:

```
INSERT INTO PEDALMCAJAS (IdCajaPed, IdCajaSuc, NumPed, IdSuc, NumEmpElab, FecElab, TotPzasSurtXCaja,
TotArtXCaja, TotPaqSurtCli, TotPaqSurtAlm, idConten, CajaStatus,Observ)
VALUES
        (       /*IdCajaPed*/           1,
                /*IdCajaSuc*/           1,
                /*NumPed*/              'P55168',
                /*IdSuc'*/              '112',
                /*NumEmpElab*/          '000001',
                /*FecElab*/             GETDATE(),
                /*TotPzasSurtXCaja*/    1,
                /*TotArtXCaja*/ 2,
                /*TotPaqSurtCli*/       1,
                /*TotPaqSurtAlm*/       1,
                /*idConten*/            'M000001',
```

[71] Ver anexos creación de base de datos FK_PEDALMDETSURT_PEDALMCAJAS

```
                /*CajaStatus*/          0,
                /*Observ*/              ''
        )
```

Resultado:

```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tI_PEDALMCAJAS, Line 27
Al insertar cajas no se pueden agregar totales.
49PROG(sa): Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Inserción correcta de valores de una caja

```
INSERT INTO PEDALMCAJAS (IdCajaPed, IdCajaSuc, NumPed, IdSuc, NumEmpElab, idConten, CajaStatus,Observ)
VALUES
(
        /*IdCajaPed*/           1,
        /*IdCajaSuc*/           1,
        /*NumPed*/              'P55168',
        /*IdSuc'*/              '112',
        /*NumEmpElab*/          '000001',
        /*idConten*/            'M000001',
        /*CajaStatus*/          0,
        /*Observ*/              ''
)
```

Resultado:

```
SELECT * FROM  PEDALMCAJAS WHERE NumPed = 'P55168'
```

IdCajaGlob	IdCajaPed	IdCajaEtapa	IdCajaSuc	NumPed	IdSuc	IdEtapa	NumEmpElab	FecElab	TotPzasSurtXCaja	TotArtXCaja	TotPaqSurtCli	TotPaqSurtAlm	idConten	CajaStatus	Observ
198031	1	NULL	1	P55168	112	NULL	000001	NULL	0	0	0	0	M000001	0	

Como se puede observar en la tabla anterior se genera un identificador único *IdCajaGlob*.

No se pueden insertar cajas duplicadas en la base de datos

Consulta:

```
INSERT INTO  PEDALMCAJAS (IdCajaPed, IdCajaSuc, NumPed, IdSuc,NumEmpElab, idConten , CajaStatus,Observ)
VALUES
(
        /*IdCajaPed*/           1,
        /*IdCajaSuc*/           1,
        /*NumPed*/              'P55168',
        /*IdSuc'*/              '112',
        /*NumEmpElab*/          '000001',
        /*idConten*/            'M000001',
        /*CajaStatus*/          0,
        /*Observ*/              ''
)
```

Resultado:

```
49PROG(sa): Msg 2627, Level 14, State 1, Line 1
Violation of UNIQUE KEY constraint 'UK_PEDALMCAJAS_SUC'. Cannot insert duplicate key in object
'dbo.PEDALMCAJAS'.
49PROG(sa):
The statement has been terminated.
```

Preparado del pedido artículos

Una vez que se ingresaron los registros de caja a la base de datos, se pueden ingresar los registros de los artículos que se encuentran en la caja.
Consulta:

```
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',1,'755860134012',6,1,1)
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',3,'755860138010',6,1,1)
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',4,'755860150005',12,2,2)
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',6,'755860151002',12,2,2)
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',7,'755860151019',12,2,2)
INSERT INTO PEDALMDETSURT VALUES ( 'P55168','112',1,'7500131501',8,'755860154010',12,2,2)
```

Resultado:

```
49PROG(sa): (6 row(s) affected)
```

Al ingresar los artículos surtidos en la base de datos, las cantidades totales y la fecha de elaboración de la caja (PEDALMCAJAS) son actualizadas automáticamente.[72]

```
SELECT * FROM PEDALMDETSURT  WHERE NumPed = 'P55168' AND IdCajaPed = 1
```

NumPed	IdSuc	IdCajaPed	OrdCli	Consec	CodArt	CantPzasSurt	CantPaqSurtCli	CantPaqSurtAlm
P55168	112	1	7500131501	1	755860134012	6	1	1
P55168	112	1	7500131501	3	755860138010	6	1	1
P55168	112	1	7500131501	4	755860150005	12	2	2
P55168	112	1	7500131501	6	755860151002	12	2	2
P55168	112	1	7500131501	7	755860151019	12	2	2
P55168	112	1	7500131501	8	755860154010	12	2	2

```
SELECT * FROM  PEDALMCAJAS WHERE NumPed = 'P55168' AND IdCajaPed = 1
```

IdCajaGlob	IdCajaPed	IdCajaEtapa	IdCajaSuc	NumPed	IdSuc	IdEtapa	NumEmpElab	FecElab	TotPzasSurtXCaja	TotArtXCaja	TotPaqSurtCli	TotPaqSurtAlm	idConten	CajaStatus	Observ
198031	1	NULL	1	P55168	112		000001	30/11/2011 15:02:25	60	6	10	10	M000001	0	

Este proceso se repite hasta que se finaliza el preparado del pedido.

Finalmente la tercera fase comienza cuando se generan los totales del pedido. Éstos se generan automáticamente cuando se modifica la bandera LOCK del encabezado del pedido que se encuentra en la tabla PEDALMGLOB.[73]
Consulta:

```
UPDATE PEDALMGLOB SET LOCK = 1 WHERE NUMPED = 'P55168'
```

Resultado:

```
49PROG(sa): (1 row(s) affected)
```

```
SELECT * FROM PEDALMGLOB WHERE NUMPED = 'P55168'
```

NUMPEDP	NUMPED	NumCli	NomCli	OrdCli	FecIni	FecElabIni	FecElabFin	FecVen	TotArts	TotCajas	TotPzasSurt	TotPaqSurtCli	TotPaqSurtAlm	idProceso	PedFac	IdTipoPed	IdTipoPrep	Lock	Obser
NULL	P55168	CHE001	TIENDAS CHEDRAUI, S.A. DE C.V.	7500131501	2011-10-01	2011-12-01 16:46	2011-12-01 17:07	2011-12-31	12	3	216	36	36	1	0	1	1	1	

[72] Ver Anexo triggers `tI_PEDALMDETSURT`

[73] Ver anexos triggers tU_PEDALMGLOB

Si se deseara volver a bloquear el pedido o regresar su estado, la base de datos lo impide.

Consulta:

```
UPDATE PEDALMGLOB SET LOCK = 1 WHERE NUMPED = 'P55168'
```

Resultado:

```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tU_PEDALMGLOB, Line 38
No se puede modificar. Pedido bloqueado
49PROG(sa): Msg 3609, Level 16, State 1, Line 2
The transaction ended in the trigger. The batch has been aborted.
```

Para poder cambiar el estado de un pedido es necesario ejecutar un procedimiento almacenado *SP_HabilitaPedido*[74] para poder habilitar el pedido.

Eliminación de registros

Eliminación de artículos surtidos.

Como ya se indicó, la base de la estructura en la cual se sostiene la elaboración de pedidos es la caja y su contenido. Esto permite agilidad en el tiempo de consultas, una inserción consistente y tiempos razonables. Para que el modelo diseñado en la base de datos sea consistente se requiere que la estructura "artículos surtidos por caja" se comporte como un solo elemento. Por lo cual existen varios trigger que impiden que un artículo sea borrado o se modifiquen sus cantidades individualmente.

Consulta:

```
DELETE PEDALMDETSURT WHERE  NumPed = 'P55168' AND CodArt = '755860134012'
```

Resultado:

```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tD_PEDALMDETSURT, Line 20
No se puede modificar artículos individuales, se necesita borrar caja
49PROG(sa): Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Consulta:

```
UPDATE PEDALMDETSURT set  CantPzasSurt = 6
WHERE  NumPed = 'P55168'
AND CodArt    = '755860134012'
AND IdSuc     = '105'
```

Resultado:

```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tU_PEDALMDETSURT, Line 21
No se puede modificar artículos individuales, se necesita borrar caja
49PROG(sa): Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

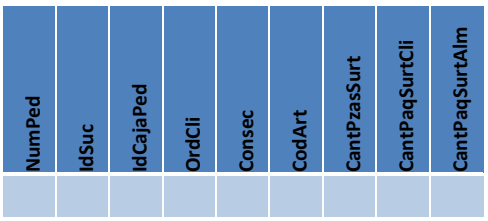
La única manera en que se puede modificar un artículo es eliminando la caja completa y volviéndola a hacer. Al eliminar una caja los artículos de esta son eliminados de la base de datos PEDALMDETSURT[75]

Consulta:

```
DELETE PEDALMCAJAS WHERE NUMPED = 'P55168' AND IDCAJAPED = 2
49PROG(sa): (1 row(s) affected)


SELECT * FROM PEDALMDETSURT
WHERE NUMPED = 'P55168' AND IDCAJAPED = 2
```

[74] Ver Anexo Procedimientos Almacenados *SP_HabilitaPedido*

[75] Ver Anexo trigger tD_PEDALMCAJAS



Eliminación de pedidos

Para poder eliminar registros de pedidos en la base de datos se requiere seguir un determinado orden. Además un pedido no puede ser eliminado si se encuentra bloqueado con la bandera LOCK.
El orden que se debe de seguir para poder eliminar un pedido es el siguiente

1. PEDALMCAJAS
2. PEDALMDETSOL
3. PEDALMART
4. PEDALMSUC
5. PEDALMGLOB

Si no se sigue este orden no se puede eliminar el pedido.

Consulta:
```
DELETE PEDALMGLOB WHERE NUMPED = 'P55168'
```

Resultado:
```
49PROG(sa): Msg 547, Level 16, State 0, Line 1
The DELETE statement conflicted with the REFERENCE constraint "FK_PEDALMART_PEDALMGLOB". The conflict
occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMART", column 'NumPed'.
49PROG(sa):
The statement has been terminated.

49PROG(sa): Msg 547, Level 16, State 0, Line 3
The DELETE statement conflicted with the REFERENCE constraint "FK_PEDALMSUC-NUMPED_PEDALMGLOB-NUMPED".
The conflict occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMSUC", column 'NumPed'.
49PROG(sa):
The statement has been terminated.
```

Consulta:
```
DELETE PEDALMART WHERE NUMPED = 'P55168'
```

Resultado:
```
9PROG(sa): Msg 547, Level 16, State 0, Line 1
The DELETE statement conflicted with the REFERENCE constraint "FK_PEDALMDETSOL_PEDALMART". The conflict
occurred in database "ProscaiAuxP_Triggers", table "dbo.PEDALMDETSOL".
49PROG(sa):
The statement has been terminated.
```

Si el pedido se encuentra bloqueado tampoco se puede eliminar el pedido ni se pueden eliminar cajas.

Consulta:
```
DELETE PEDALMCAJAS WHERE NUMPED = 'P55168'
DELETE PEDALMDETSOL WHERE NUMPED = 'P55168'
DELETE PEDALMART WHERE NUMPED = 'P55168'
DELETE PEDALMSUC WHERE NUMPED = 'P55168'
DELETE PEDALMGLOB WHERE NUMPED = 'P55168'
```

Resultado:
```
49PROG(sa): Msg 50000, Level 16, State 1, Procedure tD_PEDALMCAJAS, Line 27
El pedido se encuentra bloqueado, no se pude eliminar
49PROG(sa): Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Eliminación del pedido
Consulta:

```
EXECUTE SP_HabilitaPedido 'P55168'

DELETE PEDALMCAJAS  WHERE NUMPED = 'P55168'
DELETE PEDALMDETSOL        WHERE NUMPED = 'P55168'
DELETE PEDALMART           WHERE NUMPED = 'P55168'
DELETE PEDALMSUC           WHERE NUMPED = 'P55168'
DELETE PEDALMGLOB   WHERE NUMPED = 'P55168'
```

Resultado:
El pedido es borrado.

```
49PROG(sa): (1 row(s) affected)
49PROG(sa): (3 row(s) affected)
49PROG(sa): (20 row(s) affected)
49PROG(sa): (15 row(s) affected)
49PROG(sa): (2 row(s) affected)
49PROG(sa): (1 row(s) affected)
```

3.5.5. *ADMINISTRACIÓN Y OPTIMIZACIÓN DEL RENDIMIENTO DE LA BASE DE DATOS*

En una base real donde cualquier pérdida o falla en el manejo de la información tiene repercusiones serias, es muy trascendental poder anticipar cualquier anomalía. Por lo tanto una parte muy importante del desarrollo de la base de datos, está relacionada con la optimización de los datos y su futura administración. Es un tema muy amplio que no se tratará completamente en este trabajo. En esta sección se examinarán temas referentes a pruebas de estrés y monitoreo de la base de datos analizando los planes de consulta y el seguimiento de todos los eventos que se ejecutan en el servidor; herramientas que sirvieron para mejorar el desempeño de la base de datos, en particular los tiempos de espera de consulta remota, y la número de eventos o consultas que se requieren realizar para la elaboración de una misma tarea.

Hay varios límites que deben de ser tomados en cuenta para determinar que la base de datos está funcionando adecuadamente. Un límite importante es el *tiempo de espera de consulta remota* para especificar, en segundos, el tiempo que puede durar una operación remota antes de que *Microsoft SQL Server* suspenda la consulta y se genere un bloqueo. El valor predeterminado es de 600 segundos lo que indica que se permite una espera de diez minutos antes de que el servidor interrumpa la consulta. Con este dato se puede determinar el número máximo estimado de consultas simultáneas que se pueden hacer antes de que la base de datos tenga un bloqueo.

Planes de ejecución

En la parte referente a planes de ejecución era importante determinar el tiempo total de procesamiento de una consulta utilizando tanto la metodología de disparadores o triggers propuesta como una sin ellos.

Para analizar el plan de consultas se utilizo la herramienta *Microsoft SQL Server Management Studio*[76] de *Microsoft SQL Server* que permitió realizar una serie de consultas a la base cuando esta tenía 5,000 pedidos y unas 100,000 cajas surtidas. Arbitrariamente decidimos, el director de sistemas y yo, que el sistema debía permitir, doscientas inserciones idénticas simultáneas a la base de datos para ser considerada válida. Este número de inserciones representa casi seis veces el número total de máquinas disponibles actualmente.

Las consultas más habituales y que consumen el mayor número de recursos en la base de datos son:

- Consulta de artículos totales por pedido
- Consulta de artículos totales por pedido y sucursal
- Consulta de artículos por caja

En la inserción de datos, el modelo sin *triggers* (ver Sección *Criterios de normalización, redundancia y rendimiento*) es más rápido que el modelo con *triggers,* sin embargo el tiempo promedio para cualquier inserción a la base utilizando el modelo de triggers resulto ser menor a 0.05 segundos (ver más adelante Análisis mediante Microsoft SQL Server Management Studio), lo que significa que con 600 segundos de *tiempo de espera por consulta remota* se necesitan más de 12,000 inserciones simultáneas para que se

[76] Herramienta gráfica interactiva que permite a un administrador o a un programador de bases de datos escribir consultas, ejecutar simultáneamente varias consultas, ver los resultados, analizar el plan de consultas y recibir asistencia para mejorar el rendimiento de las consultas. Las opciones del Plan de ejecución muestran gráficamente los métodos de recuperación de datos elegidos por el optimizador de consultas de SQL Server. El plan de ejecución gráfico utiliza iconos, ordenados de una forma específica, (Ver Anexo XIII Interpretación e iconos de plan de ejecución gráfico (SQL Server Management Studio)), para representar la ejecución de instrucciones y consultas específicas en SQL Server. Además SQL Server Management Studio muestra las estadísticas y tiempos totales de ejecución.

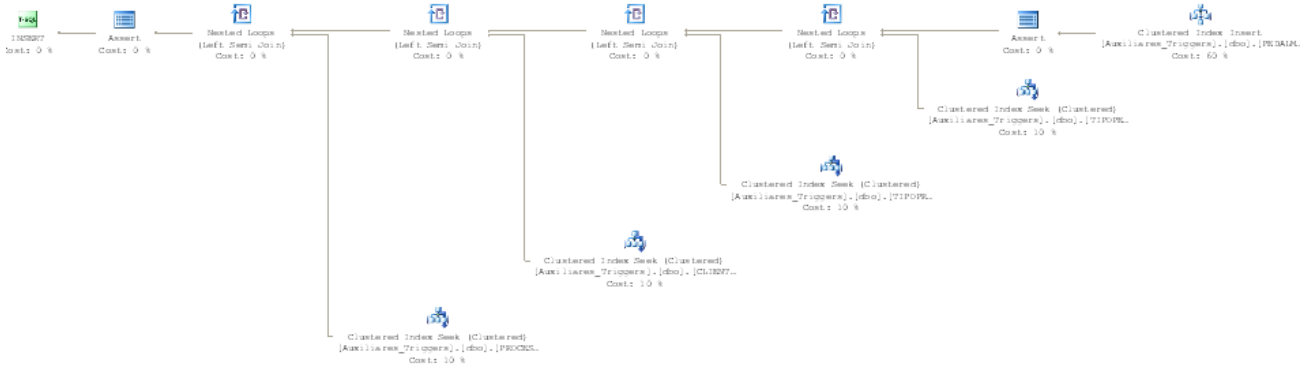
bloquee el servidor, por lo tanto es poco factible que hayan bloqueos debido a una inserción utilizando la base diseñada con triggers, y el retardo es imperceptible para el usuario. No obstante las consultas a la base de datos si pueden representar un problema como se muestra en análisis de las *consultas totales de artículos por pedido y sucursal* (Ver Pág. 120), donde las consultas sin trigger duran aproximadamente 30 segundos, solo se necesita hacer 20 consultas simultáneas para bloquear el sistema. Esta consulta no cumple con el número mínimo de inserciones simultáneas esperadas. Es muy factible que, con el crecimiento actual, una base de datos sin triggers se bloqué en un par de años.[77]

Análisis mediante Microsoft SQL Server Management Studio

Inserción de valores a base de datos con triggers

PEDALMGLOB

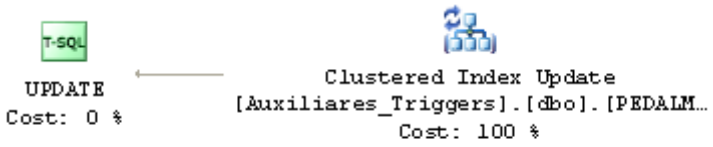
```
INSERT INTO PEDALMGLOB (NUMPED,NumCli,NomCli,OrdCli,FecIni,FecVen,idProceso,IdTipoPed,IdTipoPrep,Obser)
VALUES  (
            /*NUMPED*/   'P55168',
            /*NumCli*/   'CHE001',
            /*NomCli*/   'TIENDAS CHEDRAUI, S.A. DE C.V.',
            /*OrdCli*/   '7500131501',
            /*FecIni*/   '2011/10/1',
            /*FecVen*/   '2011/12/31',
            /*idProceso*/ 1,
            /*IdTipoPed*/ 1,
            /*IdTipoPrep*/1,
            /*Obser*/    ''
        )
```



PERFIL DE CONSULTA (INSERT PEDALMGLOB)	CONSULTA I
INSERT, DELETE ó UPDATE en consulta	2
Filas afectadas por INSERT, DELETE, ó UPDATE	1
Cantidad de SELECT	2
Filas consultadas por SELECT	1
Número de transacciones	2
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	18
Bytes enviados por cliente	936
Bytes recibidos del servidor	72,261
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**0**
Tiempo total de ejecución	**0**

[77] Para ver análisis de consultas ver Anexo XIV - Planes de ejecución y estadísticas de consultas

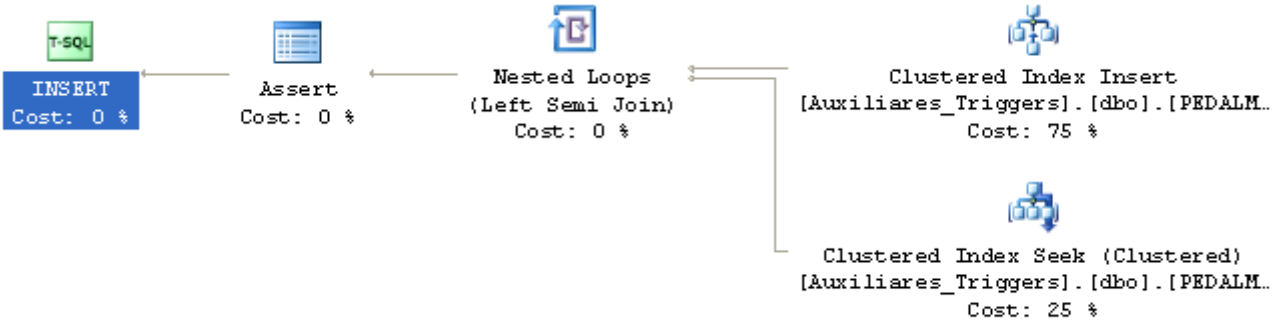
```
UPDATE PEDALMGLOB SET LOCK = 1 WHERE NUMPED = 'P55168'
```



PERFIL DE CONSULTA (UPDATE PEDALMGLOB)	CONSULTA II
INSERT, DELETE ó UPDATE en consulta	4
Filas afectadas por INSERT, DELETE, ó UPDATE	2
Cantidad de SELECT	4
Filas consultadas por SELECT	4
Número de transacciones	4
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	13
Bytes enviados por cliente	284
Bytes recibidos del servidor	52481
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**0**
Tiempo total de ejecución	**0**

PEDALMART

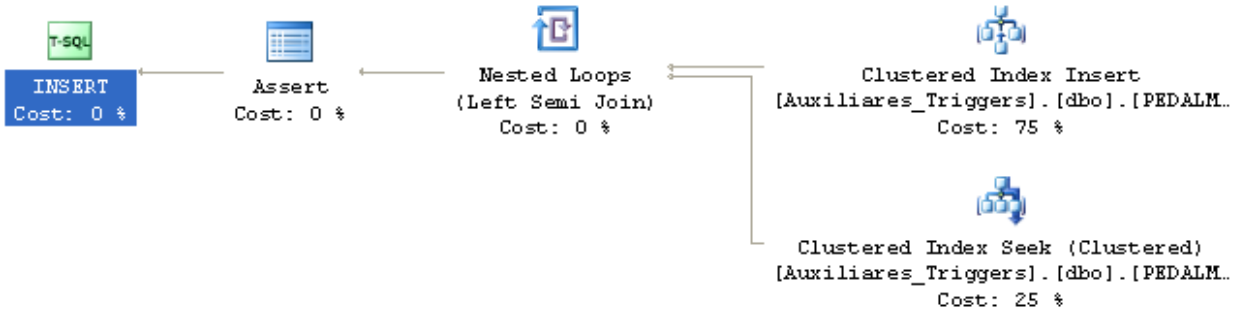
```
INSERT INTO PEDALMART  VALUES ('P55168','755860134012','TA01001BCO4-6','1043943','TOB
CRISTAL NYL BCO 4-6',6,6,0)
```



PERFIL DE CONSULTA (PEDALMART)	CONSULTA
INSERT, DELETE ó UPDATE en consulta	2
Filas afectadas por INSERT, DELETE, ó UPDATE	1
Cantidad de SELECT	1
Filas consultadas por SELECT	1
Número de transacciones	2
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	5
Bytes enviados por cliente	406
Bytes recibidos del servidor	16,518
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**0**
Tiempo total de ejecución	**0**

PEDALMSUC

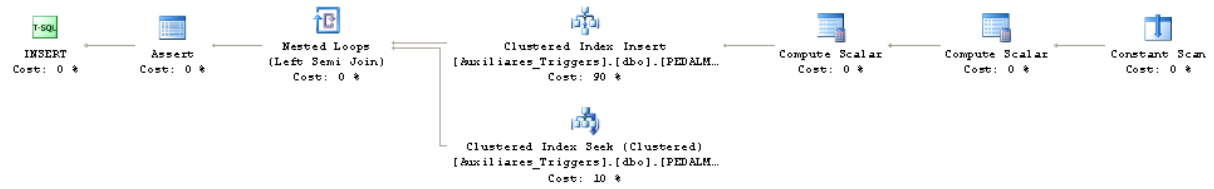
```
INSERT INTO PEDALMSUC  (NumPed, idSuc, NomSuc, Obser)
VALUES    ( /*NUMPED  */ 'P55168',
            /*SUC     */ '105',
            /*NOMSUC  */ 'OAXACA',
            /*OBSER   */ ''
          )
```



PERFIL DE CONSULTA (PEDALMSUC)	CONSULTA
INSERT, DELETE ó UPDATE en consulta	2
Filas afectadas por INSERT, DELETE, ó UPDATE	1
Cantidad de SELECT	1
Filas consultadas por SELECT	1
Número de transacciones	2
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	3
Bytes enviados por cliente	630
Bytes recibidos del servidor	11954
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**15**
Tiempo total de ejecución	**37**

PEDALMCAJAS

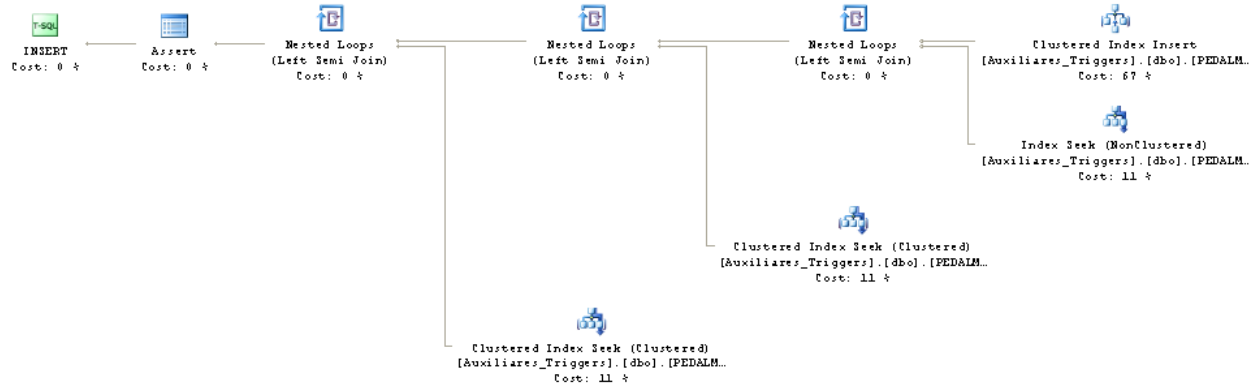
```
INSERT INTO  PEDALMCAJAS (IdCajaPed, IdCajaSuc, NumPed, IdSuc,NumEmpElab, idConten ,
CajaStatus,Observ)

VALUES
(
        /*IdCajaPed*/  1,
        /*IdCajaSuc*/  1,
        /*NumPed*/     'P55168',
        /*IdSuc'*/     '105',
        /*NumEmpElab*/ '000001',
        /*idConten*/   'M000001',
        /*CajaStatus*/ 0,
        /*Observ*/     ''
)
```



PERFIL DE CONSULTA (PEDALMCAJAS)	CONSULTA
INSERT, DELETE ó UPDATE en consulta	2
Filas afectadas por INSERT, DELETE, ó UPDATE	1
Cantidad de SELECT	4
Filas consultadas por SELECT	1
Número de transacciones	2
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	15
Bytes enviados por cliente	804
Bytes recibidos del servidor	59087
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**31**
Tiempo total de ejecución	**31**

PEDALMDETSURT

```
INSERT INTO PEDALMDETSURT VALUES ('P55168','105',1,'7500131501',1,'755860134012',6,1,1)
```



PERFIL DE CONSULTA (PEDALMDETSURT)	CONSULTA
INSERT, DELETE ó UPDATE en consulta	6
Filas afectadas por INSERT, DELETE, ó UPDATE	3
Cantidad de SELECT	3
Filas consultadas por SELECT	3
Número de transacciones	6
Red	
Número de viajes ida y vuelta al servidor	1
Paquetes TDS enviados por cliente	3
Paquetes TDS enviados por servidor	22
Bytes enviados por cliente	352
Bytes recibidos del servidor	88,226
Tiempos (Milisegundos)	
Procesamiento cliente	**0**
Tiempo de espera respuesta servidor	**15**
Tiempo total de ejecución	**15**

Consultas totales de artículos

a) Utilizando la solución propuesta

```
SELECT * FROM PEDALMGLOB
```

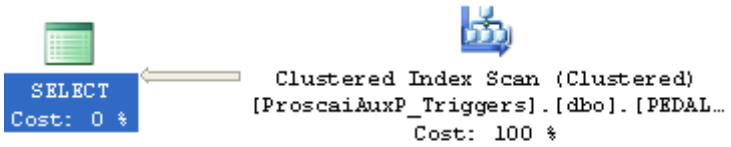


FIG. 47 PLAN DE EJECUCCIÓN TOTAL ARTÍCULOS CON TRIGGERS

b) Utilizando la base de datos sin disparadores o triggers

```
SELECT PG.NumPed, PG.ORDCLI, PG.NUMCLI,
  SUM(CantPzasSurt)        As TotPzasSurt,
  SUM(CantPaqSurtCli)      As TotPaqSurtCli,
  SUM(CantPaqSurtAlm)      As TotPaqSurtAlm
FROM     PEDALMGLOB PG, PEDALMDETSURT PDSU
WHERE    PG.NumPed      *= PDSU.NumPed
GROUP BY
       PG.NumPed, PG.ORDCLI, PG.NUMCLI
```

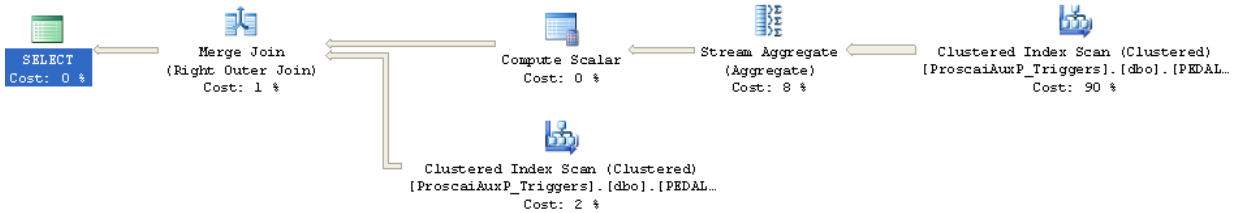


FIG. 48 PLAN DE EJECUCCIÓN TOTAL ARTÍCULOS SIN TRIGGERS

PERFIL DE CONSULTA	CONSULTA A	COSULTA B
INSERT, DELETE ó UPDATE en consulta	-	-
Filas afectadas por INSERT, DELETE, ó UPDATE	-	-
Cantidad de SELECT	2	2
Filas consultadas por SELECT	4,829	4,829
Número de transacciones	-	-
Red		
Número de viajes ida y vuelta al servidor	1	1
Paquetes TDS enviados por cliente	3	3
Paquetes TDS enviados por servidor	175	54
Bytes enviados por cliente	226	762
Bytes recibidos del servidor	716,432	220,656
Tiempos (Milisegundos)		
Procesamiento cliente	0	359
Tiempo de espera respuesta servidor	93	140
Tiempo total de ejecución	93	499

Consultas totales de artículos por pedido y sucursal

a) Utilizando la solución propuesta

```
SELECT  NumPed, idSuc,       SUM(TotPzasSurtXCaja),
        SUM(TotPaqSurtCli), SUM(TotPaqSurtAlm)
FROM    PEDALMCAJAS PC

GROUP BY NumPed, idSuc
ORDER BY NumPed, idSuc
```

FIG. 49 CONSULTA TOTALES DE ARTÍCULOS POR PEDIDO Y SUCURSAL CON TRIGGERS

b) Utilizando la base de datos sin disparadores o triggers

```
SELECT  PS.NumPed, PS.idSuc,
        SUM(PD.NumPzas)    As TotPzasSurt,
        SUM(PD.NumPaqCli) As TotPaqSurtCli,
        SUM(PD.NumPaqAlm) As TotPaqSurtAlm
FROM            PEDALMSUC PS, PEDALMCAJAS PC, PEDALMDETSURT PD
WHERE           PS.NumPed     *= PC.NumPed
AND             PD.NumPed      = PC.NumPed
AND             PS.IdSuc      *= PC.IdSuc
AND             PD.IdSuc       = PC.IdSuc
GROUP BY
        PS.NumPed, PS.idSuc
ORDER BY
        PS.NumPed, PS.idSuc
```

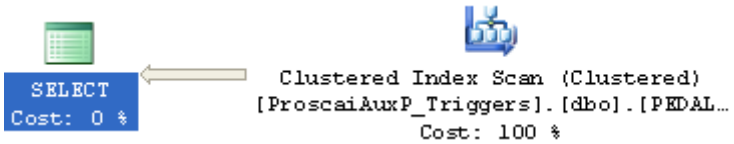
FIG. 50 CONSULTA TOTALES DE ARTÍCULOS POR PEDIDO Y SUCURSAL SIN TRIGGERS

PERFIL DE CONSULTA	CONSULTA A	COSULTA B
INSERT, DELETE ó UPDATE en consulta	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0
Cantidad de SELECT	2	2
Filas consultadas por SELECT	83,449	83,449
Número de transacciones	0	0
Red		
Número de viajes ida y vuelta al servidor	1	1
Paquetes TDS enviados por cliente	3	3
Paquetes TDS enviados por servidor	600	610
Bytes enviados por cliente	542	954
Bytes recibidos del servidor	2,457,305	2,495,509
Tiempos (Milisegundos)		
Procesamiento cliente	**0**	**0**
Tiempo de espera respuesta servidor	**218**	**625**
Tiempo total de ejecución	**343**	**25,609**

Totales de artículo por caja

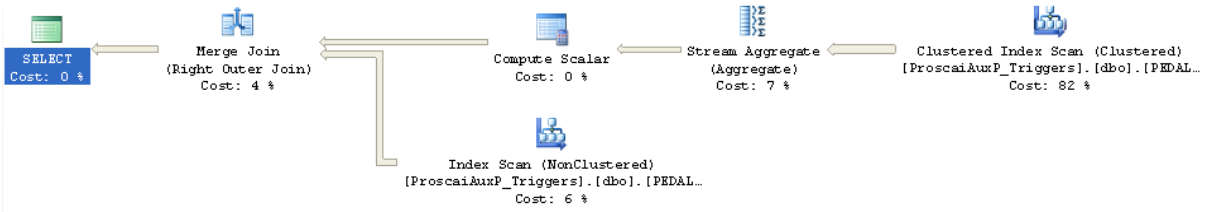
a) Utilizando la solución propuesta

```
SELECT * FROM PEDALMCAJAS
```



b) Utilizando la base de datos sin disparadores o triggers

```
SELECT  PC.NumPed, PC.IdCajaPed,
        SUM(CantPzasSurt)       As TotPzasSurt,
        SUM(CantPaqSurtCli)     As TotPaqSurtCli,
        SUM(CantPaqSurtAlm)     As TotPaqSurtAlm
FROM    PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE   PC.NumPed      *= PDSU.NumPed
GROUP BY
        PC.NumPed, PC.IdCajaPed
```



PERFIL DE CONSULTA	CONSULTA A	COSULTA B
INSERT, DELETE ó UPDATE en consulta	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0
Cantidad de SELECT	2	2
Filas consultadas por SELECT	92,041	92,041
Número de transacciones	0	0
Red		
Número de viajes ida y vuelta al servidor	1	1
Paquetes TDS enviados por cliente	3	3
Paquetes TDS enviados por servidor	564	728
Bytes enviados por cliente	408	788
Bytes recibidos del servidor	2,908,279	2,981,336
Tiempos (Milisegundos)		
Procesamiento cliente	250	0
Tiempo de espera respuesta servidor	62	421
Tiempo total de ejecución	312	421

Microsoft SQL server profiler

Esta herramienta fue útil para la creación de la base de datos, para diseñar el modelo de triggers, y para el diseño de las interfaces pues permite determinar una metodología en la inserción de datos. [78]

Mediante el *SQL Server Profiler* se pudo determinar que las consultas totales a los pedidos no se hacían cuando los pedidos se estaban elaborando, lo que hacía innecesario actualizar constantemente los totales de artículos y también, que dichas consultas podrían ser un factor de bloqueo a la base de datos. No obstante, por la estructura misma de los pedidos, se pueden realizar consultas tipo NO LOCK[79].

A diferencia de los pedidos grandes que deben surtirse a una sola sucursal, los pedidos grandes para un cliente que cuenta con varias sucursales son los que pueden generar más bloqueos por la cantidad de sumatorias que se realizan para obtener los totales y por la cantidad de gente que trabaja al mismo tiempo en el pedido,

Este último tipo de pedido se preparara al mismo tiempo en distintas mesas lo cual facilita el control administrativo del pedido por parte del personal del APT haciendo generalmente de esta manera, una sola entrega/salida que engloba, en uno solo, todos los pedidos por sucursal. Simultáneamente, como mencioné anteriormente, el encargado del APT equilibra el trabajo del pedido distribuyendo reportes de artículos totales a surtir por mesa. Los artículos solicitados en los pedidos se consultan en su totalidad al inicio de la carga y, posteriormente, se insertan solamente pequeños valores de artículos surtidos.[80] El trabajo se organiza de esta manera pues es preferible que se produzca un problema de bloqueo antes de que se haya preparado un pedido; las inserciones de datos pequeñas hacen que el escaneo de la mercancía en el preparado sea muy fluido lo que evita tiempos muertos y alienta el trabajo del empacador.

[78] SQL Server Profiler de Microsoft es una interfaz gráfica de usuario de Seguimiento SQL que se utiliza para supervisar una instancia de Motor de base de datos o de Analysis Services. Puede capturar y guardar datos acerca de cada evento en un archivo o en una tabla para analizarlos posteriormente. Por ejemplo, puede supervisar un entorno de producción para ver qué procedimientos almacenados afectan negativamente al rendimiento al ejecutarse demasiado lentamente

[79] Especifica un intervalo de nivel de table que impide el bloqueo de la consulta.

[80] Ver Transacciones a la base de datos en la sección 3.5.4

El SQL Server Profiler permitió también determinar que el tiempo promedio de inserción a la base de datos, en temporada alta, es 3 segundos aproximadamente -con lapsos de 0.5 segundos- pero no sostenidos.

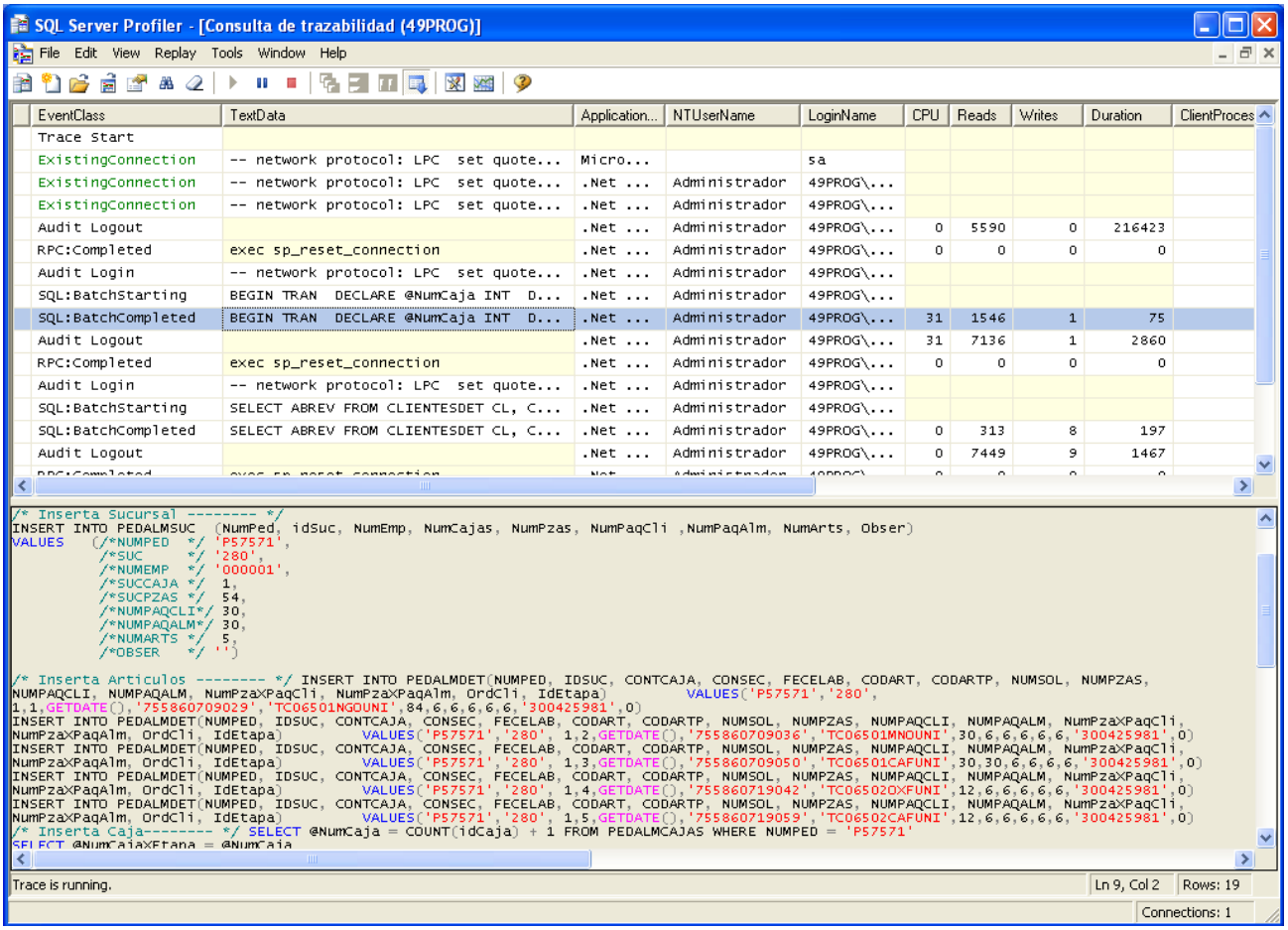


FIG. 51 INSERCIÓN DE UN PEDIDO VISTA A TRAVÉS DE MICROSOFT SQL SERVER PROFILER

3.6. DISEÑO DE LAS INTERFACES

Para poder controlar la salida de información y productos mediante procesos de distribución y trazabilidad requiere diferentes tipos de procedimientos y consultas dependiendo de la fase en la que se encuentre la elaboración de un pedido. Desarrollé diferentes interfaces para cumplir con dichas necesidades, haciendo mención de las aplicaciones más indispensables para la elaboración de un pedido.

En el siguiente diagrama (fig. 52) se muestran las principales interfaces, su secuencia de uso y los departamentos que las utilizan. El funcionamiento de cada una de ellas se explica brevemente a continuación y, en mayor detalle, el *programa de surtimiento de mercancía* que es el programa medular del sistema.

3.6.1. *DIAGRAMA SECUENCIAL Y DESCRIPCIÓN DE LAS INTERFACES DEL SISTEMA*

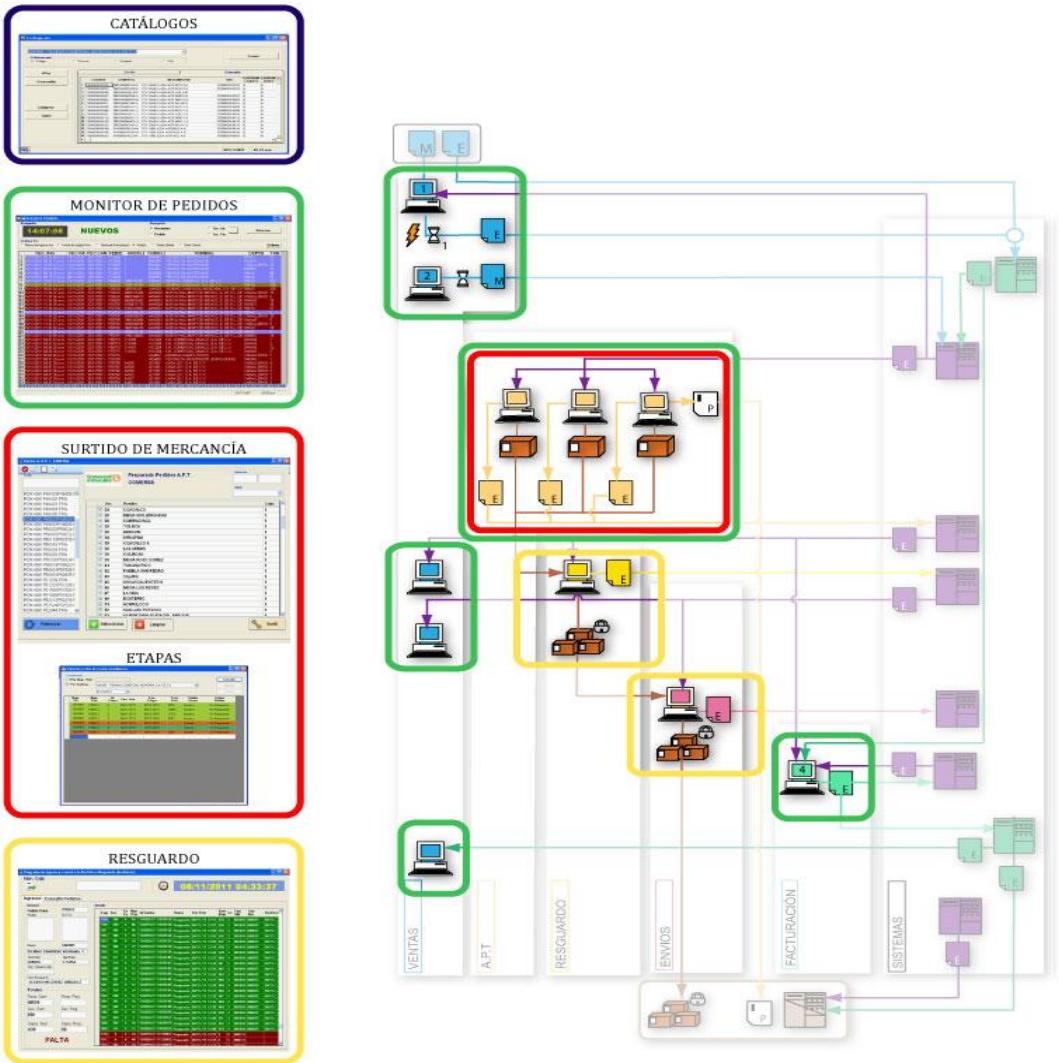


FIG. 52 PRINCIPALES INTERFACES Y SECUENCIA DE UTILIZACIÓN

Catálogos

Para elaborar un pedido, se requiere ingresar algunos valores suplementarios con antelación vinculados con el grupo 1 y 2 de las relaciones (Ver Sección 3.5.2 *Empresa y sus requerimientos*, y *Clientes y sus requerimientos*). Por ejemplo algunos clientes requieren códigos especiales por artículo, para ello existen aplicaciones como la interfaz *Catálogo de Artículos por Cliente* que permite consultar, dar de alta, modificar y eliminar dichos códigos necesarios para la elaboración de documentación e impresión de etiquetas. (Ver *Catálogos* en ilustración anterior).

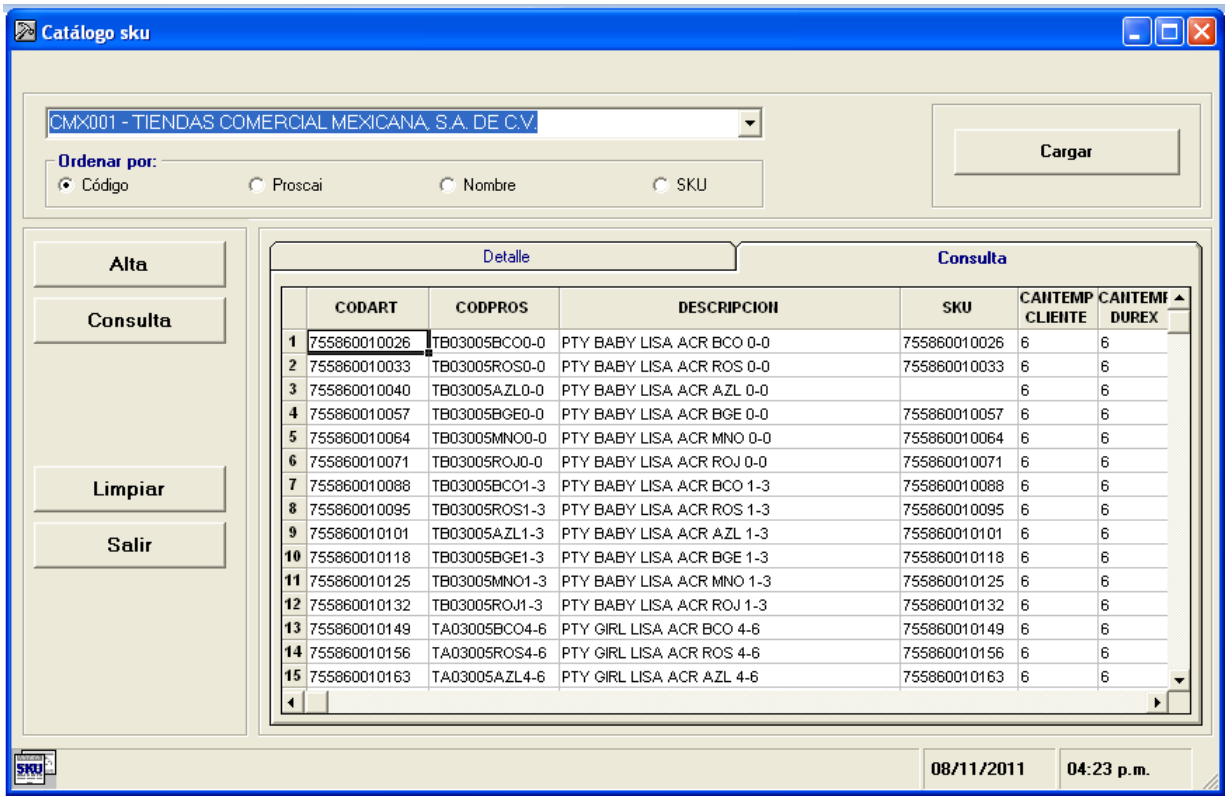


FIG. 53 PROGRAMA ARTÍCULOS ESPECIALES POR CLIENTE

La modificación de dichos catálogos es esporádica, su actualización no representa una carga de trabajo significativa durante el proceso de elaboración de pedidos.

Monitor de Pedidos

Una vez generados, los pedidos pueden ser rastreados mediante una aplicación denominada *Monitor de Pedidos.* Esta aplica permite un seguimiento puntual de su posición, estado y contenido, además de producir documentación necesaria para varias actividades como son las aclaraciones, el preparado, o algún requerimiento especial tanto de la dirección como de un cliente particular. Por ejemplo los reportes de mercancía surtida. (Ver *Monitor de pedidos* Fig. 52).

El monitor está compuesto por dos formas: una *principal* que despliega los encabezados de los pedidos recientes y otra que muestra el *detalle* de un pedido seleccionado.

Forma principal

Cuenta con un grid que despliega por default los encabezados de todos los pedidos generados en los últimos cinco días, actualizándose automáticamente cada cinco minutos. También se pueden hacer

consultas especializadas, por rango de fechas, o por algún pedido en particular. Todas las consultas se pueden ordenar de 6 maneras distintas,

- fecha de ingreso a la base de datos Auxiliares,
- fecha de ingreso a la base de datos ERP,
- fechas de cancelación,
- número de pedido,
- orden de cliente,
- y finalmente por número de cliente.

A cada pedido le corresponde un renglón en el cual se muestran datos globales como las fechas de ingreso y cancelación, el número de pedido, su orden, el número y nombre del cliente, así como la sumatoria de la cantidad de artículos por sucursal. Cada renglón está iluminado por un color indicando el estado del pedido de manera sencilla.

1 Tipo de búsqueda

2 Forma de ordenamiento de los pedidos

FIG. 54 FORMA PRINCIPAL

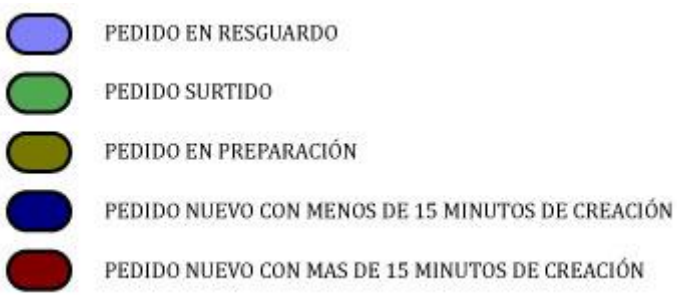


FIG. 55 CÓDIGO DE COLORES EN FORMA PRINCIPAL

Forma Pedido

Esta forma se puede dividir a su vez en dos partes: un *encabezado* y un *detalle*.

En el encabezado se encuentra la información global del pedido dividida en cuatro grupos:

- **General.** Número de cliente, Nombre del cliente, Numero de pedido, Numero del preparador responsable, Nombre del preparador responsable,
- **Totales solicitados**. Total de Artículos solicitados
- **Totales surtidos.** Artículos, piezas, empaques, cajas, sucursales y cantidad de artículos faltantes.
- **Fechas**. Creación, inicio y fin de elaboración

En el detalle del pedido existen siete distintas formas para mostrar la información; cada una permite imprimir un diferente tipo de reporte.

- **General.** Totales por artículos surtidos por pedido.
- **Faltantes.** Totales de artículos faltantes.
- **Sucursales.** Total de artículos surtidos y faltantes por sucursal.
- **Cajas.** Detalle de los artículos surtidos por caja.
- **Etapas.** Totales de artículos surtidos por etapa.
- **Ordenes.** Total de artículos surtidos por Orden (consolidados).
- **Artículos.** Sucursales ordenadas por artículo.

Estas distintas maneras de mostrar un pedido resultan de mucha utilidad para controlar los pedidos en el interior de la empresa, así como para su seguimiento fuera de esta, permitiendo aclaraciones más fidedignas.
Para visualizar de manera sencilla el estado de surtimiento de un artículo, el grid cuenta con un código de colores. El código de colores y las distintas maneras de mostrar el pedido permiten al personal identificar muy fácilmente cualquier inconsistencia que se pudiera presentar.

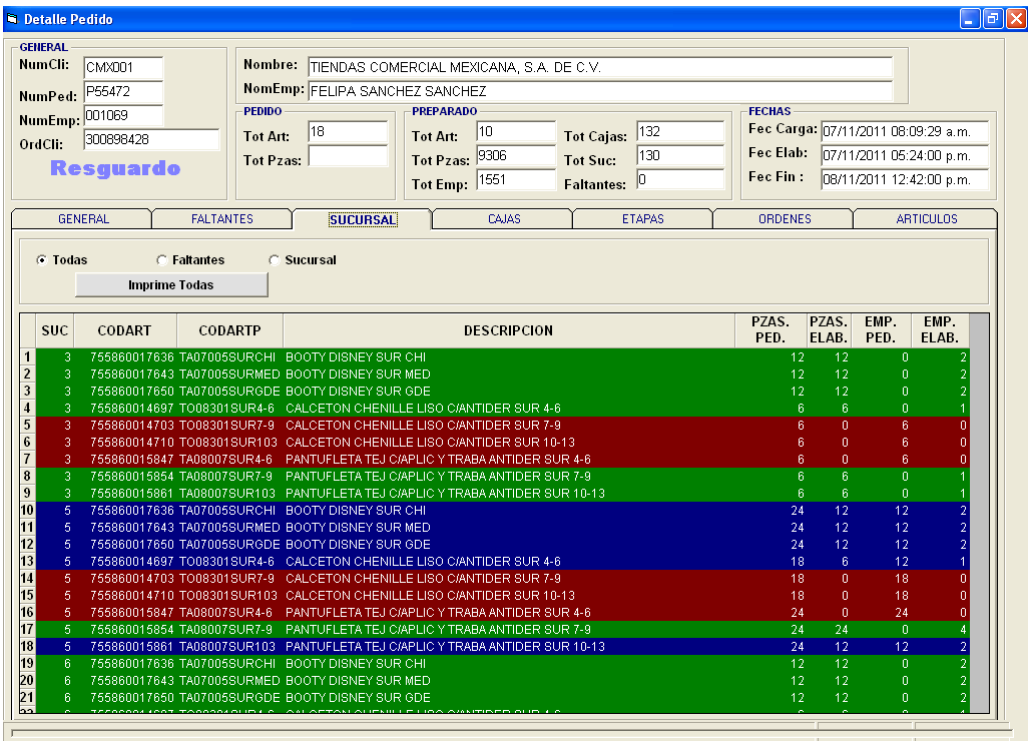


FIG. 56 DETALLE DE UN PEDIDO EN *MONITOR DE PEDIDOS*

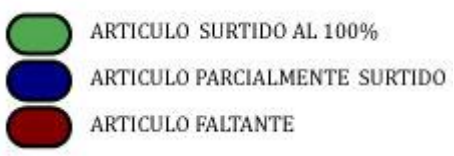


FIG. 57 CÓDIGO DE COLORES EN DETALLE PEDIDOS

Programa de surtido en APT

Para la preparación de un pedido se utiliza una aplicación denominada *Programa de Surtido en A.P.T.* Es la interfaz encargada de validar la entrega de mercancía por caja, y es el programa más complejo de todo el sistema. La sección *3.6.2 Detalle del programa de surtido en APT* explica más a detalle su funcionamiento.

El *Programa de Surtido de A.P.T.* y la interfaz *Etapas* son los que definen el surtido del pedido y crean la estructura que conservará el pedido. (Ver *Surtido de mercancía* Fig. 52).

Etapas

Como mencioné en la sección *3.3 Secuencia propuesta para el modelo comercial y su integración con el nuevo modelo de distribución* existen pedidos especiales -o porque son muy grandes y su preparación requiere de varios días o porque el cliente necesita que la mercancía le sea entregada en fechas diferentes- donde la mercancía se prepara en bloques o "*etapas*" para que la mercancía pueda ser entregada en momentos distintos. Para este tipo de pedidos diseñé un programa de denominado *Etapas* que permite consultar, modificar y agregar -No se pueden eliminar- "*etapas de preparación de mercancía*" a un pedido, -cuando las políticas de entrega de mercancía para un cliente determinado lo permiten-. Esto permite que el *Departamento de Resguardo* tenga un control de la mercancía por bloque o *etapa de entrega*, generando información adicional sobre la distribución de la mercancía.

Cada *etapa* cuenta con un encabezado que contiene el *número de cliente*, el *número del pedido*, el *número de etapa*, la *fecha de vencimiento*, la *fecha de entrega*, las *piezas surtidas*, y el *estatus* de la etapa.

Al crear una *etapa* nueva el programa impide, automáticamente, que se hagan modificaciones a cualquier caja de una *etapa* anterior, de la misma manera que a un pedido preparado. Una etapa que ya ha sido surtida se denomina *etapa cerrada*.

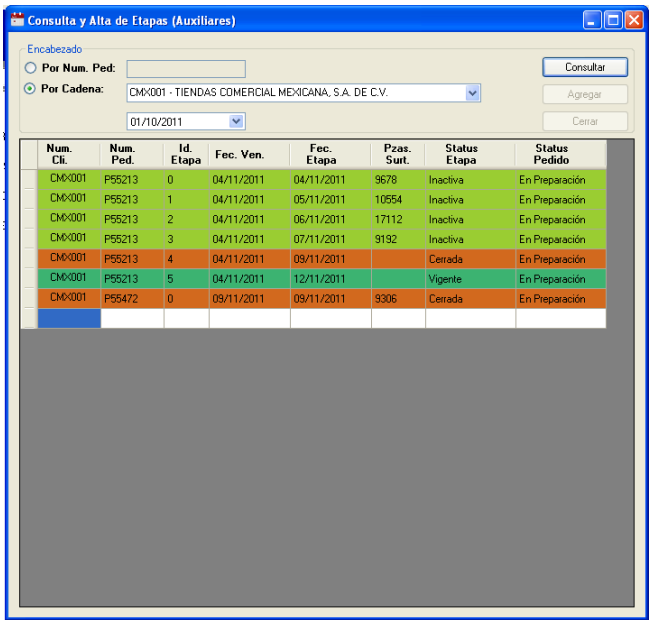


FIG. 58 PROGRAMA *ETAPAS*

Resguardo

Finalmente se utiliza el programa de *Resguardo* de Mercancía el cual permite asignarle a una caja o pedido la sección o departamento donde se encuentra. A diferencia del monitor de pedidos el programa de *resguardo* está diseñado para controlar cajas y pedidos únicamente de un departamento o sección particular. Ambos programas son complementarios y permiten a los empleados involucrados identificar con precisión el estatus y zona donde se localiza un pedido o caja.

Cuenta con dos formas,

- *Forma principal* donde se ingresan las cajas a la sección, y
- *Forma detalle general* donde se despliegan todos los pedidos surtidos.

Forma principal

Esta forma principal cuenta con un campo de texto[81], donde se ingresa, mediante el escáner de códigos de barras, el identificador único de la caja, Este identificador único se encuentra en la etiqueta de preparado de mercancía que se generó con el programa de surtido[82].

[81] Se encuentra en la parte superior izquierda de la forma.
[82] Ver 3.6.2 Detalle del programa de surtido en APT

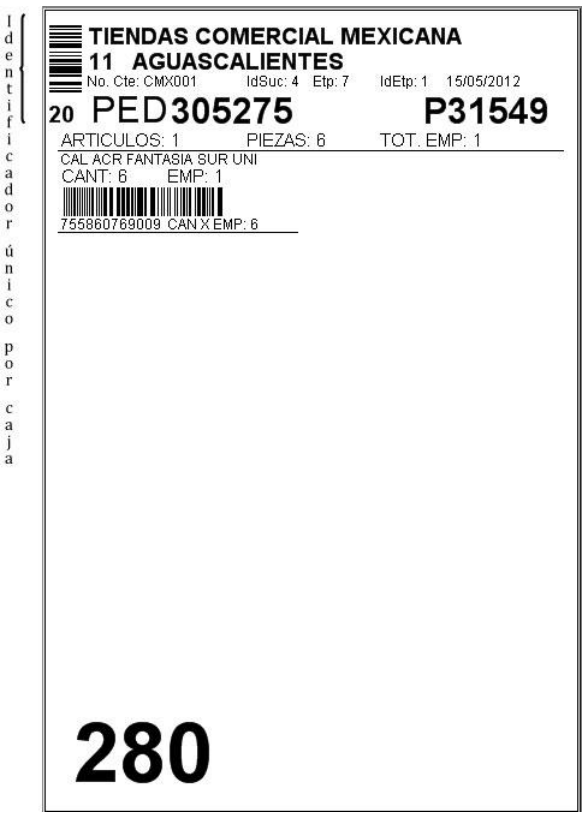


FIG. 59 ETIQUETA CAJA (IDENTIFICADOR ÚNICO)

Si la caja anterior ingresada no pertenece al mismo pedido de la nueva, es necesario que el encargado de la *"Aduana"* ingrese mediante el escáner su clave personal de identificación que se encuentra en su credencial de empleado, lo cual permite vincularlo con la caja nueva.

Al ser escaneada, la caja se registra en la base de datos y del lado izquierdo se despliega información general del pedido, - el número del pedido, el cliente, las fechas de entrega, y los totales de piezas y cajas- indicándole al empleado responsable el número de cajas que se encuentran dentro y fuera del departamento o sección en ese pedido en particular (fig. 60). Del lado derecho se encuentra un grid con el detalle de las cajas. Cada renglón representa una caja del pedido, donde se despliega, -contadores por pedido, por sucursal, y por etapa, el número de piezas en la cajas, un identificador por contenedor, el departamento actual donde se localiza la caja, la fecha en la que fue elaborada, el empleado que la preparó; si la caja se encuentra en el departamento se despliega, la hora en la que entró, y el empleado de la sección que ingreso la caja al nuevo departamento o sección. También cuenta con un código de colores indicando en verde las cajas que han pasado por la aduana, y en rojo cuando no han ingresado.

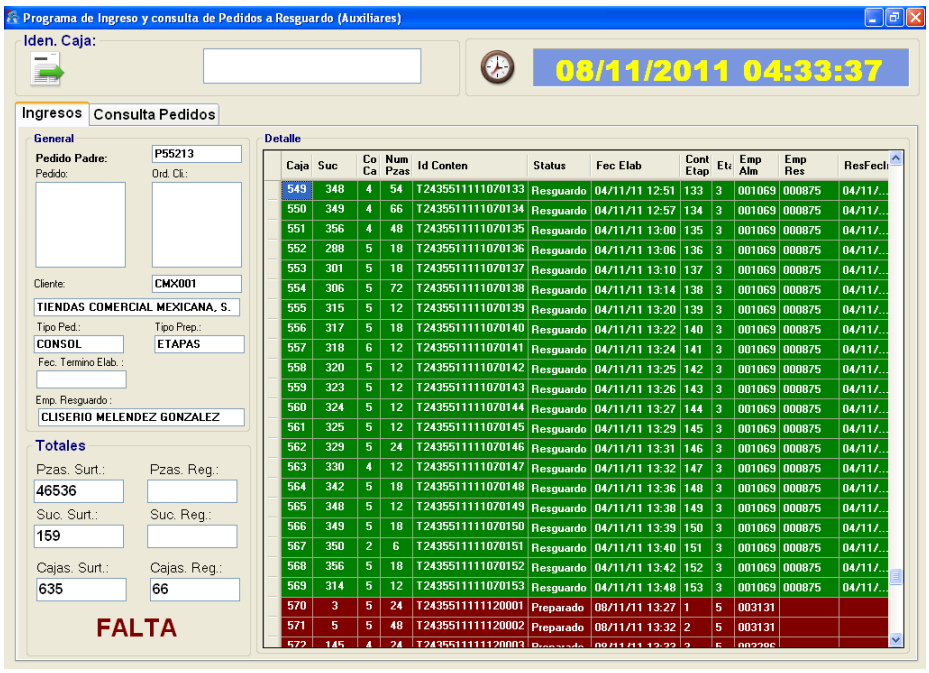


FIG. 60 DETALLE CAJAS EN PROGRAMA DE RESGUARDO

Forma detalle general (fig. 61)

Esta forma permite al encargado de la *"Aduana"* de un determinado departamento identificar los pedidos que se encuentran resguardados en su totalidad (no existe alguna caja fuera del departamento).

Cuenta del lado izquierdo diferentes tipos de búsqueda de pedidos, -día, semana, mes o un rango particular de fechas-. Del lado derecho se encuentra un grid donde se despliegan todos los pedidos elaborado dependiendo del rango seleccionado; indicando el número de pedido, el número del cliente, el total de sucursales surtidas, el número de cajas, la fecha en la cual se comenzó y finalizó de preparar el pedido en APT, el departamento o sección donde se localiza el pedido, el tipo de pedido y preparado, finalmente si el pedido ha ingresado al departamento la fecha y hora de la primera caja que ingreso.

Al igual que los demás programas cuenta con un código de colores, verde cuando el pedido se encuentra completamente en el departamento y rojo cuando aún no han ingresado en su totalidad.

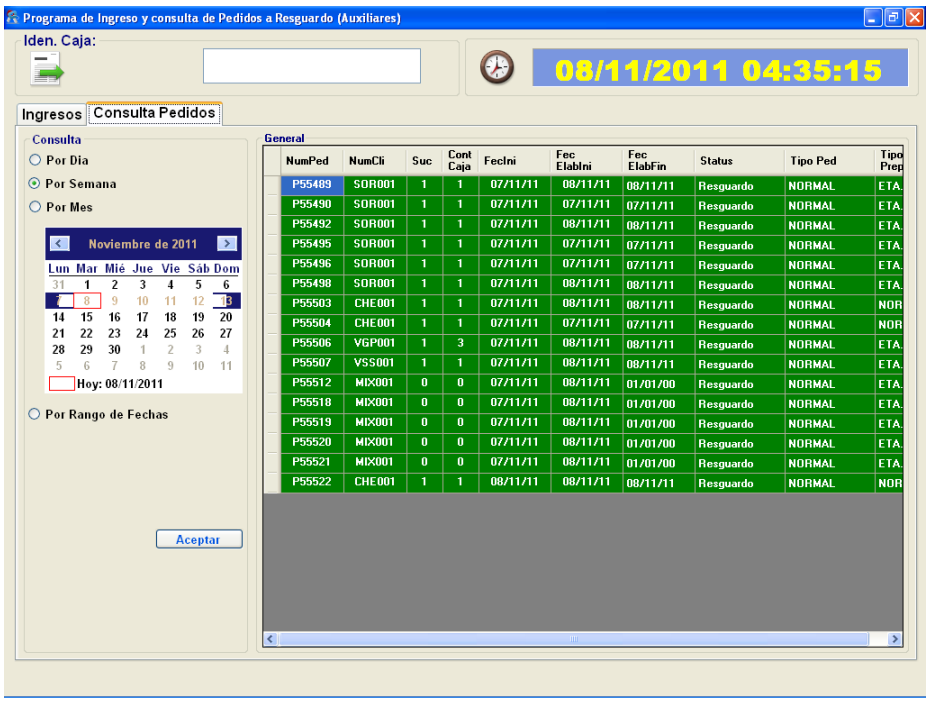


FIG. 61 CONSULTA DE UN PEDIDO POR RANGO DE FECHAS

3.6.2. *DETALLE DEL PROGRAMA DE SURTIDO EN APT*

Esta es la interfaz más importante del sistema. Es un programa que se utiliza para surtir pedidos. Tiene dos formas:

- *Forma principal* donde se seleccionan y editan los pedidos, y otra
- *Forma detalle por sucursal*, en esta forma es donde se surten los artículos que demandan cada una de las sucursales en el pedido.

Descripción de las formas

Forma Principal

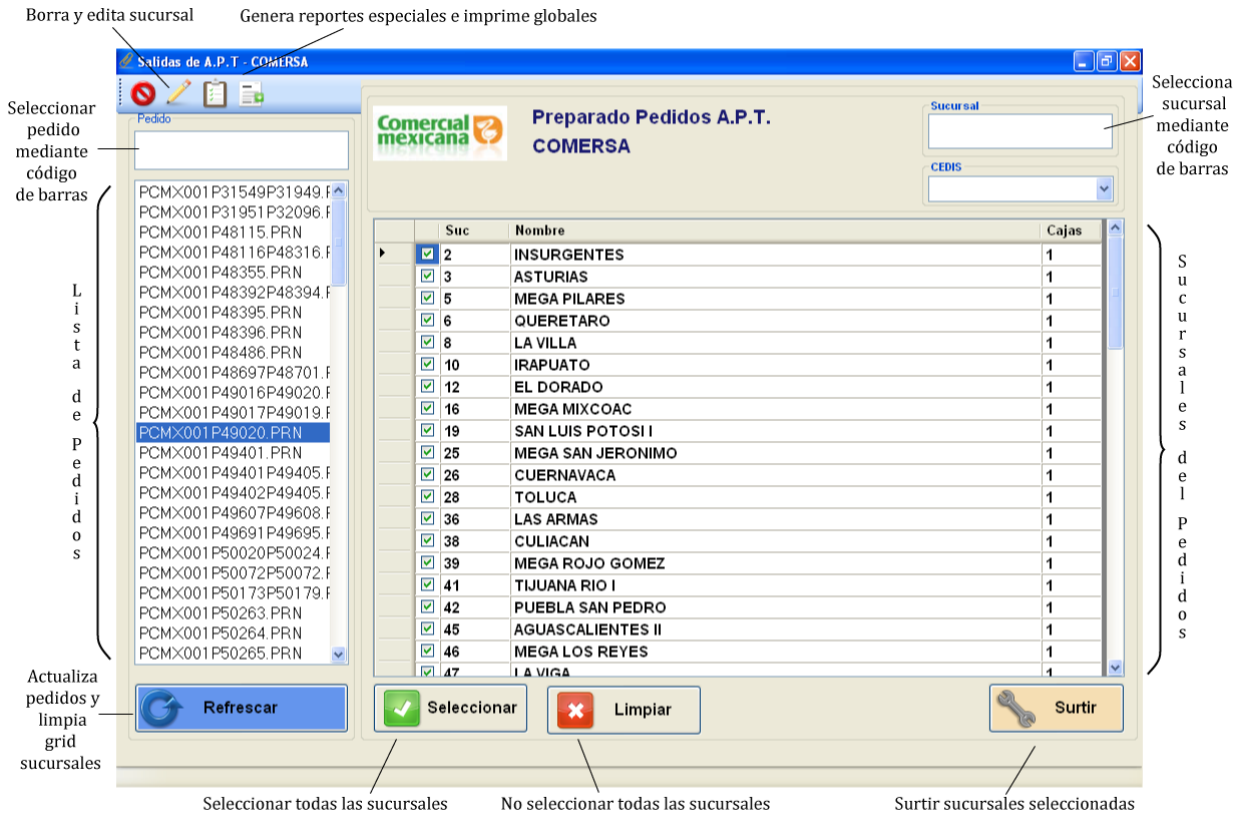


FIG. 62 FORMA PRINCIPAL CON PEDIDO DESPLEGADO

En la parte superior izquierda se encuentra un *menú de procesos especiales*, donde se pueden borrar o editar sucursales, generar reportes particulares[83] e imprimir reportes globales.[84] Abajo del menú se encuentra un campo de *captura de pedido* donde se ingresa el pedido a surtir o visualizar mediante un código de barras. A la derecha del menú y del campo, se encuentran el *logo* y el *nombre del cliente* el cual se está preparando. Del lado superior derecho se encuentra un campo de *captura de sucursal* donde se puede introducir, mediante un código de barras, la sucursal que se va a surtir o visualizar. Bajo el campo de *captura de sucursal* se encuentra un *combo de centros de distribución* que despliega los centros de distribución disponibles para el pedido[85].

[83] Reportes en papel o electrónicos que solicitan clientes de manera particular.
[84] Se ve más a detalle en la sección **procesos especiales.**
[85] Dicho campo se utiliza en muy pocas ocasiones.

En la parte intermedia de la forma, del lado izquierdo, se encuentra una *lista de pedidos* con todos los pedidos disponibles. A su derecha se encuentra un *grid de sucursales* que despliega todas las sucursales de un pedido seleccionado.

En la parte inferior de la forma se encuentran: un botón de *Refrescar* que actualiza pedidos y limpia el grid de sucursales; los botones *Seleccionar*, donde se eligen todas las sucursales de un pedido; el botón *Limpiar* que descarta todas las sucursales; y, el botón de *Surtir*, el despliega uno por una todas las sucursales seleccionadas en la forma *Detalle por Sucursal*.

Forma detalle por sucursal

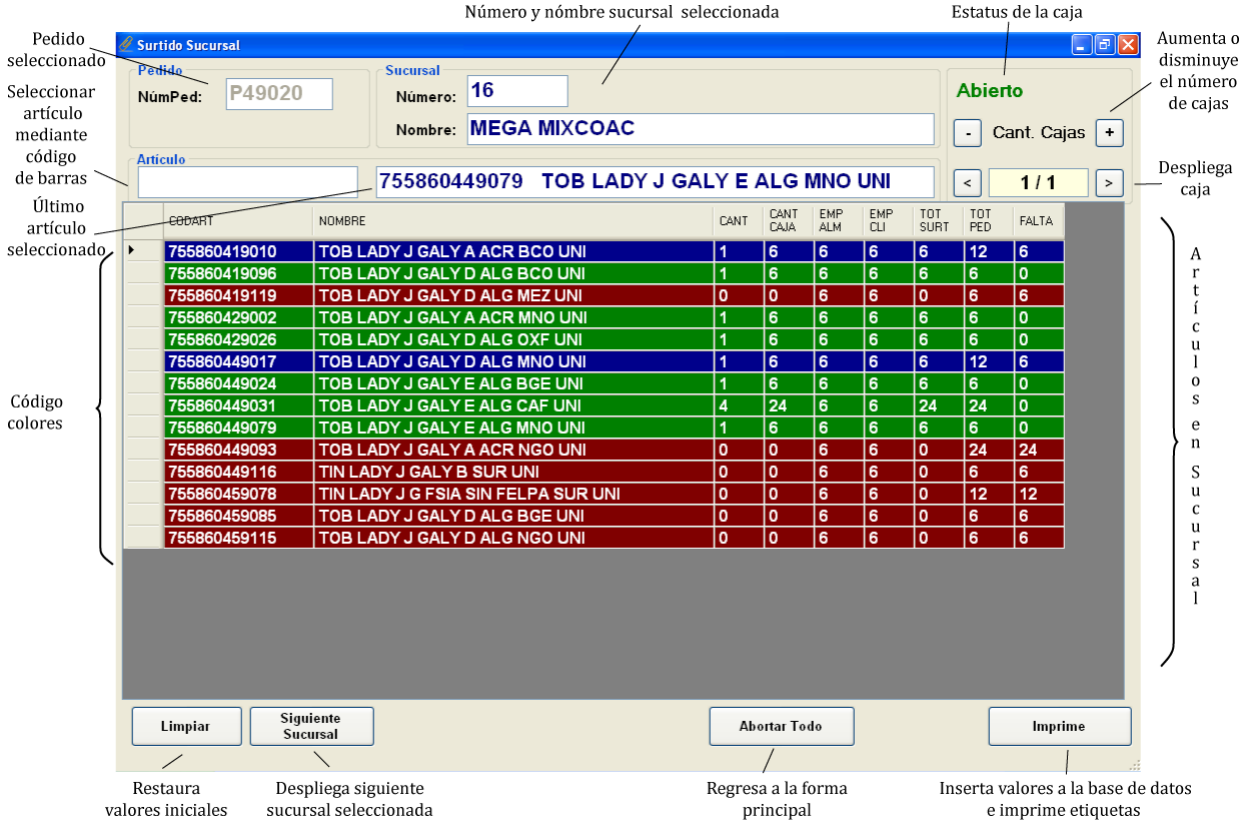


FIG. 63 FORMA DETALLE POR SUCURSAL

En la parte superior se encuentra, de izquierda a derecha, el *pedido seleccionado* y un campo donde se *seleccionan artículos mediante un código de barras*, posteriormente *número y nombre de la sucursal*, un campo donde se despliega el *último artículo seleccionado y,* finalmente, el *estatus de la caja,* que indica si el pedido se puede preparar o si ya fue elaborado; botones para *aumentar o disminuir el número de cajas* a surtir y, botones para desplegar las diferentes cajas surtidas.

En la parte intermedia de la forma se despliegan los artículos requeridos por la sucursal indicando las cantidades por paquetes y piezas (pedidas, faltantes y surtidas), así como las cantidades de piezas que contiene cada paquete.

Finalmente en la parte inferior se encuentran el botón de *Limpiar,* que restaura los valores iniciales; *Siguiente sucursal,* despliega en la misma forma la siguiente sucursal por surtir, guardando o no las cantidades modificadas dependiendo de los deseos del empacador; *Abortar Todo* que regresa a la forma principal sin desplegar todas las sucursales seleccionadas restantes; *Imprimir* que inserta valores del

número de caja, con sus respectivos artículos, a la base de datos e imprime las etiquetas de la caja correspondiente.

Seleccionar pedidos para ser surtidos o visualizados

Dependiendo del cliente, al iniciar el programa se hace una consulta a una carpeta que contiene los pedidos exportados por el programa ERP y se despliegan en la *lista de pedidos* del lado izquierdo de la pantalla.

Se selecciona el pedido ya sea buscándolo en la *lista de pedidos,* o escaneando el número de pedido que se obtiene de una solicitud de ventas mediante un código de barras (Ver *Seleccionar pedido mediante código de barras* en Fig. 62*)*.

Al ser seleccionado el pedido, se cargan los valores solicitados (que se obtienen del archivo generado por el ERP), así como los valores surtidos anteriormente (que se obtienen directamente de la base de datos *Auxiliares*).[86]

Todos los valores del pedido son almacenados en cada una de las computadoras personales distribuidas por mesa de trabajo; las operaciones se realizan individualmente en cada una de las computadoras personales, evitando hacer consultas y operaciones innecesarias en el servidor o la base de datos.

Una vez cargados todos los valores en las computadoras personales, se despliegan las sucursales a surtir del lado derecho (Ver *Sucursales del pedido* en Fig. 62*),* indicando con color blanco si no han sido surtidas y en color azul las que ya lo han sido.

Al igual que para la selección del pedido existen dos formas de seleccionar sucursales, tanto para ser consultadas como para surtirlas: una es seleccionado las sucursales en la grid *Sucursales del pedido* o escaneando un código de barras en la solicitud de ventas de surtido (ver *Seleccionar sucursal mediante código de barras* en Fig. 62*)*.

Surtido del pedido

Para surtir un pedido es necesario escogerlo como se indica en la sección anterior (*Seleccionar pedidos para ser surtidos o visualizados*). En cualquiera de las dos formas en las que se elija una sucursal, se despliega una pantalla en la que el empleado tiene que ingresar una clave personal a través de un código de barras que se encuentra en su credencial.[87]

[86] Insertando todos los valores necesarios para la preparación del pedido al inicio de la consulta con la finalidad de evitar bloqueos.
[87] Es responsabilidad de cada empleado cuidar dicha credencial, así como su número confidencial. Es una firma electrónica que identifica al empleado con el pedido y las sucursales preparadas.

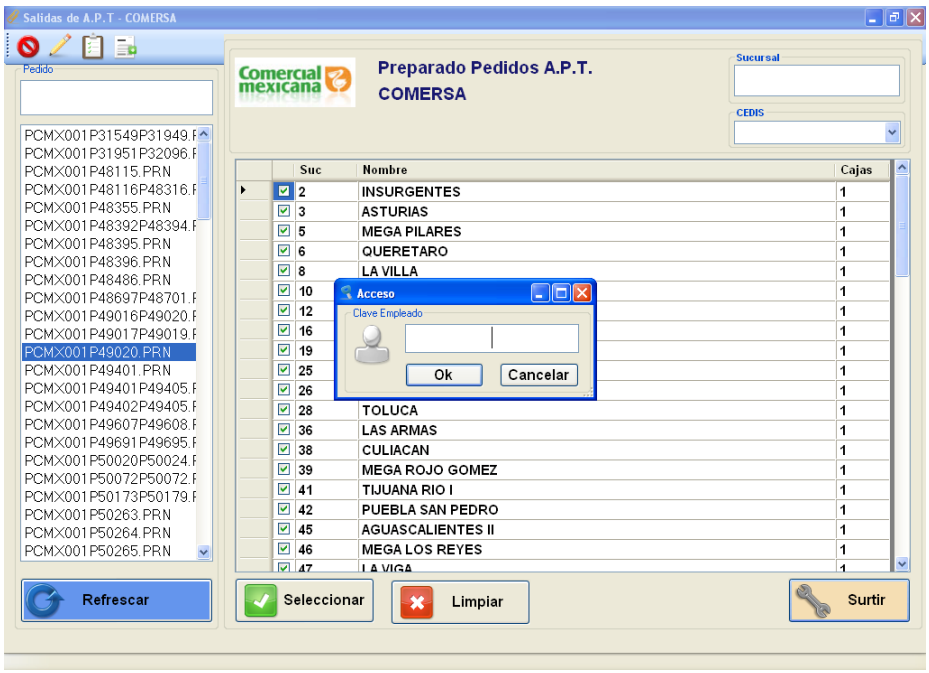


FIG. 64 SOLICITUD DE CLAVE PARA PREPARAR EL PEDIDO

Una vez validada la clave del empleado se despliega la segunda forma con los artículos solicitados por sucursal. Si se hizo una selección múltiple, cada vez que se termina de preparar una sucursal, se actualizan los datos de la segunda forma con los valores de la sucursal siguiente hasta que se terminan de elaborar todas o se presiona el botón *Abortar todo*

En la segunda forma del lado superior izquierdo se encuentra un campo de datos que permite registrar el código del artículo surtido gracias a un código de barras que se encuentra en cada producto (ver Fig. 63). Al ser detectado el código se actualizan las cantidades surtidas y el renglón donde se muestra el artículo cambian de color. Este código de colores permite a un empacador darse cuenta fácilmente de los faltantes por surtir, indicándole los que artículos no han sido surtidos.

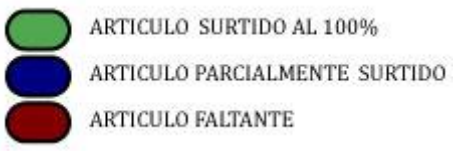


FIG. 65 CÓDIGO DE COLORES PARA SURTIDO DE MERCANCIA

Se repite el proceso hasta que la caja se encuentre abarrotada o se termine de surtir la sucursal; si una caja se encuentra llena y si no se ha surtido toda la mercancía, es necesario indicarle al sistema que se requiere una caja adicional para proseguir. Para lo cual es necesario presionar el botón con signo + que se encuentra en la parte superior derecha de la forma, (*ver* Fig. 63). Al agregar una caja se insertan los valores a la base de datos de la caja empacada, se imprime su etiqueta correspondiente y ya no se le puede hacer modificaciones a los datos relacionados con la caja en la base. A continuación se muestra una la etiqueta que comúnmente se pega a las cajas surtidas.

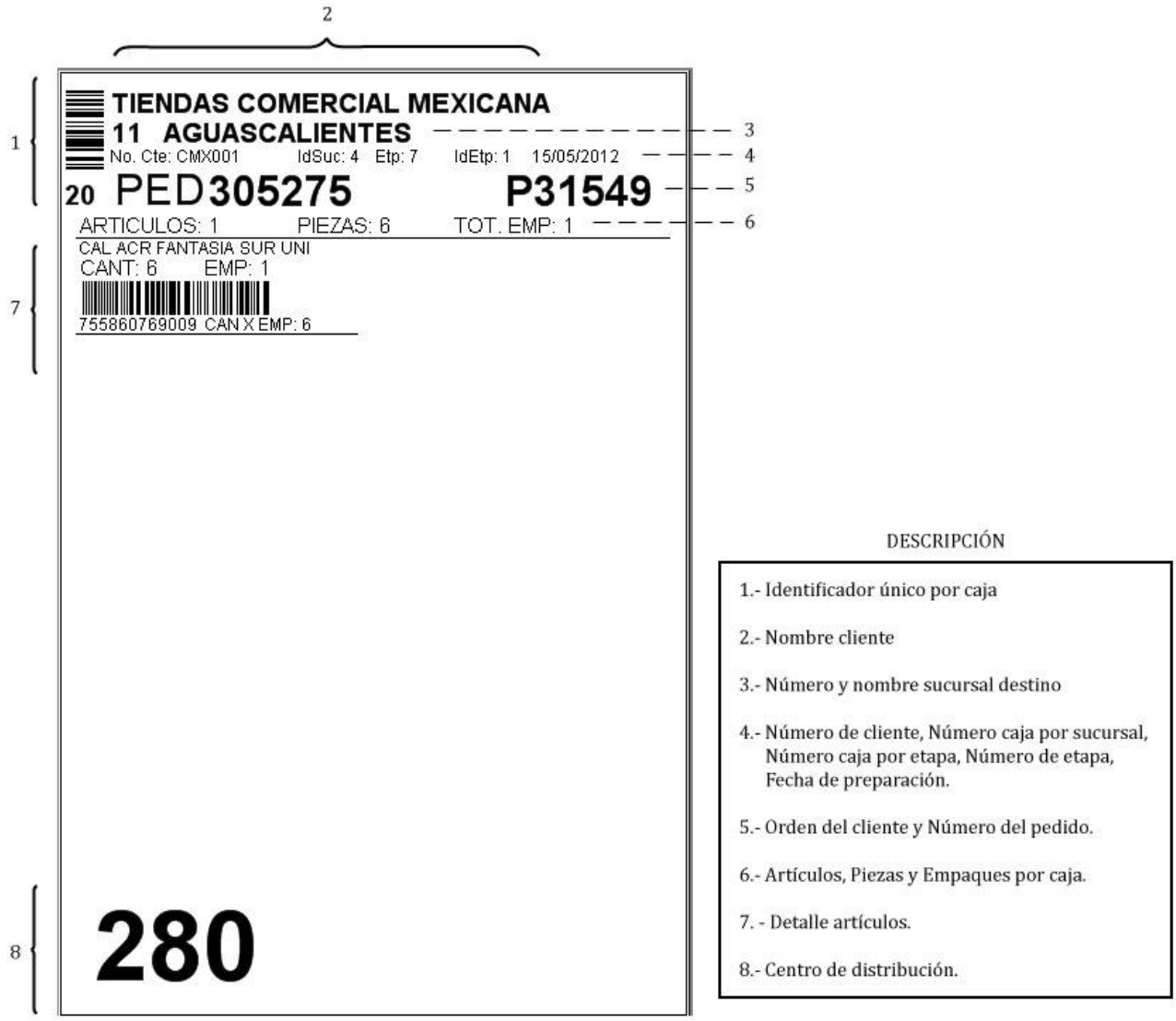


FIG. 66 DESCRIPCIÓN DE ETIQUETA GENÉRICA

FIG. 67 ETIQUETA EN CAJA (WALMART)

Al terminar de surtir la mercancía, mediante el escáner o presionando el botón *imprime,* se insertan los valores correspondientes a la base de datos, se genera la etiqueta de la última caja, y se actualiza la forma con los datos de la nueva sucursal a surtir. Este proceso optimiza al máximo la inserción de datos a la base así como los viajes ida y vuelta al servidor. Otra ventaja es que al haber una falla en la comunicación con el servidor al insertar los datos, la interfaz se bloquea y la etiqueta no se imprime, ya que es un proceso posterior. Haciendo de la etiqueta una prueba que hubo una inserción correcta a la base de datos.

Cuando se hace una selección múltiple de sucursales, al terminar de surtirlas, el programa verifica que no haya empleados trabajando en el pedido y pregunta al último operario si desea cerrar el pedido e imprimir las guía de embarque.

Si el usuario escoge imprimir una guía de embarque, se ejecutan los procedimientos almacenados de cierre del pedido, y se imprimen los globales. En este momento se verifica si el cliente tiene algún requerimiento especial y se generan los reportes correspondientes. El pedido se cierra y no puede ser modificado.[88]

Pedidos con sucursales surtidas

Cuando se selecciona pedidos en la forma principal con sucursales surtidas[89] estas aparecen en color azul, indicando que cuentan con cajas asociadas (Ver Fig. 68). Si el pedido no ha salido del almacén y se encuentra vigente, estas sucursales se pueden seleccionar para ser visualizadas, reimprimir etiquetas, o en el caso de preparación de pedidos por etapas agregar mercancía adicional. Cuando se selecciona una sucursal se despliega la *forma de detalle por sucursal* mostrando el total de artículos surtidos por caja de color azul claro (Ver Fig. 69).

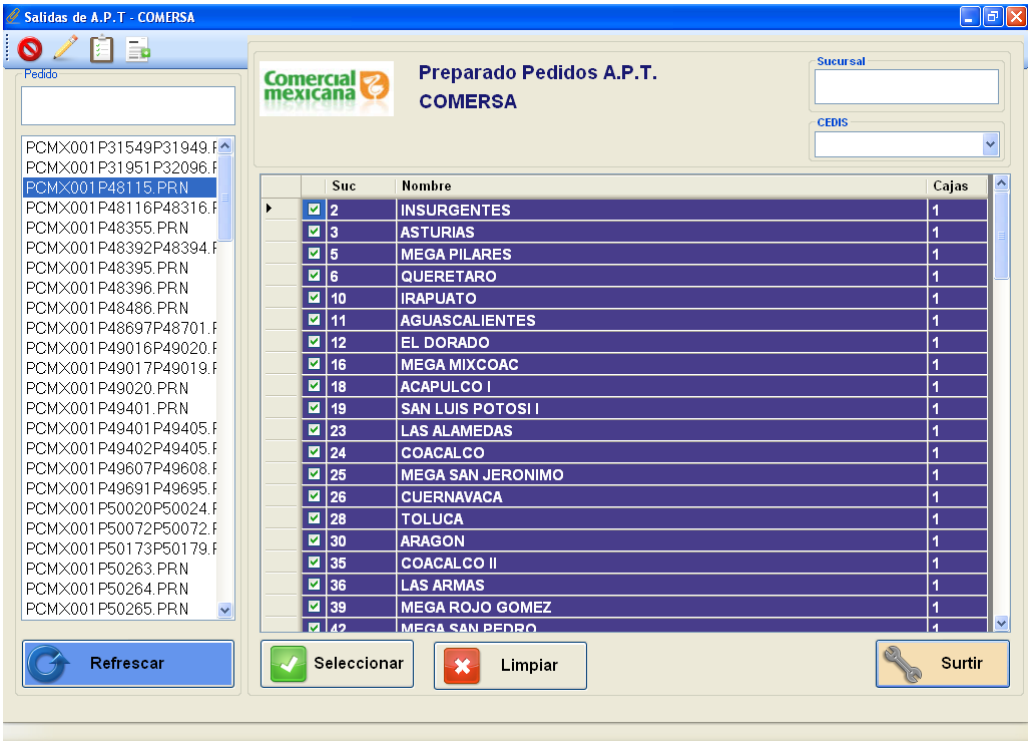


FIG. 68 PEDIDO SURTIDO FORMA PRINCIPAL

[88] Ver Anexo X Creación de la base de datos

[89] Ver Seleccionar pedidos para ser surtidos o visualizados

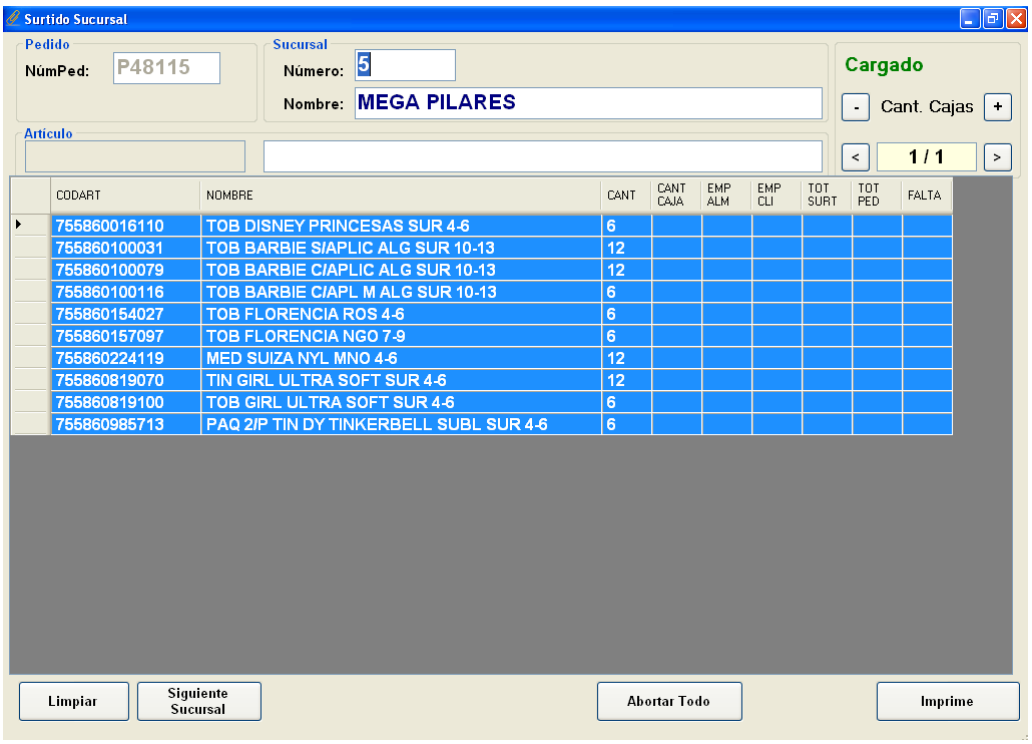


FIG. 69 DETALLE SUCURSAL SURTIDA

Para poder añadir mercancía suplementaria a una sucursal surtida, es necesario agregar una caja adicional. Cuando el tipo de pedido lo permite (pedido por etapas), se aumenta una caja presionando el botón + en la parte superior derecha de la forma. Una vez que se ha agregado la caja se surte de la misma manera que una sucursal que no ha sido surtida anteriormente.

Procesos especiales

Son cinco procesos que solamente el encargado de Almacén se encuentra habilitado para ejecutar. Se realizan una vez que un pedido se ha elaborado y son los siguientes:

- *Borrar Sucursal:* Permite la eliminación de todas las cajas que han sido surtidas en las sucursales seleccionadas. La sucursal solamente puede ser eliminada si el pedido no se encuentra bloqueado y todas sus cajas se encuentran en APT.
- *Editar Sucursal:* Permite el re empaquetado de una ó más cajas surtidas en la sucursal seleccionada. Solamente se puede editar si el pedido no se encuentra bloqueado y la caja se encuentra en el APT. A diferencia de borrar sucursal, se puede editar una caja, si esta no ha salido del almacén aunque las demás cajas de la sucursal si lo hayan hecho y el identificador global de la caja editada se conserva.
- *Imprimir documentación global y cerrar etapas:* Permite la reimpresión de documentación global así como cerrar etapas que se encuentran abiertas. Esta documentación la utilizan tanto los choferes, así como el personal que recibe la mercancía en los centros de distribución.
- *Generar reportes especiales:* Permite la reimpresión, o la reelaboración electrónica de reportes especiales dependiendo del tipo de cliente desplegado.

Características y requerimientos generales del programa de surtido

Similitudes y diferencias en la elaboración de un pedido y solución de la interfaz.

El surtido de mercancía se puede dividir en dos; una parte genérica para todos los pedidos y otra particular donde difieren los requerimientos del pedido dependiendo del cliente. Las dos partes se manejan los siguientes procesos:

Genérica:

- Visualización de pedidos del programa EPR,
- desplegar y preparar la mercancía,
- documentación general para cada pedido,
- inserción de valores a la base de datos.

Particular por cliente:

- Etiquetas especiales para clientes particulares.
- Documentación tanto electrónica como en papel para clientes particulares.

Base de datos

Existen dos características relacionadas con la base de datos:

- El acceso y las claves de la base de datos deben estar protegidos.
- La migración de una base de datos se debe ejecutar en todas las computadoras personales que utilizan la interfaz de una manera casi inmediata y transparente. (Se planea hacer cambios de base de datos de manera sistemática)

Disminución de errores en el preparado

Existen principalmente dos requerimientos referentes a la disminución de errores.

- El programa debe ser capaz de detectar errores en la consistencia de los datos como en la preparación de los pedidos. Generando diferentes mecanismos que prevengan el error humano.
- Fácil de usar. Simplificando al máximo el proceso de preparado, permitiendo a un operario con conocimientos mínimos de computación, detectar errores.

Programa multiusuario

La interfaz debe permitir que varios empleados trabajen simultáneamente en distintas sucursales de un pedido al mismo pedido. Por políticas de la empresa dos o más empleados no pueden trabajar una sucursal de un mismo pedido de manera paralela.

Sin retardos

Al preparar los pedidos no debe de haber retardos en la lectura de códigos de barras cuando se escanean artículos.

Manejo de pedidos consolidados y por etapas

El programa debe de ser capaz de distinguir la forma de elaboración de un pedido dependiendo de las características especiales de cada cliente en el momento de la preparación del mismo.

Servicios Web

Algunos clientes solicitan *avisos anticipados de embarque* mediante servicios Web, esto permite asignar citas de entrega y el pago anticipado de ciertos pedidos. El programa debe de estar diseñado para manejar éstos servicios Web de una manera dinámica.

Seguridad en la preparación de pedidos

Existen varios elementos que se deben tomar en cuenta para poder preparar un pedido, por ejemplo:

- La preparación de los pedidos solamente puede prepararse dentro de un rango de tiempo especificado.
- Una vez que el pedido sale del Almacén ya no puede ser modificado.
- El acceso a los pedidos debe estar controlado, solamente empleados autorizados pueden deben tener acceso a los pedidos.
- Se requiere un registro único por caja que permita rastrearla en cualquier parte del proceso en que se encuentre.

Asignación automática de pedidos

Los pedidos una vez preparados deben de poder ser asignado[90] en el ERP, simplificando el proceso de facturación.

Fácilmente modificable

El programa debe ser lo suficientemente flexible para poder satisfacer las demandas cambiantes de los clientes, sin que sea necesario hacer modificaciones significativas a la estructura del programa y en tiempos relativamente cortos.

Soluciones a algunos de los requerimientos generales de la aplicación.

Similitud y diferencias en la elaboración de un pedido.

Operar diferentes clientes mediante un mismo programa, presenta algunas ventajas como permitir un mayor control sobre la interfaz que se está desarrollando, evitar duplicidad de código, facilita la detección de errores, entre otras. Para lo cual se requiere que el programa pueda manejar la parte *genérica* y las *partes específicas de cada cliente* de una manera relativamente sencilla. La interfaz debe ser capaz de distinguir el cliente y debe de estar diseñado para que se le realicen modificaciones periódicas de manera relativamente sencilla como resultado de las demandas de los mismos. *****

Para insertar los parámetros iniciales a la interfaz utilicé un segundo programa ejecutable que a su vez llama al programa de surtido enviándole como parámetro inicial, el cliente. A continuación se muestra el código del programa ejecutable que manda llamar al *programa de surtido de almacén PrepAlmConXEtaIndRes.exe,* asignándole el parámetro *CHEDRAUI.*

```
Sub Main()
Shell("F:\PROSCAI\EJECUTABLES\ENVIOS\PrepAlmConXEtaIndRes.exe CHEDRAHUI", vbMaximizedFocus)
End Sub
```

Esta forma de manejar la información cuenta con una serie de ventajas adicionales. El personal visualiza programas individuales fácilmente identificables con distintos iconos. Además el programa ejecutable de surtido de almacén se encuentra protegido en otra carpeta la cual no se puede acceder y se desconoce su dirección.

[90] La asignación en un proceso anterior a la facturación en el ERP. Donde se insertan las cantidades surtidas al pedido por sucursal.

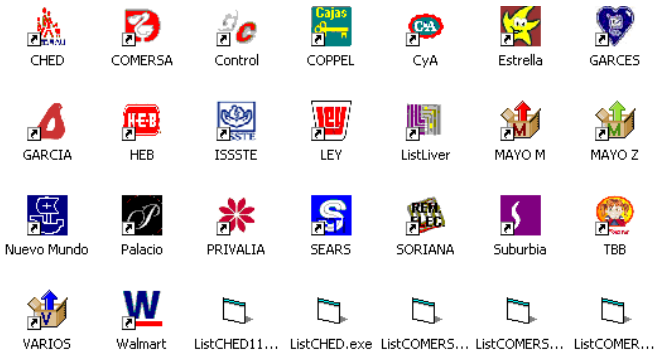


FIG. 70 DIFERENTES VISTAS DE PROGRAMA DE SURTIDO

Al iniciar la ejecución de la interfaz, se ejecutan una serie de funciones iniciales basadas en *Linq*[91] *y XML*, responsables de importar parámetros iniciales. Dentro de esas funciones se encuentra la función *InicialesCadena* siendo la responsable de importar los parámetros de la cadena.

Esta función utiliza un archivo *XML* con los parámetros requeridos, y los importa a una clase llamada *StrucCadena* donde se almacenan. El utilizar XML para manejar los parámetros de los clientes hace que su modificación sea mucho más flexible, a diferencia de enviar todos los parámetros directamente en el segundo programa. A continuación se muestra la función *InicialesCadena* y un fragmento del *XML* con la descripción de sus parámetros.

Función InicialesCadena:

```
Public Sub InicialesCadena(ByVal NomCad As String, ByRef Cad As StrucCadena, ByVal Carpeta As
String) 'Subrutina encargada de ingresar valores del Cliente y/o Cadena
        Try
        Cad.Nombre = NomCad
        ' Usando LINQ
        Dim CadXML As XElement = XElement.Load(Carpeta & NomCadXml)
        Dim BNodo As XElement
            Dim query = From r As XElement In CadXML...<Cadena>
                        Where r.@<name> = NomCad
            BNodo = query.FirstOrDefault()
            Cad.Logo = BNodo.<Logo>.Value
            Cad.TipoArch = BNodo.<TipoArchivos>.Value
            Cad.ArtXCaja = BNodo.<MaxArtCaja>.Value
            Cad.RepTipo = BNodo.<RepTipo>.Value
            Cad.RepEsp = BNodo.<RepEsp>.Value
            Cad.CedisSel = BNodo.<CedisSel>.Value
            Cad.Depart = BNodo.<Depart>.Value

        Catch ex As Exception
            If ex.GetType.ToString = "System.NullReferenceException" Then
                MsgBox("La Cadena: " & NomCad & ", no está dada de alta." & vbNewLine &
                "Informar Sistemas", MsgBoxStyle.Critical, "InicialesCadena")
                Environment.Exit(0)
            Else
                MsgBox(ex.Message)
            End If
        End Try
```

[91] Language Integrated Query (LINQ) es un proyecto de Microsoft que agrega consultas nativas semejantes a las de SQL a los lenguajes de la plataforma .NET, inicialmente a los lenguajes Visual Basic .NET y C#.
LINQ define *operadores de consulta estándar* que permiten a lenguajes habilitados con LINQ filtrar, enumerar y crear proyecciones de varios tipos de colecciones usando la misma sintaxis. Tales colecciones pueden incluir vectores (arrays), clases enumerables, XML, conjuntos de datos desde bases de datos relacionales y orígenes de datos de terceros.
El objetivo de crear LINQ es permitir que todo el código hecho en Visual Studio (incluidas las llamadas a bases de datos, datasets, XMLs) sean también orientados a objetos. Antes de LINQ, la manipulación de datos externos tenía un concepto más estructurado que orientado a objetos. Además LINQ trata de facilitar y estandarizar el acceso a dichos objetos.

```
End Sub
```

Fragmento del XML:

```
<Cadena name="COMERSA" ID="CMX">
 <Logo>comer2.jpg</Logo>
 <TipoArchivos>PCMX*.prn</TipoArchivos>
 <MaxArtCaja>0</MaxArtCaja>
 <RepTipo>CMX</RepTipo>
 <RepEsp>TRUE</RepEsp>
 <CedisSel>TRUE</CedisSel>
 <Depart>TRUE</Depart>
 <TipoPed>2</TipoPed>
 <TipoPrep>2</TipoPrep>
 </Cadena>
```

<Logo>:	Nombre de su logo. Se utiliza para desplegarlo en la parte superior de la forma principal.
<TipoArchivos>:	Tipo de archivos PRN de la cadena, donde se encuentra la información de los artículos solicitados.
<MaxArtCaja>:	Máximo número de artículos por caja. Algunas cadenas tienen un tope de artículos que se pueden enviar en una sola caja, 0 indica que el número es indefinido.
<RepTipo>:	Identificador para reportes especiales.
<RepEsp>:	Bandera que indica si requiere reportes especiales.
<CedisSel>:	Bandera que indica si la mercancía se envía a un centro de distribución.
<Depart>:	Bandera que indica si los pedidos se envían por departamento.
<TipoPed>:	Identificador de tipo de pedido. Los tipos de pedidos puedes ser individuales ó consolidados.
<TipoPrep>:	Identificador de tipo de preparado. En un solo envío o por etapas.

Estos parámetros permiten que el programa se pueda adaptar a los requerimientos de un cliente particular y que el preparador pueda identificar fácilmente al cliente que está trabajando.

A continuación se muestra el mismo programa con parámetros distintos.

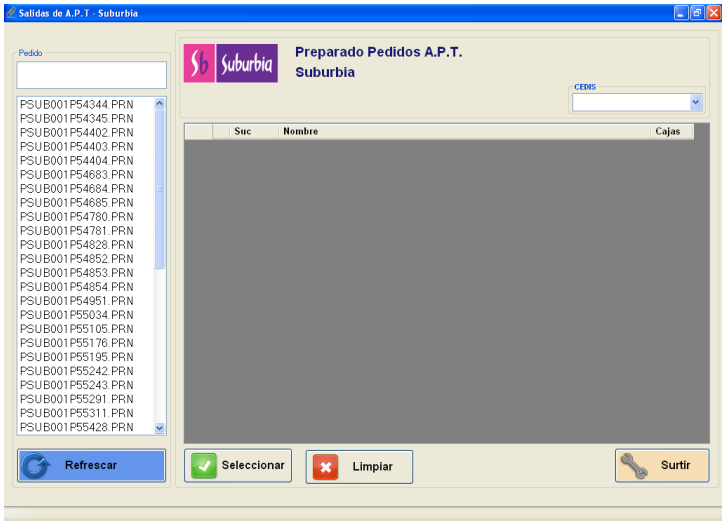


FIG. 71 VISTA SUBURBIA

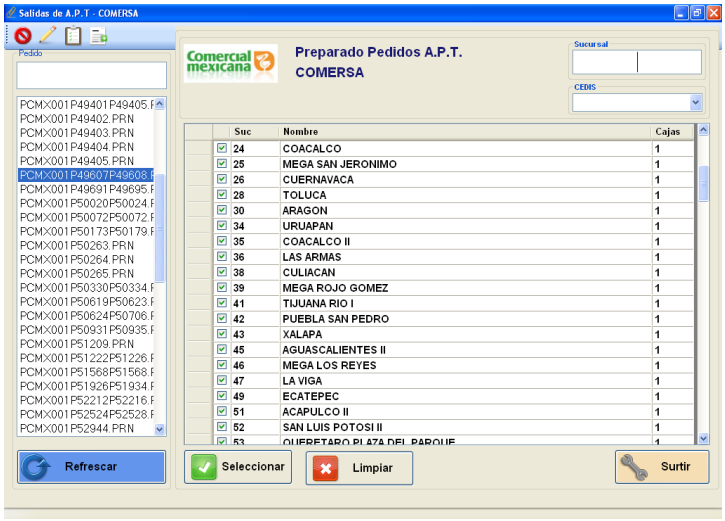


FIG. 72 VISTA COMERCIAL MEXICANA

Base de datos

Acceso a parámetros de conexión protegidos.

Para construir la cadena de conexión necesaria para establecer una conexión con la base, se requieren una serie de parámetros. Dichos parámetros se obtienen mediante *LINQ y XML* de manera similar a la forma como se extraen en la función *InicialesCadena.* Pero a diferencia de los parámetros iniciales de una cadena, los parámetros de conexión deben ser protegidos para evitar que una persona no autorizada pueda ingresar a la base. Por lo tanto la función que extrae los parámetros de la base de datos *InicialesBase* cuenta con un algoritmo de encriptado que sirve, para proteger la integridad de la base de datos. Este algoritmo se denomina *TDES*[92], el cual encripta al usuario y la contraseña que se requiere para ingresar a la base. Una vez des encriptado el usuario y contraseña se puede genera la cadena de conexión que se utiliza para establecer las conexiones a la base de datos.

A continuación se muestra el Archivo XML que se utiliza para importar los parámetros de conexión a la base de datos y la función *InicialesBase* donde se muestra el proceso de des encriptado y la generación de la cadena de conexión.

```
<?xml version="1.0" encoding="utf-8" ?>
<CONEXION>
        <SERVIDOR>49PROG</SERVIDOR>
        <BASE>Auxiliares</BASE>
        <USUARIO>/V+O26erTLw=</USUARIO>
        <PWR>Kcd6U/i6dKRLTwE+uJnd9Q==</PWR>
</CONEXION>
```

[92] **Triple DES** algoritmo de triple cifrado. También es conocido como TDES o 3DES, fue desarrollado por IBM en 1998.La variante más simple del Triple DES funciona de la siguiente manera:

$C = E_{DES}^{k_3}\left(D_{DES}^{k_2}\left(E_{DES}^{k_1}(M)\right)\right)$ Donde M es el mensaje a cifrar y k_1, k_2 y k_3 las respectivas claves DES. En la variante 3TDES las tres claves son diferentes; en la variante 2TDES, la primera y tercera clave son iguales.

```
Public Sub InicialesBase(ByRef ConIni As RegistroInicio)
        'Subrutina encargada ingresar valores de los directorios iniciales mediante XML.

        Dim Dec As New Simple3Des("oax1")

        ' Usando LINQ
        Dim ConnXML As XElement = XElement.Load(Ini.Base & "\INI\Conn.xml")
        With ConIni

            .Servidor = ConnXML.<SERVIDOR>.Value
            .BaseDeDatos = ConnXML.<BASE>.Value
            .Usuario = Dec.DecryptData(ConnXML.<USUARIO>.Value)
            .Password = Dec.DecryptData(ConnXML.<PWR>.Value)
            ConStr = "Data Source=" & .Servidor & ";Database=" & .BaseDeDatos
            If ConIni.Usuario <> "" Then
                ConStr = ConStr & ";uid=" & .Usuario & ";pwd=" & .Password
            End If
            ConStr = ConStr & ";Integrated Security=True"

        End With

    End Sub
```

Cambio de base de datos

Otra parte importante relacionada con las conexiones a base de datos es cambiar de manera casi simultánea la conexión a un determinado servidor para todos los equipos que utilizan la interfaz. Por lo tanto se requiere modificar los parámetros de conexión en todas las máquinas al mismo tiempo. Para lograrlo se utiliza una función denominada *InicialesCarga(Ini)* Dicha función utiliza un archivo XML que se encuentra una carpeta especifica del servidor donde se encuentran la aplicación indicándole a la interfaz donde se encuentran todos los directorios que se van a utilizar, esto permite que al modificar dicho XML la interfaz se direccione a un nuevo archivo con parámetros de conexión distintos, permitiendo el cambio de base de datos de manera casi inmediata. Sin modificar el programa de surtido ejecutable En la función anterior se puede observar la siguiente línea que se encuentra en el código anterior.

```
Dim ConnXML As XElement = XElement.Load(Ini.Base & "\INI\Conn.xml")
```

Donde *Ini.Base* pertenece a la clase *StrucIniciales* que contiene la dirección del archivo *Conn.xml* con los parámetros de conexión.

Conexión a la base de datos

Una vez establecidos los parámetros de conexión se utiliza *System.Data.SqlClient* que es el proveedor de datos *.NET Framework* para *SQL Server*, que se utiliza para obtener acceso a una base de datos de *SQL Server*.
Al utilizar *SqlDataAdapter*, clase que pertenece a *System.Data.SqlClient,* se puede rellenar un objeto *DataSet* residente en memoria, que sirve para consultar y actualizar la base de datos.

La interfaz cuenta con tres formas de conexión a la base de datos, La primera es una conexión que únicamente modifica valores de la base de datos sin recuperar información, la segunda utiliza un objeto *DataReader* que proporciona una forma de leer una secuencia de filas sólo hacia delante en una base de datos de SQL Server, y por ultimo una función que permite recupera y modificar mediante tablas utilizando un *DataTable*. A continuación se muestra la función de conexión utilizando la clase *DataReader* y una consulta a la base de datos.

```
Public Function ConnDr(ByRef DR As Data.SqlClient.SqlDataReader, ByVal strSQL As String) As
Data.SqlClient.SqlConnection
'ESTABLECE CONEXIÓN, SE ASIGNA EL COMANDO DE BUSQUEDA, Y SE INGRESAN VALORES AL LECTOR DE
DATOS SqlDataReader.

        ConnDr = Nothing
        Try 'Estableciendo conexión con base de datos
            Dim Cmd As Data.SqlClient.SqlCommand
            ConnDr = New Data.SqlClient.SqlConnection(ConStr)
            ConnDr.Open()
            'Asignacion del comando de búsqueda
            Cmd = New Data.SqlClient.SqlCommand(strSQL, ConnDr)
            'Ingresan datos al lector de Datos
            DR = Cmd.ExecuteReader
        Catch E As Exception
            Clipboard.Clear()
            Clipboard.SetText(strSQL)
            MsgBox(E.Message, MsgBoxStyle.Critical)
            ConnDr.Close()
        End Try
End Function

Private Sub CargaDepto(ByVal NumCli As String)
     Dim strSQL As String
     Dim DR As Data.SqlClient.SqlDataReader = Nothing
     Dim Conn As Data.SqlClient.SqlConnection = Nothing
     Try
        strSQL = "SELECT *  FROM DEPARTCLI DC, DEPARTDUR DD" & vbNewLine
        strSQL = strSQL & "WHERE DC.IDDEPDUR = DD.IDDEPDUR" & vbNewLine
        strSQL = strSQL & "AND   NUMCLIP = '" & NumCli & "' " & vbNewLine
        strSQL = strSQL & "AND   DD.IDDEPDUR = '" & Mid(Ped.ArtGen(1).CodArtP, 2, 1) & "'"

        Conn = ConnDr(DR, strSQL)

        While DR.Read()
             GBDepto.Visible = True
             TxtDepto.Text = DR("IDDEPCLI")
             TxtDeptoDesc.Text = DR("DESCRIPCION")
             Exit Sub
        End While
        GBDepto.Visible = False
        Catch Ex As Exception

               MsgBox(Ex.Message, MsgBoxStyle.Critical, "Error Carga Departamento")

        Finally
            Conn.Close()
        End Try

End Sub
```

A prueba de errores

Como ya mencioné anteriormente la eliminación y detección de errores, tiene un enfoque sistémico. Particularmente este programa tiene algunos mecanismos para evitar el error. La mayoría de los comandos de la interfaz se operan mediante el lector de códigos de barras lo que simplifica y limita las operaciones que puede realizar un preparador, evitando errores involuntarios. Indica de forma clara mediante un código de colores las cantidades surtidas. Verifica que las cantidades de empaque sean múltiplos de las cantidades de piezas solicitadas y permite ajustar las cantidades a surtir, bloqueado la inserción de datos inválidos. Cuenta con una función que impide que un pedido no sea surtido al 100%.

Esta última modalidad se encuentra deshabilitada debido a que existe un retardo en la actualización de inventarios en el ERP, lo que ocasionaba algunos errores a la hora de preparar los pedidos.

Programa multiusuario y sin retardos

El programa está diseñado para que varios empleados puedan trabajar en un mismo pedido al mismo tiempo y que la captura de artículos mediante códigos de barras no genere retardos. Esto se logra mediante un trabajo conjunto entre la interfaz y el manejo de la base de datos.

Referente al trabajo multiusuario los valores que se insertan a la base de datos no contienen contadores o identificadores globales que se pudieran duplicar por algún conteo simultaneo que se estuviera realizando, la inserción de datos se limita a los valores por caja, que abarca una cantidad limitada de datos que se encuentran en su totalidad en una sola estación de trabajo, Lo cual permite dosificar la información que se inserta en la base de datos sin que se superponga. Los contadores e identificadores globales los asigna la base, quien identifica el orden en el cual los valores se insertan de una manera exacta.

Para reducir el retardo en la interfaz se evita hacer consultas continuas e innecesarias a la base de datos. Esto se logra importando la totalidad del pedido al inicio de la carga cuando este se selecciona. Almacenando la información en cada una de la máquinas mediante unas series de clases que contienen una estructura similar a las tablas de la base. Esto permite que los valores del pedido se trabajen independientemente en cada una de las maquinas evitando colas.

Las clases que están relacionadas con la inserción de pedidos son las siguientes:

- ClsArtículos
- ClsArtCant
- ClsArtGen
- ClsCaja
- ClsPedido
- ClsSucursal
- ClsSucursales

Para evitar el retardo al escanear un código, solamente se modifican las clases que se encuentran en la memoria dinámica de la estación. Esto hace que la lectura de los códigos sea sumamente ágil, actualizando las clases de una manera casi instantánea.

Como ya mencioné la inserción de datos se realiza cuando se imprime una etiqueta. En ese momento el preparador tiene que cerrar la caja utilizando cinta inviolable que le toma aproximadamente 30 segundos, tiempo suficiente para hacer la inserción a la base sin que el preparador lo perciba. Esto permite que la inserción de datos de mercancía sea muy eficiente.

Web Services

La interfaz cuenta con una serie de procesos especiales que permiten establecer comunicaciones remotas para el intercambio de información comercial a través de servicios Web, esto permite que de manera automática se actualicen las bases de datos de los clientes. Generando citas de entrega en un menor tiempo.

Para establecer una conexión con servicios Web remoto mediante Visual Basic .NET, es necesario dar de alta la referencia donde se encuentra alojado el servicio. A continuación se muestra la conexión a un servicio Web de un cliente denominado *JumpStartService* .

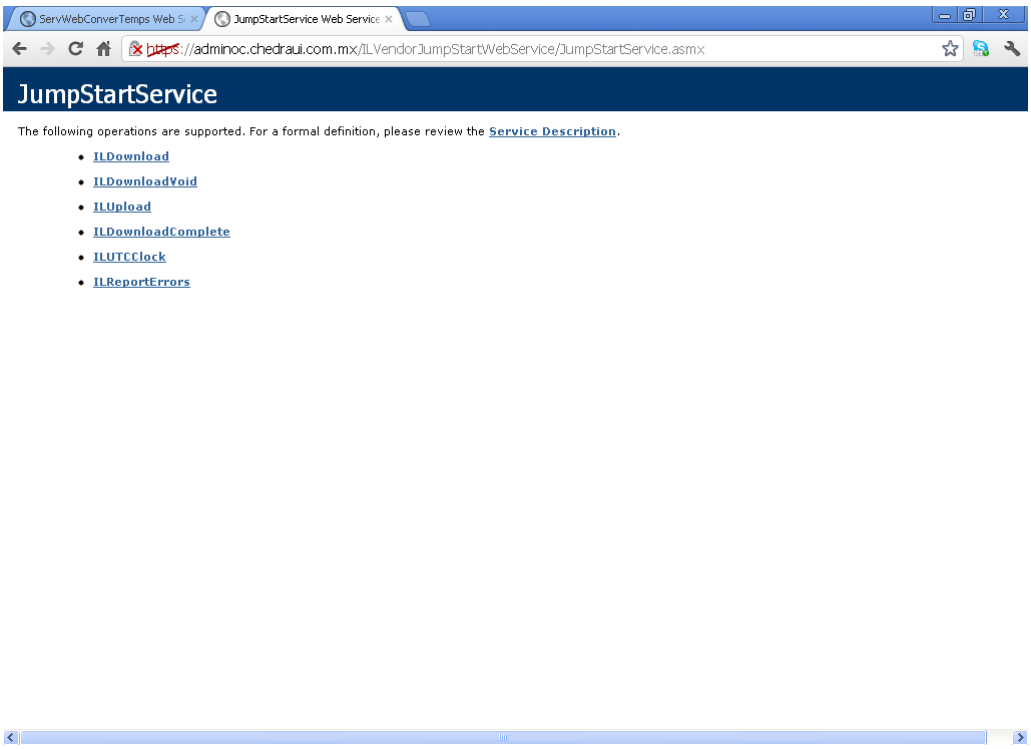


FIG. 73 SERVICIO WEB

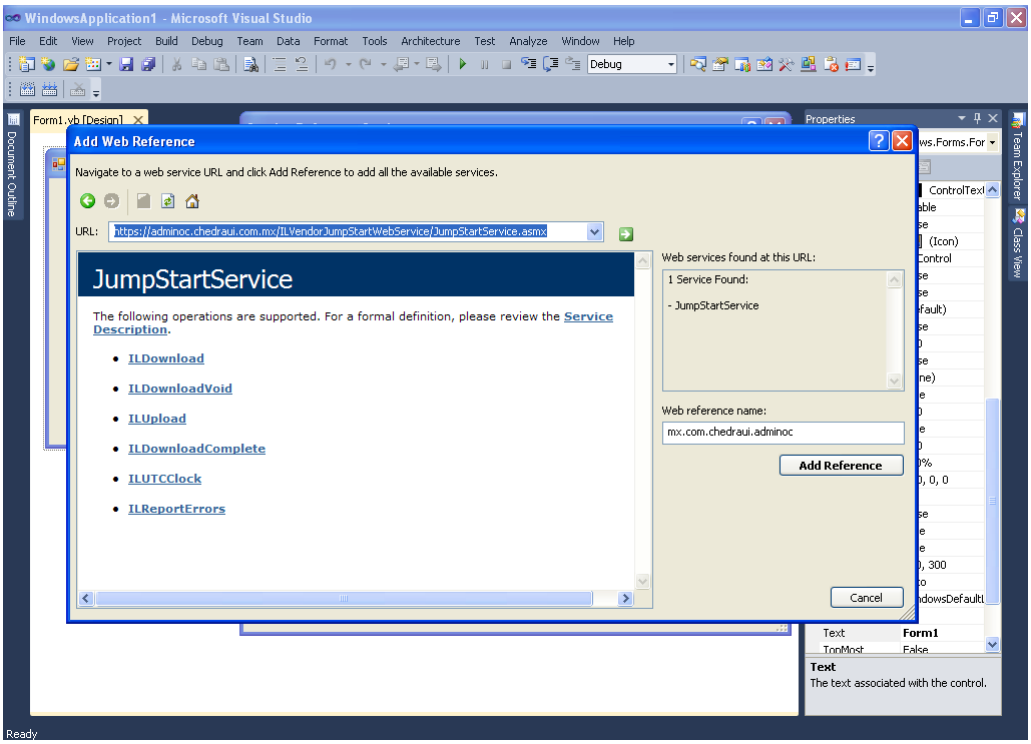


FIG. 74 SERVICIO WEB REFERENCIADO EN VISUAL BASIC.NET

Una vez que el servicio Web se encuentra dado de alta, este se comporta como cualquier referencia de Visual Basic .NET. A continuación se muestra la función de transferencia de información mediante el Servicio Web a una cadena.

```
Private Sub ChedrauiWebService(ByVal Usuario As String, ByVal Contraseña As String, ByVal
NombreMaquina As String, ByVal Directorio As String, ByVal DirectorioSistema As String, ByVal
VersionApp As String, ByVal VersionSO As String, ByVal TamañoAreaTrabajo As String, ByVal
TipoSeguimiento As String, ByVal Except As String, ByVal InfoAdicional As String, ByVal
txn_marquer As String)

        Dim Respuesta As String
        Dim WebService As New mx.com.chedraui.adminoc.JumpStartService


        WebService.InitializeLifetimeService()
        Respuesta = WebService.ILDownload(1, VersionApp, txn_marquer)
        Respuesta = WebService.ILDownloadComplete(1, 1, txn_marquer)
        WebService.ILDownloadVoid(1, 1, "1")
        If WebService.ILReportErrors(Usuario, NombreMaquina, Contraseña, "", "", _
                                     Directorio, DirectorioSistema, VersionApp, _
                                     VersionSO, TamañoAreaTrabajo, TipoSeguimiento, _
                                     Except, InfoAdicional, txn_marquer) Then

            WebService.ILUpload(Directorio, 1, VersionApp, txn_marquer)

        End If
        WebService.ILUTCClock()
        WebService.Discover()
        WebService.Abort()
        WebService.Dispose()

    End Sub
```

En el caso de este Servicio Web, se indica la ruta donde se encuentran archivos en formato CSV previamente creados. Estos archivos contienen el detalle de lo surtido. El Servidor remoto establece un enlace y lee todos los archivos que se encuentran en esa carpeta. Mediante el portal de cliente el personal de ventas verifica que los archivos se encuentran correctamente cargados y obtienen la fecha para la entrega de la mercancía.

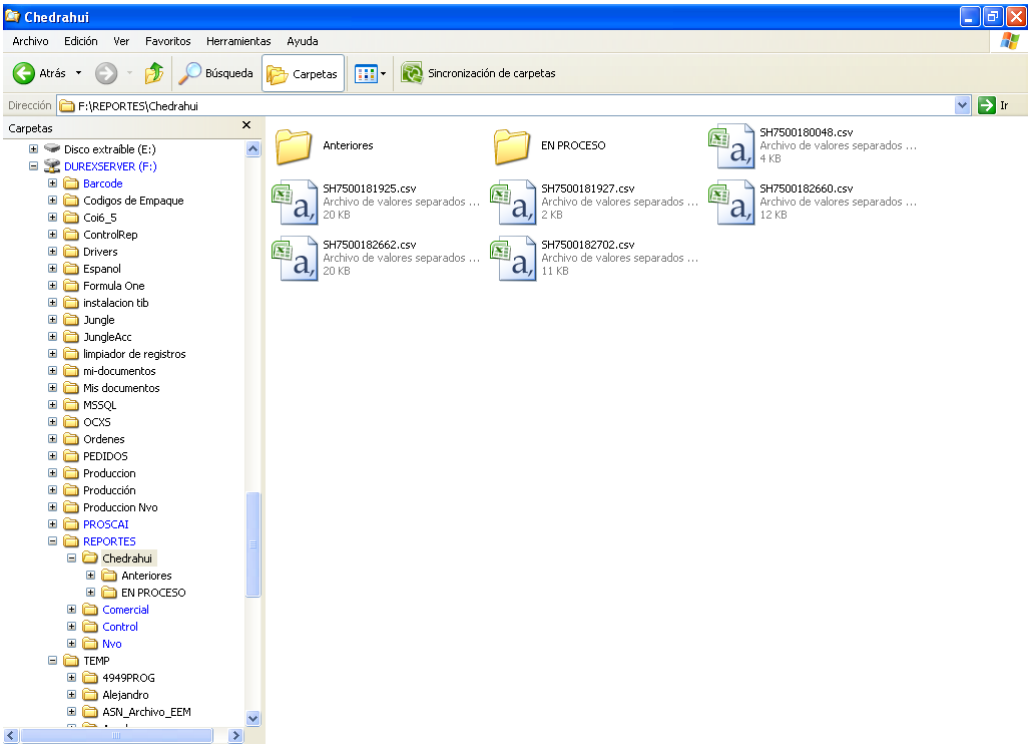


FIG. 75 CARPETA CON ARCHIVOS .CSV

SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116000_00100	6	M1036000000000289921 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116002_00100	6	M1036000000000289921 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116015_00101	6	M1036000000000289922 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116002_00102	6	M1036000000000289924 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116002_00104	6	M1036000000000289926 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116009_00104	6	M1036000000000289926 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	1323460_00104	6	M1036000000000289926 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116009_00105	6	M1036000000000289928 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116015_00105	6	M1036000000000289928 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	1238014_00105	6	M1036000000000289928 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	116018_00108	6	M1036000000000289930 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	1323460_00108	6	M1036000000000289930 6
SH001036201204120005	7500182662	20120411	20120412	20120411	20120418	20120419	115998_00011	6	M1036000000000289801 6

FIG. 76 FRAGMENTO DE UN ARCHIVO .CSV

```
POST /ILVendorJumpStartWebService/JumpStartService.asmx HTTP/1.1
Host: adminoc.chedraui.com.mx
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://jumpstart.infolink.manh.com/InfolinkVendorJumpStartWebServices/ILUpload"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ILUpload
xmlns="http://jumpstart.infolink.manh.com/InfolinkVendorJumpStartWebServices">
      <origFileName>string</origFileName>
      <mode>int</mode>
      <appVersion>string</appVersion>
      <txn_marker>string</txn_marker>
    </ILUpload>
  </soap:Body>
</soap:Envelope>
```

Fácilmente modificable

La estructura de la interfaz está diseñada mediante módulos que pueden ser compilados independientemente del programa principal. Esto permite que las modificaciones que se realicen para los clientes se puedan hacer sin alterar el ejecutable que contiene la parte genérica.

Dentro del programa ejecutable principal se encuentran funciones que mandan llamar clases que se localizan dentro de una DLL. Estas funciones llaman clases concatenando el tipo genérico de clase y un identificador particular de la cadena. Este identificador se obtiene de los parámetros insertados inicialmente mediante el XML en *InicialesCadena*[93]. A continuación se muestra un ejemplo de cómo se llama una clase que se encuentra dentro de una DLL.

```
CallByName(Cls, "EtiquetaCaja" & Cad.RepTipo, vbMethod, TxtIdSuc.Text, TxtNomSuc.Text, 4,
TotCajas, "CodBar", Ped.Suc(TxtIdSuc.Text), Vacio)
```

EtiquetaCaja es el identificador genérico que indica en este caso que se imprima una etiqueta, *Cad.RepTipo* es el identificador del tipo de reporte, el cual pertenece a una clase tipo *StrucCadena* que contiene los parámetros iniciales de la cadena. Permitiendo que mediante una modificación a los

[93] Ver función *InicialesCadena en Similitud y diferencias en la elaboración de un pedido.*

parámetros de la cadena en el archivo XML, se canalice la impresión de etiquetas hacia otra clase distinta sin tener que modificar el ejecutable principal.

4. RESULTADOS

1. Se Reestructuró el modelo comercial integrando un sistema de empaquetado y trazabilidad mediante una filosofía orientada a servicios. No solamente maneja procesos más elaborados en la preparación de pedidos, sino que es una metodología completa de vinculación con el cliente. En la actualidad el 100% de los pedidos de los clientes de la empresa se elaboran utilizando el nuevo sistema.

   Como se muestra en el capítulo de implementación en la sección *Secuencia propuesta para el modelo comercial y su integración con el nuevo modelo de distribución*, se creó un nuevo modelo de distribución y trazabilidad que es capaz de importar información e integrarla de manera automática al sistema comercial (*ERP*). El sistema se implemento de manera exitosa desde hace varios años.

2. La estabilidad de la base de datos es otro punto importante a evaluar dentro de los resultados de esta tesis, hasta el momento se han surtido *492,824* cajas y *28, 632,078* de piezas sin ningún contratiempo. Mostrando lo robusto de la misma.

   De la consultas en el *Anexo XVI*, *Consulta de resultados a la base de datos*, se muestra que han sido *86,500* piezas surtidas diarias con 6 máquinas al 80% de su capacidad. Tomando en cuenta los resultados obtenidos de *Microsoft SQL Profiler* se puede comprobar que el Almacén de Producto Terminado puede manejar aproximadamente en un día *130,000* piezas o *22,500* paquetes trabajando con 8 mesas de trabajo (número máximo de máquinas en APT). Por mesa se puede preparar *16,250* piezas diariamente o *2,800* paquetes, sin contratiempos. Lo que implica que el sistema tiene una capacidad de procesar al menos *47,000,000* de piezas al año. Cumpliendo sobradamente con los requerimientos actuales de la empresa. Inclusive se podrían aumentar el número de estaciones sin que la integridad de la base de datos se encuentre comprometida aumentando la capacidad de surtido del sistema.

3. Otra forma de evaluar el desempeño de la aplicación es ver la aceptación que ha tenido el sistema dentro de la empresa. A continuación se muestra algunas gráficas obtenidas mediante consultas a la base de datos que presentan la integración del sistema dentro del modelo comercial.[94]

   Actualmente el sistema se utiliza para surtir todos los pedidos de la empresa, Surtiendo a 179 distintos clientes.

[94] Ver Anexo XVI Consulta de resultados a la base de datos

El aumento de pedidos surtidos utilizando el sistema también ha sido considerable, en la actualidad se han procesado casi 20,000 pedidos. Mostrando no sólo la aceptación del sistema sino la estabilidad de la base de datos.

4. El sistema redujo los tiempos de preparado de mercancía, facturación, envío de documentación y embarque en más de un 50%

   Los tiempos de preparado se redujeron considerablemente si no se toman en cuenta problemas en el suministro de mercancía al Almacén de Producto Terminado.[95] La diferencia en los tiempos de preparado de cajas entre los dos sistemas varía dependiendo de la composición de artículos que contiene cada caja. Una caja con artículos de diferente tipo en el que un empleado tiene que contar y anotar las cantidades de cada uno de los artículos en la caja –como se hacía con anterioridad-, es un proceso complicado y lento siendo común que se generen errores en el

conteo. A diferencia del sistema actual que el escaneo de la mercancía permite la sumatoria automática de los artículos surtidos, haciendo de este último un proceso más fácil y rápido. El otro caso, una caja que contiene muchos artículos de un mismo tipo resulta fácil de contar y empacar, anotar el número total de artículos por caja resulta menos complicado. Mediante el sistema actual este proceso, en que se tiene que escanear cada uno de los artículos solicitados, resulta un poco más tardado. Sin embargo utilizando el modelo anterior en cualquiera de los dos casos es necesaria, una doble revisión de las cajas. Después debe hacerse el pegado de etiquetas y, una vez que el pedido ha sido terminado, se procede a su facturación manual. Al organizar todos éstos pasos en un mismo proceso se logra que los tiempos de preparado se reduzcan en más de un 70% pues todo se genera automáticamente – inclusive en las cajas con un solo artículo-.

5. El sistema genera toda la información que solicitan los clientes de manera automática e instantánea, permitiendo a la empresa enviarla según sus conveniencias, para que esta sea oportuna.
   El sistema actualiza tanto la información referente a los pedidos -utilizando tecnología de *Servicios Web*- como la carga de ASN, a través de portales de los distintos clientes permitiendo que el manejo de la información sea mucho más eficiente y rápida; clientes como *Comercial Mexicana, Chedraui, Liverpool, Walmart* reciben, una vez que el pedido ha sido preparado, toda la información que requieren en relación a él de manera casi instantánea, inclusive antes de que los pedidos estén facturados.

6. Se redujeron las pérdidas de mercancía, y se mejoró el manejo del inventario.

   El control del inventario en relación al manejo de la mercancía ha mejorado de manera considerable. Aunque el sistema de inventario es controlado por el ERP se ha generado información suficiente para detectar las fugas de mercancías de manera más puntual.

7. Métodos de validación de información,

   La generación automática, tanto de información como de impresión de documentación, evitó una gran cantidad de errores en: la captura manual de cantidades en el surtido, el conteo de mercancía, la inserción manual de valores al ERP, la generación manual de documentación al cliente, la impresión manual y el pegado de etiquetas en cajas.

   Además, los responsables de Departamento de Envíos tienen, mediante tecnologías de trazabilidad, un seguimiento puntual de las cajas hasta la entrega de la mercancía evitándose así la pérdida de cajas en alguna parte del proceso.

8. Se logró evitar errores involuntarios generando un modelo a prueba de error.

   El sistema ha afectado de manera muy positiva el proceso de surtido y envío de mercancía. Se han evitando la mayoría de los errores involuntarios que se generaban tanto en el Almacén de Producto Terminado (*APT*), como en la facturación de pedidos. La metodología desarrollada para el surtido y envío de mercancía, la capacitación del personal, los procesos de conteo de mercancía y la generación automática de documentación, han permitido que los errores en el surtido de mercancía se redujeran en más de un 90%.

   Un preparador puede, mediante la interfaz de surtido, detectar de manera muy precisa las piezas faltantes y puede determinar fácilmente si un pedido se encuentra surtido al 100%. El jefe de surtido de un pedido puede verificar que las cantidades totales por empleado concuerden con las cantidades

solicitadas al almacén mediante las guías de preparación. El personal de Facturación puede revisar las cantidades, tanto de inventario como de surtido del *ERP*, haciendo un modelo de detección de errores por capas con tres niveles de detección en el preparado; a esto se adiciona la ventaja de que los empleados son los que verifican las cantidades entregadas. Anteriormente existía un encargado responsable de verificar dichas cantidades y el Jefe de Almacén tenía que hacer una segunda inspección por pedido y los resultados eran mucho menores a los obtenidos con el nuevo sistema.

9. El sistema redujo la recaptura de información, evitando errores. En la *Fig. 2 Diagrama secuencial de la salida y facturación de pedidos* se indican los momentos de posible error en el proceso de salida de pedidos. En la solución propuestas como se muestra en la *Fig. 26 Diagrama secuencial de la solución propuesta*, se reducen en más de un 99% los errores por capturas manuales, empaquetado manual, impresión de guías de salida, elaboración de etiquetas, elaboración de avisos anticipados de embarque (*ASN*), pegado de etiquetas, y acomodo de cajas.

10. Mediante el nuevo volumen de datos se cuenta con mucha información que permite evaluar el desempeño, tanto de los departamentos como de los empleados, permitiendo sacar reportes de eficiencia que se entregan a la dirección. Esto permite corregir de manera más rápida posibles errores en la metodología de cada uno de los departamentos, y desarrollar nuevas prácticas de negocios.

11. Se implementaron estándares de *Servicios Web*, que permitirán el establecimiento de arquitecturas orientadas a servicios. *Framework 4.0* de *.NET* es la respuesta de *Microsof*t al creciente mercado de los negocios en entornos Web, como competencia a la plataforma J*ava* de *Oracle Corporation* y a los diversos *framework* de desarrollo web basados en *PHP*. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo. Siendo el estándar para el desarrollo de *Servicios WEB*, por parte de *Microsoft*.

12. Se establecieron motores y lenguajes de datos compatibles con los lineamientos de la empresa.

    El lenguaje de programación *Visual Basic.NET* y *MSQLSERVER* 2008, que son el lenguaje de programación y el manejador de bases de datos que adoptó la empresa para el desarrollo de sus aplicaciones, son el marco de referencia para el manejo de la base de datos del sistema que se desarrolló.

13. Se optimizaron los tiempos de programación y adaptación del sistema.

    Dado que todos los sistemas de la empresa se encuentran diseñados en *Visual Basic* y *MSSQLSERVER*, y no se piensa hacer un cambio de ERP, la solución que desarrollé, es la mejor opción para esta. Dado que el costo por el manejo de *MSQLSERVER y Visual Basic* ya había sido compensado pues los demás sistemas están elaborados con esas herramientas; programar en éstos lenguajes no representaba un costo adicional y, desde el punto de vista administrativo, al área de sistemas le resulta más fácil gestionar software que este desarrollado en un mismo lenguaje de programación. Por otro lado el ERP no cuenta con sistemas de trazabilidad y surtido de pedidos lo suficientemente especializado como para cubrir los requerimientos de la empresa. El sistema que desarrollé llena ese vacío ya que permite una serie de funcionalidades adicionales, diseñadas a medida, cubriendo requerimientos muy puntuales.

En relación al tiempo de desarrollo del sistema, una vez que fue elaborado el plan de trabajo. Las entregas eran mucho más rápidas que los cambios administrativos que se requerían para implementarse. Por ejemplo el desarrollo de aplicaciones de resguardo tardó en elaborarse un mes, sin embargo para que entrara en funcionamiento el personal de sistemas tuvo que hacer pruebas exhaustivas, posteriormente contratar o cambiar de puesto al personal responsable del manejo del software, ya que son puestos nuevos creado específicamente con el objetivo de recibir cajas en una aduana, posteriormente se requiere capacitar al personal. En promedio esto toma de uno a dos meses por entrega. De manera similar agregar un cliente al sistema, desde el punto de vista de desarrollo, no requiere más que la modificación de algunos parámetros y la adecuación de ciertos reportes especiales dependiendo del cliente. Pero desde el punto de vista administrativo, se requieren arreglos comerciales entre el cliente y la empresa, periodos de prueba asignado por el cliente, capacitación del personal de ventas responsable de gestionar las entregas, capacitar el personal de almacén, validación de los procesos, tanto de parte del cliente como de la empresa, procesos que podían llegar a tardar hasta cuatro meses, tiempo necesario para entregar una nueva aplicación contemplada en el plan de trabajo.

Un ejemplo que indica que tan considerable ha sido la reducción de errores gracias al sistema desarrollado, involucrando todos los incisos anteriores consiste en la entrega de mercancía a *Comercial Mexicana.* Esta entrega, que se realiza a través de Centros de Distribución, se divide en 3 diferentes tipos, dependiendo de los niveles de servicio que *Comercial Mexicana* le otorga a sus proveedores: Vía I, Vía II y Vía III. La prioridad para recepción de mercancía así como la asignación de citas depende del tipo de Vía siendo la Vía III la de mayor preferencia y la I la menor. La entrega de mercancía por la Vía III es mucho más rápida, ya que no se realiza auditoria a las cajas que se entregan.[96]

La empresa, objeto de este estudio, pasó de entregar su mercancía de Vía I a Vía III en tan solo un año a partir de que el sistema que diseñe entró en vigor. Los niveles de servicio fueron tan altos que paso a formar parte de un programa piloto de *Comercial Mexicana* denominado, *Sistema de Entrega por Pallet Blindado*. Donde, en premio al servicio dado, se elabora un pago anticipado (por letra electrónica) por el total de la mercancía surtida. El pago del pedido no se realiza por los totales de la factura electrónica que se envía posteriormente sino en función de los totales enviados por el *aviso anticipado de embarque (ASN)* -que se generan mediante el sistema que diseñé- que los envía electrónicamente para que se asigne una cita de entrega de mercancía. La factura consolidada de varios pedidos surtidos se envía posteriormente de manera electrónica al portal de *Comercial Mexicana*. Otra ventaja es que no se realiza ninguna auditoria a las cajas que se envían directamente, sin revisar, a las sucursales que les corresponde; al ser las sucursales el único lugar donde se verifica que las cantidades enviadas sean correctas, la entrega de mercancía es muy rápida y se evitan problemas de entrega. Ahora bien, si los datos de la factura no concuerdan con los valores entregados en el ASN o en las cantidades entregadas por sucursales, *Comercial Mexicana* aplica una serie de penalizaciones elevadas; hasta el momento la empresa no ha recibido una sola. Esto ha permitido que la empresa se pueda capitalizar de una manera más rápida y ha evitado considerablemente las devoluciones aumentando su competitividad. Otra ventaja adicional que ha tenido la implementación del sistema en *Comercial Mexicana* ha sido que durante el primer año se generó un ahorro de aproximadamente $450,000 pesos, debido a los pedidos consolidados. Ya que varios pedidos podían ser enviados en un mismo flete y no se tenían que usar cajas individuales por pedido, consolidando todos los artículos para una sola sucursal en cajas compartidas.

[96] Ver Anexo Comercial Mexicana ó manual de proveedores Comercial Mexicana http://www.provecomer.com.mx/prvd/manual_prov.pdf

FIG. 77 EMBALANDO CAJAS PARA PALLET BLINDADO

5. DISCUSIÓN DE RESULTADOS

Si bien ha sido factible diseñar y establecer un sistema de trazabilidad y distribución de mercancía al 100%, su orientación a servicios no ha podido ser implementado en su totalidad debido a que las actualizaciones al programa ERP para poder establecer una comunicación a través de *Servicios Web* para la importación y exportación de valores, actualmente no se encuentran disponibles en el mercado. Lo cual ha imposibilitado que el sistema que desarrollé se pueda integrar al ERP con un modelo completamente orientado a servicios. Sin embargo el mecanismo de descarga de información mediante documentos PRN en carpetas compartidas ha funcionado sin problemas, y las interfaces están diseñadas para poder migrar fácilmente a un modelo basado en *Servicios Web* una vez que el ERP sea actualizado.

Los resultados obtenidos muestran que las tablas de pedidos y trazabilidad de la base de datos que fundamentan esta tesis están estructuradas de manera tal, que la posibilidad o corrupción de información sea improbable. El acceso a la información es altamente eficiente y los errores posibles están casi totalmente controlados mediante el modelo de disparadores o *triggers* y los procedimientos almacenados desarrollados. Además, está prevista la migración de la información de los pedidos antiguos a tablas históricas no mencionadas en esta tesis, evitando el incremento de tiempo de espera en las consultas.

En cuanto a la reducción de costos, la empresa redujo sus gastos de personal en un 20%, y en los clientes importantes, como Comercial Mexicana, se ha mostrado una reducción significativa en la utilización de cajas así como la disminución en gastos por fletes. Los métodos de validación de información así como de detección de errores involuntarios han reducido significativamente los gastos de la empresa por devoluciones. Aunque no se cuenta con un registro anterior del total de cajas perdidas en el trayecto de la fábrica a los clientes, se sabe que éste era un problema importante que se ha reducido prácticamente a cero. Además se ha establecido un control más eficiente en la salida de productos del APT, lo que ha permitido detectar más fácilmente posibles fugas de este. En cuanto a las devoluciones como a las fugas de almacén se podrían reducir aún más este tipo de errores si se contara con sistemas de RFID que permitiera detectar la mercancía de manera automática.

Si bien los tiempos de entrega se han reducido en más de un 50% actualmente la naturaleza del ERP, que impide que el sistema sea totalmente orientado a servicios, genera retardo en la actualización de inventarios desde la preparación de un pedido a su facturación.

En lo referente al envío de información a los clientes ésta ha llegado a ser, en algunos casos, demasiado rápido. Se han presentado casos donde un pedido que no está surtido al 100% y sobre el cual se envía información electrónica, no puede ser resurtido debido a que la información ya ha sido enviada al cliente y la mercancía llega minutos después[97]. Lo anterior condujo a modificar algunos procesos de información electrónica ampliando el margen de tiempo entre la preparación de la mercancía y la información que sobre ella se enviará.

Al permitir el establecimiento de diferentes tipos de enlaces entre la base datos de la empresa con las bases de datos de los clientes importantes, estructurados a la medida del cliente y en los tiempos establecidos por el mismo, el sistema ha permitido que las relaciones entre los clientes y la empresa se consoliden. Sin embargo, los servicios al cliente podrían ser ampliados a otros sectores del sistema de trazabilidad desde la parte productiva como al envío y recepción de mercancía.

[97] Se han presentado incidentes de este tipo por la falta de comunicación entre departamentos, y a la falta de un sistema de trazabilidad interno.

La reducción de costos, tiempos de entrega, así como la consolidación de relaciones comerciales de una empresa son factores que aumentan su competitividad. Lo que sumado al aumento de la información y su disposición en la toma de decisiones abren nuevas oportunidades para el desarrollo de la misma.

CONCLUSIONES

La defensa de esta tesis para obtener el título de Ingeniero en Computación, me ha permitido estructurar y sustentar mi práctica profesional con estándares universitarios, que han incidido en la calidad de mi trabajo.

También disfrutar de la investigación de una hipótesis para ver reflejada en la práctica, la teoría de la ingeniería en computación y sus beneficios productivos. Este ejercicio permitió sintetizar y estructurar muchos de los conocimientos adquiridos durante la licenciatura posibilitando la apertura de nuevos proyectos sustentados en tecnología de punta.

Considero que ésta experiencia práctica puede ser útil para los interesados en la aplicación de bases de datos y trazabilidad ya que complementa la investigación universitaria, y muestra los matices de la comunicación de la ingeniería en computación con el funcionamiento de una empresa. Donde el ingeniero es un agente de definición de los procesos de la misma, posicionándose y articulándose adecuadamente con los procesos administrativos y gerenciales.

La formación teórica difícilmente puede mostrar la complejidad de los procesos de ajuste que se observan en las bases de datos reales, donde las variaciones del volumen de la información y sus modificaciones estructurales juegan un papel fundamental.

El desarrollar sistemas orientados a servicios implica evaluar el riesgo de la dependencia con sistemas externos que pueden variar impredeciblemente, como es el caso del *ERP*, por lo que el ingeniero no debe limitarse a la parte teórica sino desarrollar su capacidad de percibir detalladamente la dinámica de la empresa, de sus objetivos y su contexto.

APÉNDICE

I. PIEZAS PROMEDIO POR TIPO DE CLIENTE (2008)

Cantidad de piezas promedio por tipo de cliente		
Tipo Cliente	**Total Pzas**	**Promedio**
Tiendas Auto Servicio	2,482,603	118,219.19
Varios	280,518	3,339.50
Almacenes y Tiendas Comerciales tipo Z	383,488	18,261.33
Almacenes y Tiendas Comerciales tipo M	182,372	45,593.00
Total general	3,328,981	25,607.55

II. PIEZAS PROMEDIO PEDIDO POR CLIENTE (2008)

Cantidad de piezas promedio por pedido por cliente	
NUMCLI	**PZAS**
CMX001	827,082
CHE001	408,888
WMX002	381,402
WMX001	259,752
SOR001	185,418
MIX001	179,172
SUB001	143,142
ZMDB01	127,988
SOR002	63,228
VDC001	63,096
LIV001	58,812
ZMAA01	57,324
VGI001	41,268
GAR001	39,691
VAZ001	32,838
LEY001	32,664
SEO001	30,678
ZMJZ01	28,104
ZMME01	25,200
ZMRG01	22,020
ZMJZ04	15,804
ZMSE02	15,288
VAP002	14,892
VRM001	14,226
ZMAM01	13,704
ZMLH01	11,412
ZMSE01	11,184
ZMRO01	11,164
IST001	10,578
GCC001	9,748
VTC001	8,850
ZMME03	8,664
SEA001	8,652
VGP001	8,346
ZMMP01	8,220
VTG001	7,488
VBO001	6,744
VDM001	6,036
TBB001	5,814
ZMSE03	5,436
PRI001	5,358
VNM003	5,034
ZMME02	5,004
ZMCF01	4,692
CYA001	4,460
ZMJC01	4,236
VCA010	3,726
VVL002	3,714
VJL001	3,456

III. MODELO EN ESPIRAL

El modelo en espiral fue propuesto por Boehm en 1986. Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Se suele interpretar como que dentro de cada ciclo de la espiral se sigue un *Modelo en Cascada*[98], pero no necesariamente debe ser así. El *Modelo en Espiral* puede verse como un modelo evolutivo que conjuga la naturaleza iterativa del *Modelo de prototipos (MCP)* [99] con los aspectos controlados y sistemáticos del *Modelo en Cascada*, con el agregado de gestión de riegos.

El modelo está dividido en cuatro partes o actividades fundamentales:

- Determinar Objetivos
- Análisis del riesgo
- Planificación
- Desarrollar y probar

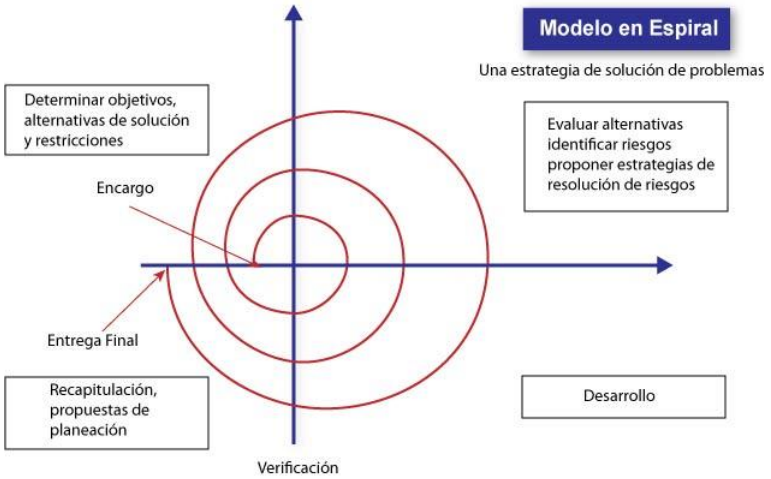


FIG. 78 CUATRO FASES O ACTIVIDADES DEL MODELO EN ESPIRAL

Este modelo tiene en cuenta fuertemente el riesgo que aparece a la hora de desarrollar software. Para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgo más asumible y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelve a evaluar las distintas nuevas alternativas y riesgos y se realiza otra vuelta de la espiral, así hasta que llegue un momento en el que el producto software desarrollado sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo.

[98] Enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma secuencial. Siguiendo la siguiente metodología de desarrollo. Análisis de requisitos, Diseño del Sistema, Diseño del Programa, Codificación, Pruebas, Implantación y Mantenimiento

[99] El Modelo de prototipos que pertenece a los modelos de desarrollo evolutivo, El prototipo debe ser construido en poco tiempo, usando los programas adecuados, de diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

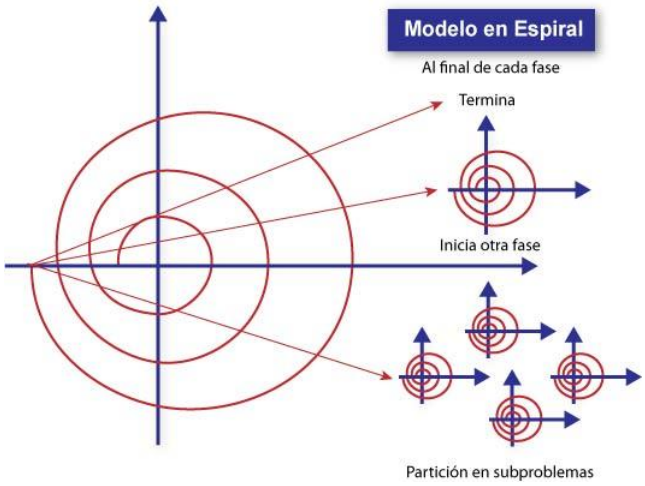


FIG. 79 TÉRMINO DEL MODELO EN ESPIRAL

Las razones por las cuales se tomó la decisión de utilizar un modelo en espiral fueron las siguientes:

- Establecer objetivos concretos cuantificables a corto plazo, que pudieran ser integrados paulatinamente al modelo comercial sin tener un sistema final terminado.
- Detectar posibles fuentes de riesgo, que permitan encontrar soluciones viables.
- Permite hacer evaluaciones periódicas que permitan evaluar el beneficio de dicho sistema, si las soluciones propuestas efectivamente contribuyen a una mejora sustancial y de no ser así permite hacer cambios de objetivos de una manera más rápida.
- Establecer soluciones más ágiles y certeras, que estén a la par de los procesos de mejora continua que demandan los clientes.

IV. CANTIDAD DE RENGLONES POR TABLA (DIC. 2011)

TABLA	REGISTROS
PEDALMDETSURT	585,690
PEDALMDETSOL	253,580
PEDALMCAJAS	92,040
PEDALMSUC	83,452
PEDALMART	57,835
CAJAPROCESO	29,388
MSKU	21,145
PEDALMGLOB	4,828
CLIENTESDET	3,282
MARTIC	2,091
PEDALMETAPAS	1,431
CLIENTESGLOB	327
CEDIS	227
CATEMPLEADOS	19
DEPARTCLI	8
PROCESOCOLOR	6
PROCESOS	6
PUÉSTOS	6
DEPARTDUR	5
PEDALMTEMP	3
STATUSETAPA	3
TIPOPEDIDO	2
TIPOPREPARACION	2

V. LISTA DE CLIENTES SURTIDOS CON EL NUEVO SISTEMA (DIC. 2011)

2008	2009	2010	2011			
CHE001	CHE001	CHE001	ASA001	VBM003	VHT001	VSS001
WMX001	CMX001	CMX001	CHE001	VBM004	VIR002	VSS002
	CYA001	CYA001	CMX001	VBN001	VIR003	VTC001
	GCC001	GCC001	COP001	VBO001	VIS001	VTE001
	LIV001	LEY001	CYA001	VCA008	VJL001	VTG001
	MIX001	LIV001	GAR001	VCA010	VJO001	VTO001
	VNM001	MIX001	GCC001	VCB002	VJO004	VUC001
	VNM002	PHI001	HEB001	VCC002	VJU001	VUN001
	VNM003	SEA001	IST001	VCC003	VJU004	VUP001
	WMX001	SOR001	LEY001	VCC004	VKM001	VVA003
	WMX002	SOR002	LIV001	VCD001	VLA004	VVE002
		SUB001	MGZ001	VCE001	VLI002	VVL002
		VNM001	MIX001	VCE002	VMA002	VWA001
		VNM002	MLM001	VCE005	VMA003	VYU001
		VNM003	MOV004	VCP001	VMA008	WMX001
		WMX001	MTM001	VCS001	VMC005	WMX002
		WMX002	MTS001	VCT001	VMC013	ZMAA01
			PHI001	VCW001	VMD001	ZMAM01
			PRI001	VCZ002	VME006	ZMCF01
			SEO001	VDA001	VMI001	ZMCF02
			SOR001	VDC001	VMO001	ZMCF03
			SOR002	VDD001	VMR001	ZMDB01
			SUB001	VDL001	VMR004	ZMJC01
			TBB001	VDM001	VMS005	ZMJV01
			TGR001	VEG001	VNM001	ZMJZ01
			VAD001	VEM001	VNM002	ZMJZ03
			VAJ001	VEM002	VNM003	ZMJZ04
			VAL002	VER001	VOB001	ZMLH01
			VAL004	VES001	VOF001	ZMME01
			VAM001	VES002	VOH001	ZMME02
			VAM002	VET001	VON001	ZMME03
			VAM007	VFE001	VPC001	ZMMP01
			VAP001	VFT001	VPP002	ZMRG01
			VAP002	VGI001	VRA002	ZMRO01
			VAR001	VGM002	VRA003	ZMSE01
			VAR004	VGN001	VRM001	ZMSE02
			VAT001	VGN002	VSA001	ZMSE03
			VAV001	VGN003	VSB001	
			VAZ001	VGP001	VSD001	
			VBA003	VGR001	VSG001	
			VBC001	VGT002	VSI001	
			VBE001	VHD001	VSI003	
			VBM001	VHI004	VSL002	

VI. SINTAXIS COMPLETA INSTRUCCIONES MICROSOFT SQL SERVER

CREATE TABLE [100]

```
CREATE TABLE
   [ database_name.[ owner ] . | owner. ] table_name
   ( { < column_definition >
      | column_name AS computed_column_expression
      | < table_constraint > ::= [ CONSTRAINT constraint_name ] }

         | [ { PRIMARY KEY | UNIQUE } [ ,...n ]
   )

[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]

< column_definition > ::= { column_name data_type }
   [ COLLATE < collation_name > ]
   [ [ DEFAULT constant_expression ]
      | [ IDENTITY [ ( seed , increment ) [ NOT FOR REPLICATION ] ] ]
   ]
   [ ROWGUIDCOL]
   [ < column_constraint > ] [ ...n ]

< column_constraint > ::= [ CONSTRAINT constraint_name ]
   { [ NULL | NOT NULL ]
      | [ { PRIMARY KEY | UNIQUE }
         [ CLUSTERED | NONCLUSTERED ]
         [WITH FILLFACTOR = fillfactor]
         [ON {filegroup | DEFAULT} ] ]
      ]
      | [ [ FOREIGN KEY ]
         REFERENCES ref_table [ ( ref_column ) ]
         [ ON DELETE { CASCADE | NO ACTION } ]
         [ ON UPDATE { CASCADE | NO ACTION } ]
         [ NOT FOR REPLICATION ]
      ]
      | CHECK [ NOT FOR REPLICATION ]
      ( logical_expression )
   }
< table_constraint > ::= [ CONSTRAINT constraint_name ]
   { [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      { ( column [ ASC | DESC ] [ ,...n ] ) }
      [WITH FILLFACTOR = fillfactor]
      [ ON { filegroup | DEFAULT } ]
   ]
   | FOREIGN KEY
      [ ( column [ ,...n ] ) ]
      REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
      [ NOT FOR REPLICATION ]
   | CHECK [ NOT FOR REPLICATION ]
      ( search_conditions )
   }
```

ALTER TABLE [101]

```
ALTER TABLE table
{ [ ALTER COLUMN column_name
   { new_data_type [ ( precision [ , scale ] ) ]
      [ COLLATE < collation_name > ]
      [ NULL | NOT NULL ]
      | {ADD | DROP } ROWGUIDCOL }
   ]
   | ADD
      { [ < column_definition > ]
      | column_name AS computed_column_expression
      } [ ,...n ]
   | [ WITH CHECK | WITH NOCHECK ] ADD
```

[100] Significado de los argumentos: http://msdn.microsoft.com/en-us/library/ms174979.aspx
[101] Significado de los argumentos: http://msdn.microsoft.com/en-us/library/ms190273.aspx

```
    { < table_constraint > } [ ,...n ]
  | DROP
    { [ CONSTRAINT ] constraint_name
      | COLUMN column } [ ,...n ]
  | { CHECK | NOCHECK } CONSTRAINT
    { ALL | constraint_name [ ,...n ] }
  | { ENABLE | DISABLE } TRIGGER
    { ALL | trigger_name [ ,...n ] }
}

< column_definition > ::=
  { column_name data_type }
  [ [ DEFAULT constant_expression ] [ WITH VALUES ]
  | [ IDENTITY [ ( seed , increment ) [ NOT FOR REPLICATION ] ] ]
    ]
  [ ROWGUIDCOL ]
  [ COLLATE < collation_name > ]
  [ < column_constraint > ] [ ...n ]

< column_constraint > ::=
  [ CONSTRAINT constraint_name ]
  { [ NULL | NOT NULL ]
    | [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      [WITH FILLFACTOR = fillfactor]
      [ ON { filegroup | DEFAULT } ]
      ]
    | [ [ FOREIGN KEY ]
      REFERENCES ref_table [ ( ref_column ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
      [ NOT FOR REPLICATION ]
      ]
    | CHECK [ NOT FOR REPLICATION ]
      ( logical_expression )
  }

< table_constraint > ::=
  [ CONSTRAINT constraint_name ]
  { [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    { ( column [ ,...n ] ) }
    [WITH FILLFACTOR = fillfactor]
    [ ON { filegroup | DEFAULT } ]
    ]
    |   FOREIGN KEY
      [ ( column [ ,...n ] ) ]
      REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
      [ NOT FOR REPLICATION ]
    | DEFAULT constant_expression
      [ FOR column ] [ WITH VALUES ]
    |   CHECK [ NOT FOR REPLICATION ]
      ( search_conditions )
  }
```

SELECT[102]

```
SELECT statement ::=
  < query_expression >
  [ ORDER BY { order_by_expression | column_position [ ASC | DESC ] }
    [,...n]   ]
  [ COMPUTE
    { { AVG | COUNT | MAX | MIN | SUM } ( expression ) } [ ,...n ]
    [ BY expression [ ,...n ] ]
  ]
  [ FOR { BROWSE | XML { RAW | AUTO | EXPLICIT }
      [ , XMLDATA ]
      [ , ELEMENTS ]
      [ , BINARY base64 ]
    }
```

[102] Significado de argumentos, así como la sintaxis de cada una de las clausulas: http://msdn.microsoft.com/es-mx/library/ms189499.aspx

```
]
   [ OPTION ( < query_hint > [ ,...n ]) ]

< query expression > ::=
   { < query specification > | ( < query expression > ) }
   [ UNION [ ALL ] < query specification | ( < query expression > ) [...n ] ]

< query specification > ::=
   SELECT [ ALL | DISTINCT ]
      [ { TOP integer | TOP integer PERCENT } [ WITH TIES ] ]
      < select_list >
   [ INTO new_table ]
   [ FROM { < table_source > } [ ,...n ] ]
   [ WHERE < search_condition > ]
   [ GROUP BY [ ALL ] group_by_expression [ ,...n ]
      [ WITH { CUBE | ROLLUP } ]
   ]
   [ HAVING < search_condition > ]
```

VII. CARACTERÍSTICAS TÉCNICAS HP PROLIANT

ESPECIFICACIONES TÉCNICAS	
Número de procesadores	1 o 2
Processor core available	4 o 6
Memoria, máxima	192 GB
Ranuras de memoria	18 ranuras DIMM
Tipo de memoria	RDIMM o UDIMM DDR3
Ranuras de expansión	2
Controlador de red	(1) 2 Puertos 1 GbE NC362i
Tipo de fuente de alimentación	opcional
Controlador de almacenamiento	(1) Matriz Inteligente B110i SATA RAID; (1) Smart Array P410/BBWC de 256 MB
Administración de infraestructura	Lights-Out 100i (LO100i)

Costo:

- Precio asequible para un sistema de dos procesadores altas prestaciones. Los costos de un servidor Proliant varian desde los $20,000 hasta los $110,000 pesos por servidor
- Optimizado para bastidor, formato de 1U para una implantación rápida y mantenimiento eficaz de soluciones en clúster y cuadrícula a gran escala.
- El chasis de fácil acceso y las etiquetas en los componentes internos agilizan y facilitan el mantenimiento y la ampliación.
- Con Easy Setup CD para una instalación rápida.
- Incluye agentes SNMP para una gestión más exhaustiva.

Tamaño:

- Gama de los más recientes procesadores Intel Xeon de 4 y 6 núcleos.
- Rendimiento para soluciones de aplicaciones de 32 bits y de 64 bits.
- Incluye DDR3 DIMM registrados y sin búfer, líderes del sector, con ampliación a 192GB para aumentar el ancho de banda y conseguir un mayor tiempo de funcionamiento y rendimiento intensivo de la memoria.
- Tarjeta de red integrada HP NC362i de doble puerto de 1Gb con equilibrado de carga, recuperación ante fallos y soporte lateral.
- En la actualidad dispone de bahías internas más ampliables para atender sus requisitos de almacenamiento, hasta cuatro unidades SATA grandes de 3,5" sin conexión caliente o soporte de hasta 8 bahías para discos duros pequeños SATA/SAS de 2,5" con conexión caliente.

VIII. CARACTERÍSTICAS TÉCNICAS IMPRESORA ZEBRA S4M

Métodos de impresión: Transferencia Térmica directa y térmica (opcional)

Construcción: en metal fundido y marco con caja de chapa de metal

Lenguaje de programación: EPL-Page Mode, ZPL I / ZPL II

ESPECIFICACIONES DE LA IMPRESORA	
Resolución	203 dpi (8 puntos / mm),300 dpi (12 puntos por mm)
Memoria	4 MB Flash, 8 MB de memoria DRAM, Ancho de impresión, 4.09 "/ 104 mm, Longitud de impresión,
Velocidad de impresión	203 dpi: 6 "(152 mm) / seg,300 ppp: 6 "(152 mm) / seg
Sensores	Reflexivo, Transmisivo

CARACTERÍSTICAS DE LOS MEDIOS	
Máxima de etiqueta y ancho de la cubierta	0.75 "/ 19.4 mm a 4.50" / 114 mm
La etiqueta mínima y ancho de la cubierta	0,75 "(19,4 mm)
Longitud máxima de etiqueta y revestimiento Máximo no continuo	39 "/ 991 mm
Longitud máxima de etiqueta y revestimiento (opcional)	39 "(991 mm)
Diámetro del núcleo	8.0 "(203 mm) O.D. en un 3" (76 mm) I.D. núcleo 6.0 "(152 mm) O.D. en un 1" (25 mm) I.D. núcleo
Grosor del soporte	0,003 "/ 0,076 mm a 0,010" / 0,25 mm
Tipos	continuo, troquelado, etiquetas, negro marca

CARACTERÍSTICAS DE LA CINTA	
Diámetro exterior	3,2 "/ 81,3 mm
La longitud estándar	984 '/ 300 m, o 1476' / 450 m
Anchura de la cinta	1.57 "/ 40 mm a 4.33" / 110 mm
Aprobaciones de agencias	IEC 60950 EN 55022 Clase B EN55024 EN 61000-3-2 EN 61000-3-3

CARACTERÍSTICAS FÍSICAS	
Ancho	10.7 "/ 272 mm
Altura	11,6 "(295 mm)
Profundidad	18,8 "(477 mm)
Peso	27,2 libras (12,4 kg)
Peso de embarque	33,5 libras (15,2 kg)

CÓDIGOS DE BARRAS
Lineal
El código 11
Código 39
Código 93
Código 128 con subconjuntos A, B / / C y códigos UCC Case

UPC-A
UPC-E
EAN-8
EAN-13
UPC-A con 2 o 5 extensiones de dígitos
Plessey
Postnet
Estándar 2-de-5
Industrial 2-de-5
Interleaved 2-de-5
Logmars
MSI
Codabar
Planeta Código
RSS (simbología de espacio reducido)
2-dimensional
Codablock
Código 49
Data Matrix
MaxiCode
Código QR
MicroPDF417
TLC 39
RSS / GS1 DataBar familia (12 códigos de barras)

Los medios de comunicación de metal cubierta con ventana transparente ampliada Element Energy Equalizer ™ (E3 ®) para una calidad de impresión superior LCD con retro iluminación del panel de control Paralelo, USB 1.1 y RS-232 puertos serie Unicode ™

IX. CARACTERÍSTICAS TÉCNICAS LECTOR DE CÓDIGOS DE BARRAS MOTOROLA SYMBOL

Características técnicas	
Patrón de lectura	Línea única
Profundidad de campo	Desde contacto hasta 8 in./20.32 cm con 100% símbolos UPC/EAN
Resolución mínima	30%
Interfaces compatibles	Interfaz de teclado, RS-232, USB
Tecnología	Láser
Tono	2± 65°
Rotación (Inclinación)	3 ± 30° from normal
Ángulo horizontal	4± 60°

X. CREACIÓN DE LA BASE DE DATOS

```
/****** Object:  Table [dbo].[CLIENTESGLOB]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[CLIENTESGLOB] (
        [NUMCLI] [char] (6) NOT NULL ,
        [NOMCLI] [varchar] (150) NOT NULL ,
        [CLIDIR] [varchar] (150) NULL ,
        [CD] [varchar] (50) NULL ,
        [EDO] [varchar] (20) NULL ,
        [CP] [varchar] (10) NULL ,
        [RFC] [varchar] (16) NULL ,
        [REFER] [varchar] (20) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[DEPARTDUR]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[DEPARTDUR] (
        [IDDEPDUR] [char] (1) NOT NULL ,
        [DESCRIPCION] [varchar] (20) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMACUMDET]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PEDALMACUMDET] (
        [NumPed] [varchar] (6) NOT NULL ,
        [IdSuc] [varchar] (6) NOT NULL ,
        [ContCaja] [int] NOT NULL ,
        [Consec] [int] NOT NULL ,
        [FecElab] [smalldatetime] NOT NULL ,
        [CodArt] [varchar] (13) NOT NULL ,
        [CodArtP] [varchar] (13) NOT NULL ,
        [NumSol] [int] NOT NULL ,
        [NumPzas] [int] NOT NULL ,
        [NumPaqCli] [int] NULL ,
        [NumPaqAlm] [int] NULL ,
        [NumPzaXPaqCli] [int] NOT NULL ,
        [NumPzaXPaqAlm] [int] NOT NULL ,
        [Precio] [money] NULL ,
        [Ordcli] [varchar] (20) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMACUMGLOB]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PEDALMACUMGLOB] (
        [NumPed] [varchar] (6) NOT NULL ,
        [OrdCli] [varchar] (20) NOT NULL ,
        [NumCli] [varchar] (6) NOT NULL ,
        [NumEmp] [varchar] (6) NOT NULL ,
        [NumSuc] [int] NOT NULL ,
        [NumCajas] [int] NOT NULL ,
        [NumPzas] [int] NOT NULL ,
        [NumPaqCli] [int] NOT NULL ,
        [NumPaqAlm] [int] NOT NULL ,
        [NumArts] [int] NOT NULL ,
        [FecIni] [smalldatetime] NOT NULL ,
        [FecVen] [smalldatetime] NOT NULL ,
        [FecElabIni] [smalldatetime] NOT NULL ,
        [FecElabFin] [smalldatetime] NOT NULL ,
        [Obser] [text] NULL ,
        [Agente] [varchar] (30) NULL ,
        [Status] [int] NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMACUMSUC]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PEDALMACUMSUC] (
        [NumPed] [varchar] (6) NOT NULL ,
        [idSuc] [varchar] (6) NOT NULL ,
        [NumEmp] [varchar] (6) NOT NULL ,
        [NumCajas] [int] NOT NULL ,
        [NumPzas] [int] NOT NULL ,
        [NumPaqCli] [int] NOT NULL ,
        [NumPaqAlm] [int] NOT NULL ,
        [NumArts] [int] NOT NULL ,
        [Obser] [text] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

```
/****** Object:  Table [dbo].[PEDALMART]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PEDALMART] (
        [NumPed] [char] (6) NOT NULL ,
        [CodArt] [varchar] (13) NOT NULL ,
        [CodArtP] [varchar] (13) NOT NULL ,
        [Sku] [varchar] (20) NULL ,
        [DescAr] [varchar] (40) NOT NULL ,
        [NumPzaXPaqCli] [int] NOT NULL ,
        [NumPzaXPaqAlm] [int] NOT NULL ,
        [ArtSurt] [bit] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMDET]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PEDALMDET] (
        [NumPed] [varchar] (6) NOT NULL ,
        [IdSuc] [varchar] (6) NOT NULL ,
        [ContCaja] [int] NOT NULL ,
        [Consec] [int] NOT NULL ,
        [FecElab] [smalldatetime] NOT NULL ,
        [CodArt] [varchar] (13) NOT NULL ,
        [CodArtP] [varchar] (13) NOT NULL ,
        [NumSol] [int] NOT NULL ,
        [NumPzas] [int] NOT NULL ,
        [NumPaqCli] [int] NULL ,
        [NumPaqAlm] [int] NULL ,
        [NumPzaXPaqCli] [int] NOT NULL ,
        [NumPzaXPaqAlm] [int] NOT NULL ,
        [Precio] [money] NULL ,
        [OrdCli] [varchar] (20) NULL ,
        [IdEtapa] [int] NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PROCESOS]    Script Date: 31/10/2011 05:51:25 p.m. ******/
CREATE TABLE [dbo].[PROCESOS] (
        [IdProceso] [tinyint] NOT NULL ,
        [DescProceso] [varchar] (50) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PUÉSTOS]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[PUÉSTOS] (
        [PUESTO] [int] NOT NULL ,
        [DEPTO] [varchar] (7) NOT NULL ,
        [NIVEL] [varchar] (2) NOT NULL ,
        [DESCRIPCION] [varchar] (30) NOT NULL
) ON [PRIMARY]
GO


/****** Object:  Table [dbo].[STATUSETAPA]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[STATUSETAPA] (
        [IdStatusEtapa] [int] NOT NULL ,
        [DescEtapa] [varchar] (16) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[TIPOPEDIDO]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[TIPOPEDIDO] (
        [IDTIPOPED] [int] NOT NULL ,
        [TipoPed] [varchar] (10) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[TIPOPREPARACION]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[TIPOPREPARACION] (
        [IDTIPOPREP] [int] NOT NULL ,
        [TipoPrep] [varchar] (20) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[TIPORECEP]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[TIPORECEP] (
        [CVE_RECEP] [numeric](2, 0) NOT NULL ,
        [DESCRIPCION] [varchar] (30) NOT NULL
) ON [PRIMARY]
```

```
GO


/****** Object:  Table [dbo].[CATEMPLEADOS]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[CATEMPLEADOS] (
        [NUMEMP] [char] (6) NOT NULL ,
        [NOMEMP] [varchar] (30) NOT NULL ,
        [APPAT] [varchar] (30) NOT NULL ,
        [APMAT] [varchar] (30) NULL ,
        [PUESTO] [int] NOT NULL ,
        [STATUS] [binary] (8) NULL ,
        [PASSWORD] [varchar] (6) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[CEDIS]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[CEDIS] (
        [NUMCLI] [char] (6) NOT NULL ,
        [IDCEDIS] [smallint] NOT NULL ,
        [DET] [varchar] (8) NOT NULL ,
        [NOMBRE] [varchar] (30) NOT NULL ,
        [DIR1] [varchar] (100) NULL ,
        [TEL] [varchar] (20) NULL ,
        [ABREV] [varchar] (8) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[CLIENTESDET]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[CLIENTESDET] (
        [NUMCLI] [char] (6) NOT NULL ,
        [IDSUC] [varchar] (5) NOT NULL ,
        [NOMSUC] [varchar] (100) NOT NULL ,
        [SUCDIR] [varchar] (150) NULL ,
        [SUCCD] [varchar] (50) NULL ,
        [SUCEDO] [varchar] (20) NULL ,
        [SUCCP] [varchar] (10) NULL ,
        [RFC] [varchar] (16) NULL ,
        [REFER] [varchar] (20) NULL ,
        [IDCEDIS] [smallint] NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[DEPARTCLI]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[DEPARTCLI] (
        [NUMCLI] [char] (6) NOT NULL ,
        [IDDEPCLI] [varchar] (6) NOT NULL ,
        [IDDEPDUR] [char] (1) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[MARTIC]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[MARTIC] (
        [CODART] [varchar] (13) NOT NULL ,
        [CODARTP] [char] (13) NOT NULL ,
        [DESCAR] [varchar] (40) NOT NULL ,
        [UNIMED] [tinyint] NOT NULL ,
        [TIPMAT] [int] NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMGLOB]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[PEDALMGLOB] (
        [NUMPEDP] [char] (6) NULL ,
        [NUMPED] [char] (6) NOT NULL ,
        [NumCli] [char] (6) NULL ,
        [NomCli] [varchar] (150) NULL ,
        [OrdCli] [varchar] (20) NOT NULL ,
        [FecIni] [smalldatetime] NOT NULL ,
        [FecElabIni] [smalldatetime] NOT NULL ,
        [FecElabFin] [smalldatetime] NULL ,
        [FecVen] [smalldatetime] NOT NULL ,
        [TotArts] [int] NOT NULL ,
        [TotCajas] [int] NOT NULL ,
        [TotPzasSurt] [int] NOT NULL ,
        [TotPaqSurtCli] [int] NOT NULL ,
        [TotPaqSurtAlm] [int] NOT NULL ,
        [idProceso] [tinyint] NOT NULL ,
        [PedFac] [bit] NOT NULL ,
```

```
        [IdTipoPed] [int] NULL ,
        [IdTipoPrep] [int] NULL ,
        [Lock] [bit] NOT NULL ,
        [Obser] [text] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PROCESOCOLOR]    Script Date: 31/10/2011 05:51:26 p.m. ******/
CREATE TABLE [dbo].[PROCESOCOLOR] (
        [IdProceso] [tinyint] NOT NULL ,
        [IdColor] [char] (10) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[MSKU]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[MSKU] (
        [NUMCLI] [char] (6) NOT NULL ,
        [CODART] [varchar] (13) NOT NULL ,
        [SKU] [varchar] (20) NOT NULL ,
        [CANEMP] [numeric](3, 0) NULL ,
        [CANEMPD] [numeric](3, 0) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMETAPAS]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMETAPAS] (
        [NumPed] [char] (6) NOT NULL ,
        [IdEtapa] [int] NOT NULL ,
        [FechaEntrega] [datetime] NULL ,
        [IdStatusEtapa] [int] NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMSUC]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMSUC] (
        [NumPed] [char] (6) NOT NULL ,
        [idSuc] [varchar] (6) NOT NULL ,
        [NomSuc] [varchar] (100) NULL ,
        [Obser] [text] NULL ,
        [CajasXSuc] [int] NOT NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMTEMP]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMTEMP] (
        [NUMPED] [char] (6) NOT NULL ,
        [NUMEMP] [char] (6) NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMCAJAS]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMCAJAS] (
        [IdCajaGlob] [int] IDENTITY (1, 1) NOT NULL ,
        [IdCajaPed] [int] NOT NULL ,
        [IdCajaEtapa] [int] NULL ,
        [IdCajaSuc] [int] NOT NULL ,
        [NumPed] [char] (6) NOT NULL ,
        [IdSuc] [varchar] (6) NOT NULL ,
        [IdEtapa] [int] NULL ,
        [NumEmpElab] [varchar] (6) NULL ,
        [FecElab] [datetime] NULL ,
        [TotPzasSurtXCaja] [smallint] NOT NULL ,
        [TotArtXCaja] [smallint] NOT NULL ,
        [TotPaqSurtCli] [smallint] NOT NULL ,
        [TotPaqSurtAlm] [smallint] NOT NULL ,
        [idConten] [varchar] (30) NULL ,
        [CajaStatus] [int] NULL ,
        [Observ] [varchar] (9) NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMDETSOL]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMDETSOL] (
        [NumPed] [char] (6) NOT NULL ,
        [IdSuc] [varchar] (6) NOT NULL ,
        [Consec] [int] NOT NULL ,
        [CodArt] [varchar] (13) NOT NULL ,
        [CantPzasSol] [int] NOT NULL ,
```

```
        [CantPaqSolCli] [int] NULL ,
        [CantPaqSolAlm] [int] NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[CAJAPROCESO]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[CAJAPROCESO] (
        [Numped] [char] (6) NOT NULL ,
        [IdCajaPed] [int] NOT NULL ,
        [idProceso] [tinyint] NOT NULL ,
        [NumEmp] [char] (6) NOT NULL ,
        [FechaIngreso] [smalldatetime] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PEDALMDETSURT]    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE TABLE [dbo].[PEDALMDETSURT] (
        [NumPed] [char] (6) NOT NULL ,
        [IdSuc] [varchar] (6) NOT NULL ,
        [IdCajaPed] [int] NOT NULL ,
        [OrdCli] [varchar] (20) NULL ,
        [Consec] [int] NOT NULL ,
        [CodArt] [varchar] (13) NOT NULL ,
        [CantPzasSurt] [int] NOT NULL ,
        [CantPaqSurtCli] [int] NULL ,
        [CantPaqSurtAlm] [int] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CLIENTESGLOB] WITH NOCHECK ADD
        CONSTRAINT [PK_CLIENTESGLOB] PRIMARY KEY  CLUSTERED
        (
                [NUMCLI]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[DEPARTDUR] WITH NOCHECK ADD
        CONSTRAINT [PK_DEPARTDUR] PRIMARY KEY  CLUSTERED
        (
                [IDDEPDUR]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMART] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMART] PRIMARY KEY  CLUSTERED
        (
                [NumPed],
                [CodArt]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PROCESOS] WITH NOCHECK ADD
        CONSTRAINT [PK_PROCESO] PRIMARY KEY  CLUSTERED
        (
                [IdProceso]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PUÉSTOS] WITH NOCHECK ADD
        CONSTRAINT [PK_PUÉSTOS] PRIMARY KEY  CLUSTERED
        (
                [PUESTO]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[STATUSETAPA] WITH NOCHECK ADD
        CONSTRAINT [PK_STATUSETAPA] PRIMARY KEY  CLUSTERED
        (
                [IdStatusEtapa]
        )  ON [PRIMARY]
GO


ALTER TABLE [dbo].[TIPOPEDIDO] WITH NOCHECK ADD
        CONSTRAINT [PK_TIPOPEDIDO] PRIMARY KEY  CLUSTERED
        (
                [IDTIPOPED]
        )  ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[TIPOPREPARACION] WITH NOCHECK ADD
        CONSTRAINT [PK_TIPOPREPALM] PRIMARY KEY  CLUSTERED
        (
                [IDTIPOPREP]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[UNIDADMEDIDA] WITH NOCHECK ADD
        CONSTRAINT [PK_UNIDADMEDIDA] PRIMARY KEY  CLUSTERED
        (
                [UNIMED]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CATEMPLEADOS] WITH NOCHECK ADD
        CONSTRAINT [PK_CATEMPLEADOS] PRIMARY KEY  CLUSTERED
        (
                [NUMEMP]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CEDIS] WITH NOCHECK ADD
        CONSTRAINT [PK_CEDIS] PRIMARY KEY  CLUSTERED
        (
                [NUMCLI],
                [IDCEDIS]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CLIENTESDET] WITH NOCHECK ADD
        CONSTRAINT [UK_CLIENTESDET] PRIMARY KEY  CLUSTERED
        (
                [NUMCLI],
                [IDSUC]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[DEPARTCLI] WITH NOCHECK ADD
        CONSTRAINT [PK_DEPARTCLI] PRIMARY KEY  CLUSTERED
        (
                [NUMCLI],
                [IDDEPCLI],
                [IDDEPDUR]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[MARTIC] WITH NOCHECK ADD
        CONSTRAINT [PK_MARTIC] PRIMARY KEY  CLUSTERED
        (
                [CODART]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMGLOB] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMGLOB] PRIMARY KEY  CLUSTERED
        (
                [NUMPED]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PROCESOCOLOR] WITH NOCHECK ADD
        CONSTRAINT [PK_ColorStatus] PRIMARY KEY  CLUSTERED
        (
                [IdProceso]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[MSKU] WITH NOCHECK ADD
        CONSTRAINT [PK_MSKU] PRIMARY KEY  CLUSTERED
        (
                [NUMCLI],
                [CODART]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMETAPAS] WITH NOCHECK ADD
```

```
        CONSTRAINT [PK_PEDALMETAPAS] PRIMARY KEY  CLUSTERED
        (
                [NumPed],
                [IdEtapa]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMSUC] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMSUC] PRIMARY KEY  CLUSTERED
        (
                [NumPed],
                [idSuc]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMTEMP] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMTEMP] PRIMARY KEY  CLUSTERED
        (
                [NUMPED],
                [NUMEMP]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMCAJAS] WITH NOCHECK ADD
        CONSTRAINT [UK_PEDALMCAJAS_PED] UNIQUE  CLUSTERED
        (
                [NumPed],
                [IdCajaPed]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMDETSOL] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMDETSOL] PRIMARY KEY  CLUSTERED
        (
                [NumPed],
                [IdSuc],
                [CodArt]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CAJAPROCESO] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMCAJAPROCESO] PRIMARY KEY  CLUSTERED
        (
                [Numped],
                [IdCajaPed],
                [idProceso]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMDETSURT] WITH NOCHECK ADD
        CONSTRAINT [PK_PEDALMDETSURT] PRIMARY KEY  CLUSTERED
        (
                [NumPed],
                [IdSuc],
                [CodArt],
                [IdCajaPed]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CLIENTESGLOB] ADD
        CONSTRAINT [UK_CLIENTESGLOB] UNIQUE  NONCLUSTERED
        (
                [NUMCLI],
                [NOMCLI]
        )  ON [PRIMARY] ,
        CONSTRAINT [CK_CLIENTESGLOB] CHECK (len([NOMCLI]) > 1 and [NUMCLI] like '[A-Z][A-Z][A-Z][0-
9][0-9][0-9]')
GO

 CREATE  INDEX [IX_CLIENTESGLOB] ON [dbo].[CLIENTESGLOB]([NUMCLI]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[DEPARTDUR] ADD
        CONSTRAINT [UK_DEPARTDUR] UNIQUE  NONCLUSTERED
        (
                [IDDEPDUR],
                [DESCRIPCION]
        )  ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[PEDALMART] ADD
        CONSTRAINT [DF_PEDALMART_ArtSurt] DEFAULT (0) FOR [ArtSurt]
GO

ALTER TABLE [dbo].[TIPOMATERIAL] ADD
        CONSTRAINT [UK_TIPOMATERIAL] UNIQUE  NONCLUSTERED
        (
                [TIPMAT],
                [DESCRIPCION]
        )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[CATEMPLEADOS] ADD
        CONSTRAINT [CK_CATEMPLEADOS] CHECK (len([NOMEMP]) > 1 and len([APPAT]) > 1)
GO

 CREATE  UNIQUE  INDEX [UK_CATEMPLEADOS] ON [dbo].[CATEMPLEADOS]([NUMEMP], [NOMEMP]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CLIENTESDET] ADD
        CONSTRAINT [CK_CLIENTESDET] CHECK (len([IDSUC]) > 0 and [NUMCLI] like '[A-Z][A-Z][A-Z][0-1][0-
1][0-1]')
GO

ALTER TABLE [dbo].[DEPARTCLI] ADD
        CONSTRAINT [CK_DEPARTCLI] CHECK ([NUMCLI] like '[A-Z][A-Z][A-Z][0-9][0-9][0-9]' and
len([IDDEPCLI]) > 0 and len([IDDEPDUR]) > 0)
GO

ALTER TABLE [dbo].[MARTIC] ADD
        CONSTRAINT [UK_MARTIC_CODARTP] UNIQUE  NONCLUSTERED
        (
                [CODARTP]
        )  ON [PRIMARY]
GO

 CREATE  INDEX [UK_MARTIC_DESCAR] ON [dbo].[MARTIC]([DESCAR]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMGLOB] ADD
        CONSTRAINT [DF_PEDALMGLOB_FecElabIni] DEFAULT (getdate()) FOR [FecElabIni],
        CONSTRAINT [DF_PEDALMGLOB_TotArts] DEFAULT (0) FOR [TotArts],
        CONSTRAINT [DF_PEDALMGLOB_TotCajas] DEFAULT (0) FOR [TotCajas],
        CONSTRAINT [DF_PEDALMGLOB_TotPzasSurt] DEFAULT (0) FOR [TotPzasSurt],
        CONSTRAINT [DF_PEDALMGLOB_TotPaqSurtCli] DEFAULT (0) FOR [TotPaqSurtCli],
        CONSTRAINT [DF_PEDALMGLOB_TotPaqSurtAlm] DEFAULT (0) FOR [TotPaqSurtAlm],
        CONSTRAINT [DF_PEDALMGLOB_PedFac] DEFAULT (0) FOR [PedFac],
        CONSTRAINT [DF_PEDALMGLOB_Lock] DEFAULT (0) FOR [Lock]
GO

 CREATE  INDEX [PEDALMGLOB14] ON [dbo].[PEDALMGLOB]([NUMPEDP]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PEDALMSUC] ADD
        CONSTRAINT [DF_PEDALMSUC_CajasXSuc] DEFAULT (0) FOR [CajasXSuc]
GO

ALTER TABLE [dbo].[PEDALMCAJAS] ADD
        CONSTRAINT [DF_PEDALMCAJAS_TotPzaSurtXCaja] DEFAULT (0) FOR [TotPzasSurtXCaja],
        CONSTRAINT [DF_PEDALMCAJAS_TotArtXCaja] DEFAULT (0) FOR [TotArtXCaja],
        CONSTRAINT [DF_PEDALMCAJAS_TotPaqSurtCli] DEFAULT (0) FOR [TotPaqSurtCli],
        CONSTRAINT [DF_PEDALMCAJAS_TotPaqSurtAlm] DEFAULT (0) FOR [TotPaqSurtAlm],
        CONSTRAINT [PK_PEDALMCAJAS] PRIMARY KEY  NONCLUSTERED
        (
                [IdCajaGlob]
        )  ON [PRIMARY] ,
        CONSTRAINT [UK_PEDALMCAJAS_SUC] UNIQUE  NONCLUSTERED
        (
                [IdCajaSuc],
                [IdSuc],
                [NumPed]
        )  ON [PRIMARY]
GO

 CREATE  INDEX [PEDALMDETSOL13] ON [dbo].[PEDALMDETSOL]([NumPed]) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[CATEMPLEADOS] ADD
        CONSTRAINT [FK_CATEMPLEADOS-PUESTO_PUÉSTOS-PUESTO] FOREIGN KEY
        (
                [PUESTO]
        ) REFERENCES [dbo].[PUÉSTOS] (
                [PUESTO]
        )
GO

ALTER TABLE [dbo].[CEDIS] ADD
        CONSTRAINT [FK_CEDIS-NUMCLI_CLIENTESGLOB-NUMCLI] FOREIGN KEY
        (
                [NUMCLI]
        ) REFERENCES [dbo].[CLIENTESGLOB] (
                [NUMCLI]
        )
GO

ALTER TABLE [dbo].[CLIENTESDET] ADD
        CONSTRAINT [FK_CLIENTESDET-NUMCLI_CLIENTESGLOB-NUMCLI] FOREIGN KEY
        (
                [NUMCLI]
        ) REFERENCES [dbo].[CLIENTESGLOB] (
                [NUMCLI]
        ) ON DELETE CASCADE  ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[DEPARTCLI] ADD
        CONSTRAINT [FK_DEPARTCLI-IDDEPDUR_DEPARTDUR-IDDEPDUR] FOREIGN KEY
        (
                [IDDEPDUR]
        ) REFERENCES [dbo].[DEPARTDUR] (
                [IDDEPDUR]
        ),
        CONSTRAINT [FK_DEPARTCLI-NUMCLI_CLIENTESGLOB-NUMCLI] FOREIGN KEY
        (
                [NUMCLI]
        ) REFERENCES [dbo].[CLIENTESGLOB] (
                [NUMCLI]
        )
GO

ALTER TABLE [dbo].[MARTIC] ADD
        CONSTRAINT [FK_MARTIC_UNIDADMEDIDA] FOREIGN KEY
        (
                [UNIMED]
        ) REFERENCES [dbo].[UNIDADMEDIDA] (
                [UNIMED]
        )
GO

ALTER TABLE [dbo].[PEDALMGLOB] ADD
        CONSTRAINT [FK_PEDALMGLOB-IdProceso_PROCESOS-IdProceso] FOREIGN KEY
        (
                [idProceso]
        ) REFERENCES [dbo].[PROCESOS] (
                [IdProceso]
        ),
        CONSTRAINT [FK_PEDALMGLOB-IDTIPOPED_TIPOPEDIDO-IDTIPOPED] FOREIGN KEY
        (
                [IdTipoPed]
        ) REFERENCES [dbo].[TIPOPEDIDO] (
                [IDTIPOPED]
        ) ON DELETE CASCADE  ON UPDATE CASCADE ,
        CONSTRAINT [FK_PEDALMGLOB-IDTIPOPREP_TIPOPREPALM-IDTIPOPREP] FOREIGN KEY
        (
                [IdTipoPrep]
        ) REFERENCES [dbo].[TIPOPREPARACION] (
                [IDTIPOPREP]
        ) ON DELETE CASCADE  ON UPDATE CASCADE ,
        CONSTRAINT [FK_PEDALMGLOB-NUMCLI_CLIENTESGLOB-NUMCLI] FOREIGN KEY
        (
                [NumCli]
        ) REFERENCES [dbo].[CLIENTESGLOB] (
                [NUMCLI]
        )
GO
```

```
ALTER TABLE [dbo].[PROCESOCOLOR] ADD
        CONSTRAINT [FK_PROCESOCOLOR-IdProceso_PROCESOS-IdProceso] FOREIGN KEY
        (
                [IdProceso]
        ) REFERENCES [dbo].[PROCESOS] (
                [IdProceso]
        )
GO

ALTER TABLE [dbo].[MSKU] ADD
        CONSTRAINT [FK_MSKU-CODART_MARTIC-CODART] FOREIGN KEY
        (
                [CODART]
        ) REFERENCES [dbo].[MARTIC] (
                [CODART]
        ) ON DELETE CASCADE  ON UPDATE CASCADE ,
        CONSTRAINT [FK_MSKU-NUMCLI_CLIENTESGLOB-NUMCLI] FOREIGN KEY
        (
                [NUMCLI]
        ) REFERENCES [dbo].[CLIENTESGLOB] (
                [NUMCLI]
        )
GO

ALTER TABLE [dbo].[PEDALMETAPAS] ADD
        CONSTRAINT [FK_PEDALMETAPAS-IdStatusEtapa_STATUSETAPA-IdStatusEtapa] FOREIGN KEY
        (
                [IdStatusEtapa]
        ) REFERENCES [dbo].[STATUSETAPA] (
                [IdStatusEtapa]
        ),
        CONSTRAINT [FK_PEDALMETAPAS-NUMPED_PEDALMGLOB-NUMPED] FOREIGN KEY
        (
                [NumPed]
        ) REFERENCES [dbo].[PEDALMGLOB] (
                [NUMPED]
        )
GO

ALTER TABLE [dbo].[PEDALMSUC] ADD
        CONSTRAINT [FK_PEDALMSUC-NUMPED_PEDALMGLOB-NUMPED] FOREIGN KEY
        (
                [NumPed]
        ) REFERENCES [dbo].[PEDALMGLOB] (
                [NUMPED]
        )
GO

ALTER TABLE [dbo].[PEDALMTEMP] ADD
        CONSTRAINT [FK_PEDALMTEMP-NUMEMP_CATEMPLEADOS-NUMEMP] FOREIGN KEY
        (
                [NUMEMP]
        ) REFERENCES [dbo].[CATEMPLEADOS] (
                [NUMEMP]
        ),
        CONSTRAINT [FK_PEDALMTEMP-NUMPED_PEDALMGLOB-NUMPED] FOREIGN KEY
        (
                [NUMPED]
        ) REFERENCES [dbo].[PEDALMGLOB] (
                [NUMPED]
        )
GO

ALTER TABLE [dbo].[PEDALMCAJAS] ADD
        CONSTRAINT [FK_PEDALMCAJAS_PEDALMSUC] FOREIGN KEY
        (
                [NumPed],
                [IdSuc]
        ) REFERENCES [dbo].[PEDALMSUC] (
                [NumPed],
                [idSuc]
        )
GO

ALTER TABLE [dbo].[PEDALMDETSOL] ADD
        CONSTRAINT [FK_PEDALMDETSOL_PEDALMART] FOREIGN KEY
        (
```

```
                [NumPed],
                [CodArt]
        ) REFERENCES [dbo].[PEDALMART] (
                [NumPed],
                [CodArt]
        ),
        CONSTRAINT [FK_PEDALMDETSOL-NumPedidSuc_PEDALMSUC-NumPedidSuc] FOREIGN KEY
        (
                [NumPed],
                [IdSuc]
        ) REFERENCES [dbo].[PEDALMSUC] (
                [NumPed],
                [idSuc]
        )
GO

ALTER TABLE [dbo].[CAJAPROCESO] ADD
        CONSTRAINT [FK_PEDALMCAJAPROCESO_PEDALMCAJAS] FOREIGN KEY
        (
                [Numped],
                [IdCajaPed]
        ) REFERENCES [dbo].[PEDALMCAJAS] (
                [NumPed],
                [IdCajaPed]
        ),
        CONSTRAINT [FK_PEDALMCAJAPROCESO-IdProceso_PROCESO-IdProceso] FOREIGN KEY
        (
                [idProceso]
        ) REFERENCES [dbo].[PROCESOS] (
                [IdProceso]
        )
GO

alter table [dbo].[CAJAPROCESO] nocheck constraint [FK_PEDALMCAJAPROCESO_PEDALMCAJAS]
GO

ALTER TABLE [dbo].[PEDALMDETSURT] ADD
        CONSTRAINT [FK_PEDALMDETSURT_PEDALMART] FOREIGN KEY
        (
                [NumPed],
                [CodArt]
        ) REFERENCES [dbo].[PEDALMART] (
                [NumPed],
                [CodArt]
        ),
        CONSTRAINT [FK_PEDALMDETSURT_PEDALMCAJAS] FOREIGN KEY
        (
                [NumPed],
                [IdCajaPed]
        ) REFERENCES [dbo].[PEDALMCAJAS] (
                [NumPed],
                [IdCajaPed]
        ) ON DELETE CASCADE  ON UPDATE CASCADE ,
        CONSTRAINT [FK_PEDALMDETSURT_PEDALMDETSOL] FOREIGN KEY
        (
                [NumPed],
                [IdSuc],
                [CodArt]
        ) REFERENCES [dbo].[PEDALMDETSOL] (
                [NumPed],
                [IdSuc],
                [CodArt]
        )
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  View dbo.VwPedAlmDet    Script Date: 31/10/2011 05:51:27 p.m. ******/
CREATE VIEW dbo.VwPedAlmDet
AS
SELECT     PDO.NUMPED, PDO.IDSUC, PDO.CONSEC, PDO.CODART, PDO.CODARTP, PDO.SKU, PDO.DESCAR,
           PDO.CANTPZASSOL, PDO.CANTPAQSOLCLI,
           PDO.CANTPAQSOLALM, PDO.NUMPZAXPAQCLI, PDO.NUMPZAXPAQALM,
           SUM(CANTPZASSURT)   AS CANTPZASSURT,
           SUM(CANTPAQSURTCLI) AS CANTPAQSURTCLI,
```

```
           SUM(CANTPAQSURTALM) AS CANTPAQSURTALM,
           PDO.CANTPZASSOL - SUM(CANTPZASSURT) AS FALTANTEPZAS,
           PDO.CANTPAQSOLCLI - SUM(CANTPAQSURTCLI) AS FALTANTEPAQCLI, PDO.CANTPAQSOLALM -
           SUM(CANTPAQSURTALM) AS FALTANTEPAQALM
FROM       PEDALMDETSOL PDO, PEDALMDETSURT PDU
WHERE      PDU.NUMPED =* PDO.NUMPED
AND PDU.IDSUC =* PDO.IDSUC
AND PDU.CODART =* PDO.CODART
GROUP BY PDO.NUMPED, PDO.IDSUC, PDO.CONSEC, PDO.CODART, PDO.CODARTP, PDO.SKU, PDO.DESCAR,
PDO.CANTPZASSOL, PDO.CANTPAQSOLCLI,
                      PDO.CANTPAQSOLALM, PDO.NUMPZAXPAQCLI, PDO.NUMPZAXPAQALM

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  View dbo.VwPedalmCaja    Script Date: 31/10/2011 05:51:27 p.m. ******/
create view VwPedalmCaja as

SELECT     PC.IDCAJAGLOB, PC.IdCajaPed, PC.IdCajaEtapa, PC.IdCajaSuc,  PDO.NUMPED, PDO.IDSUC,
PC.IdEtapa, PC.NumEmpElab, PC.FecElab, PC.idConten,
           PC.CajaStatus, PC.Observ, PDO.CONSEC, PDO.CODART, PDO.CODARTP, PDO.SKU, PDO.DESCAR,
           PDO.CANTPZASSOL,
           PDO.CANTPAQSOLCLI, PDO.CANTPAQSOLALM, PDO.NUMPZAXPAQCLI, PDO.NUMPZAXPAQALM,
           SUM(CANTPZASSURT)                          AS CANTPZASSURT,
           SUM(CANTPAQSURTCLI)                        AS CANTPAQSURTCLI,
           SUM(CANTPAQSURTALM)                        AS CANTPAQSURTALM,
           PDO.CANTPZASSOL - SUM(CANTPZASSURT)        AS FALTANTEPZAS,
           PDO.CANTPAQSOLCLI -   SUM(CANTPAQSURTCLI) AS FALTANTEPAQCLI,
           PDO.CANTPAQSOLALM - SUM(CANTPAQSURTALM)    AS FALTANTEPAQALM
FROM    PEDALMDETSOL PDO, PEDALMDETSURT PDU, PEDALMCAJAS PC
WHERE   PDU.NUMPED =* PDO.NUMPED
AND     PDU.IDSUC =* PDO.IDSUC
AND     PDU.CODART =* PDO.CODART
AND     PC.NUMPED =* PDO.NUMPED
AND     PC.IDSUC =* PDO.IDSUC

GROUP BY PDO.NUMPED, PDO.IDSUC, PDO.CONSEC, PDO.CODART, PDO.CODARTP, PDO.SKU, PDO.DESCAR,
PDO.CANTPZASSOL, PDO.CANTPAQSOLCLI,
PDO.CANTPAQSOLALM, PDO.NUMPZAXPAQCLI, PDO.NUMPZAXPAQALM, PC.IDCAJAGLOB, PC.IdCajaPed, PC.IdCajaEtapa,
PC.IdCajaSuc,
PC.IdEtapa, PC.NumEmpElab, PC.FecElab, PC.idConten, PC.CajaStatus, PC.Observ

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
```

XI. CONSIDERACIONES ACERCA DE OPERACIONES CON VARIAS FILAS PARA TRIGGERS DML

Cuando se escribe el código para un trigger DML, hay que tener en cuenta que la instrucción que hace que se active el disparador puede ser una sola instrucción que afecte a varias filas de datos, en lugar de una sola fila. Este comportamiento es habitual para los tragares UPDATE y DELETE, ya que estas instrucciones suelen afectar a varias filas. No es tan corriente para los tragares INSERT, porque la instrucción INSERT básica sólo agrega una fila. Sin embargo, dado que un trigger INSERT puede ser activado por una instrucción INSERT INTO (table_name) SELECT, la inserción de muchas filas puede tener como resultado la invocación de un solo trigger.

Las consideraciones acerca de los procesos que afectan a varias filas son especialmente importantes cuando la función de un trigger DML consiste en volver a calcular automáticamente los valores de resumen de una tabla y almacenar los resultados en otra para realizar cálculos continuos.

Los tragares DML descritos en los ejemplos siguientes están diseñados para almacenar el total acumulativo de una columna en otra tabla de la base de datos.

Almacenar un total acumulativo para una inserción de una sola fila

La primera versión del trigger DML funcionará correctamente para la inserción de una fila cuando se cargue una fila de datos en la tabla PEDALMCAJAS. El trigger DML se activa mediante una instrucción INSERT y la nueva fila se carga en la tabla inserted mientras dure la ejecución del trigger. La instrucción UPDATE lee el valor de las columnas CantPzasSurt, CantPaqSurtCli y CantPaqSurtAlm para la fila y lo agrega al valor existente en la columna TotPzaSurtXCaja, TotPaqSurtCli, TotPaqSurtAlm de la tabla PEDALMCAJAS. La cláusula WHERE garantiza que la fila actualizada de la tabla PEDALMDETSURT coincida con el NUMPED e IDCAJAPED de la fila de la tabla inserted.

```
- ============================================
-- Autor:                Carlos David Servìn
-- Creado:       23/08/2011
-- Desc:     Suma la cantidad total de artículos en caja
-- ============================================

ALTER TRIGGER  [tI_PEDALMDETSURT]
   ON [dbo].[PEDALMDETSURT]   FOR  INSERT
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
     --evitar resultados adicionales que interfieran con consultas select
        --SET NOCOUNT ON

        UPDATE PEDALMCAJAS SET
                                TotPzaSurtXCaja = TotPzaSurtXCaja + I.CantPzasSurt,
                                TotPaqSurtCli   = TotPaqSurtCli   + I.CantPaqSurtCli,
                                TotPaqSurtAlm   = TotPaqSurtAlm   + I.CantPaqSurtAlm,
                                TotArtXCaja     = TotArtXCaja     + 1
   FROM  PEDALMCAJAS, Inserted I
   WHERE PEDALMCAJAS.NUMPED     = I.NUMPED
   AND   PEDALMCAJAS.IDCAJAPED  = I.IDCAJAPED
END
```

Almacenar un total acumulativo para una inserción de una o varias filas

En caso de una inserción de varias filas, el trigger DML del ejemplo A no funcionaría correctamente; la expresión situada a la derecha de una asignación de una instrucción UPDATE (`TotPzaSurtXCaja + I.CantPzasSurt`) debe tener un solo valor, no una lista de valores. Por lo tanto, el efecto del trigger es recuperar un valor de una sola fila de la tabla inserted y agregarlo al valor `CantPzasSurt` existente en la

tabla PEDALMCAJAS para un valor NUMPED-IDCAJAPED específico. Esta operación puede no tener el efecto esperado si se da un solo valor NUMPED-IDCAJAPED más de una vez en la tabla inserted.

Para actualizar correctamente la tabla PEDALMCAJAS, el trigger debe permitir que pueda haber varias filas en la tabla inserted. Puede hacerlo utilizando la función SUM que calcula el total CantPzasSurt para un grupo de filas de la tabla inserted para cada NumPed. La función SUM se incluye en una sub consulta correlacionada (la instrucción SELECT entre paréntesis). Esta subconsulta devuelve un único valor para cada IdCajaPed de la tabla inserted que coincida o esté correlacionado con un IdCajaPed de la tabla PedAlmCajas.

```
- ============================================
-- Autor:                Carlos David Servìn
-- Creado:       23/08/2011
-- Desc:    Suma la cantidad total de artículos en caja
-- ============================================

ALTER TRIGGER  [tI_PEDALMDETSURT]
   ON [dbo].[PEDALMDETSURT]   FOR  INSERT
AS
BEGIN
--  SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
--evitar resultados adicionales que interfieran con consultas select
--SET NOCOUNT ON

                UPDATE  PC SET
                                PC.TotPzaSurtXCaja = PC.TotPzaSurtXCaja + CantPzasSurt,
                                PC.TotPaqSurtCli  = PC.TotPaqSurtCli   + CantPaqSurtCli,
                                PC.TotPaqSurtAlm  = PC.TotPaqSurtAlm   + CantPaqSurtAlm,
                                PC.TotArtXCaja    = PC.TotArtXCaja     + Cont
                FROM    PEDALMCAJAS PC INNER JOIN
                                (   SELECT   SUM (CantPzasSurt)   As CantPzasSurt,
                                             SUM (CantPaqSurtCli) As CantPaqSurtCli,
                                             SUM (CantPaqSurtAlm) As CantPaqSurtAlm ,
                                             COUNT(1) AS Cont,NUMPED,IDCAJAPED
                                        FROM   Inserted I
                                        GROUP BY
                                                  I.NUMPED, I.IDCAJAPED
                                )
                                  AS  PD
                                  ON  PC.NUMPED     = PD.NUMPED
                                  AND PC.IDCAJAPED  = PD.IDCAJAPED
END
```

Este trigger también funciona correctamente para la inserción de una sola fila; la suma de los valores de la columna LineTotal corresponde a la suma de una sola fila. Sin embargo, con este trigger la subconsulta correlacionada y el operador IN que se utiliza en la cláusula WHERE requieren un procesamiento adicional por parte de SQL Server. Esto no es necesario para una inserción de una sola fila.

Almacenar un total acumulativo basado en el tipo de inserción

Puede cambiar el trigger a fin de utilizar el método óptimo para el número de filas. Por ejemplo, es posible utilizar la función @@ROWCOUNT en la lógica del trigger para que distinga entre la inserción de una fila y la de varias filas.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

-- ============================================
-- Autor:                Carlos David Servìn
-- Creado:               23/08/2011
-- Desc:    Suma la cantidad total de artículos en caja
-- ============================================

ALTER TRIGGER  [tI_PEDALMDETSURT]
   ON [dbo].[PEDALMDETSURT]   FOR  INSERT
```

```
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --            evitar resultados adicionales que interfieran con consultas select
        --SET NOCOUNT ON
IF @@ROWCOUNT = 1
BEGIN
        UPDATE PEDALMCAJAS SET
                                TotPzaSurtXCaja = TotPzaSurtXCaja + I.CantPzasSurt,
                                TotPaqSurtCli   = TotPaqSurtCli   + I.CantPaqSurtCli,
                                TotPaqSurtAlm   = TotPaqSurtAlm   + I.CantPaqSurtAlm,
                                TotArtXCaja     = TotArtXCaja     + 1
   FROM  PEDALMCAJAS, Inserted I
   WHERE PEDALMCAJAS.NUMPED     = I.NUMPED
   AND   PEDALMCAJAS.IDCAJAPED  = I.IDCAJAPED
END

ELSE
        BEGIN
                UPDATE  PC SET
                        PC.TotPzaSurtXCaja = PC.TotPzaSurtXCaja + CantPzasSurt,
                        PC.TotPaqSurtCli   = PC.TotPaqSurtCli   + CantPaqSurtCli,
                        PC.TotPaqSurtAlm   = PC.TotPaqSurtAlm   + CantPaqSurtAlm,
                        PC.TotArtXCaja     = PC.TotArtXCaja     + Cont
                FROM    PEDALMCAJAS PC INNER JOIN
                                (   SELECT SUM (CantPzasSurt)   As CantPzasSurt,
                                                SUM (CantPaqSurtCli) As CantPaqSurtCli,
                                                SUM (CantPaqSurtAlm) As CantPaqSurtAlm ,
                                                COUNT(1) AS Cont,NUMPED,IDCAJAPED
                                        FROM   Inserted I
                                        GROUP BY
                                                I.NUMPED, I.IDCAJAPED
                                )
                                  AS  PD
                                  ON  PC.NUMPED     = PD.NUMPED
                                  AND PC.IDCAJAPED  = PD.IDCAJAPED
        END
END
```

XII. DISPARADORES O TRIGGERS

PEDALMGLOB

Modifica Fila

```
/****** Object:  Trigger dbo.tU_PEDALMGLOB    Script Date: 31/10/2011 05:51:27 p.m. ******/

CREATE TRIGGER [tU_PEDALMGLOB]
   ON  [dbo].[PEDALMGLOB] AFTER  UPDATE
AS
BEGIN
        IF      UPDATE(NUMPEDP) OR
                UPDATE(NUMPED)  OR
                UPDATE(NumCli)  OR
                UPDATE(OrdCli)  OR
                UPDATE(FecIni)  OR
                UPDATE(FecVen)  OR
                UPDATE(TotPzasSurt)  OR
                UPDATE(TotArts)      OR
                UPDATE(TotCajas)     OR
                UPDATE(TotPaqSurtCli)OR
                UPDATE(TotPaqSurtAlm)OR
                UPDATE(IdTipoPed)    OR
                UPDATE(IdTipoPrep)
        BEGIN

              RAISERROR ('No se puede modificar detalles seleccionados de un pedido dato de alta',
              16, 1)
            ROLLBACK TRANSACTION

        END
        IF UPDATE(Lock)
                BEGIN
                                DECLARE @Lock   AS INT
                                DECLARE @NumPed AS CHAR(6)

                                SELECT  @Lock     = D.Lock,
                                                @NumPed   = D.NumPed
                                FROM    DELETED D


                                IF @Lock = 1
                                        BEGIN
                                           RAISERROR ('No se puede modificar. Pedido bloqueado',  16,
                                           1)
                                                ROLLBACK TRANSACTION
                                        END

                            ELSE
                                        BEGIN

                                    --      ALTER TABLE PEDALMCAJAS DISABLE TRIGGER tU_PEDALMCAJAS
                                        UPDATE  PG SET
                                        PG.TotPzasSurt     = PC.TotPzasSurtXCaja,
                                        PG.TotPaqSurtCli   = PC.TotPaqSurtCli,
                                        PG.TotPaqSurtAlm   = PC.TotPaqSurtAlm,
                                        PG.TotCajas        = PC.TotCajas,
                                        PG.TotArts                 = PA.TotArtSurt,
                                        PG.FecElabFin      = GETDATE(),
                                        PG.Lock            = 1
                                        FROM    PEDALMGLOB PG
                                   INNER JOIN
                                              (
                                               SELECT NUMPED,
                                               SUM (TotPzasSurtXCaja) As TotPzasSurtXCaja,
                                               SUM (TotPaqSurtCli)     As TotPaqSurtCli,
                                               SUM (TotPaqSurtAlm)     As TotPaqSurtAlm ,
                                               COUNT(1) AS TotCajas
                                               FROM   PEDALMCAJAS
                                               GROUP  BY
                                               NUMPED
                                              )
                                              AS  PC
                                              ON  PG.NUMPED = PC.NUMPED
```

```
                                    INNER JOIN (
                                             SELECT NUMPED, COUNT (1) AS TotArtSurt
                                                  FROM PEDALMART
                                                  WHERE ARTSURT = 1

                                                    GROUP BY NUMPED) AS  PA
                                    ON  PA.NUMPED = PC.NUMPED

                                      -- ALTER TABLE PEDALMCAJAS ENABLE TRIGGER           tU_PEDALMCAJAS

                                           END

                               END

END


GO
```

PEDALMCAJAS

Insertar Fila

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Trigger dbo.tI_PEDALMCAJAS    Script Date: 31/10/2011 05:51:27 p.m. ******/


-- ============================================
-- Autor:               Carlos David Servìn
-- Creado:              23/08/2011
-- Desc:         Lleva un conteo del número de cajas por sucursal
-- ============================================
CREATE TRIGGER [tI_PEDALMCAJAS]
   ON [dbo].[PEDALMCAJAS]   FOR  INSERT
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --             evitar resultados adicionales que interfieran con consultas select
        --SET NOCOUNT ON
        DECLARE @Cont As INT
        SELECT  @Cont = @@ROWCOUNT
        SET NOCOUNT ON;
                UPDATE PS
                SET   PS.CajasXSuc = PS.CajasXSuc + @Cont
                FROM  PEDALMSUC PS, INSERTED I
                WHERE PS.NUMPED = I.NUMPED
                AND   PS.IDSUC  = I.IDSUC

END



GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Borrar Fila

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Trigger dbo.tD_PEDALMCAJAS    Script Date: 31/10/2011 05:51:27 p.m. ******/
```

```
-- =============================================
-- Autor:                Carlos David Servìn
-- Creado:               23/08/2011
-- Desc:         Lleva un conteo del número de cajas por sucursal
-- =============================================
CREATE TRIGGER [tD_PEDALMCAJAS]
   ON [dbo].[PEDALMCAJAS]   FOR  DELETE
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --             evitar resultados adicionales que interiferan con consultas select
        --SET NOCOUNT ON
        DECLARE @Cont As INT
        SELECT @Cont = @@ROWCOUNT
        SET NOCOUNT ON;

                UPDATE PS
                SET   PS.CajasXSuc = PS.CajasXSuc - @Cont
                FROM  PEDALMSUC PS, DELETED D
                WHERE PS.NUMPED = D.NUMPED
                AND   PS.IDSUC  = D.IDSUC

END



GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Modificar Fila

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go


-- =============================================
-- Autor:                Carlos David Servìn
-- Creado:       23/08/2011
-- Desc:    Impide modificar cantidades de artículos
-- =============================================

ALTER TRIGGER [tU_PEDALMCAJAS]
   ON  [dbo].[PEDALMCAJAS] AFTER  UPDATE
AS
BEGIN

        IF    UPDATE(TotPzaSurtXCaja) OR UPDATE(TotArtXCaja)
           OR UPDATE(TotPaqSurtCli)   OR UPDATE(TotPaqSurtAlm)

        BEGIN
                RAISERROR ('No se puede modificar cantidades de artículos, se necesita borrar caja',
16, 1)
            ROLLBACK TRANSACTION

        END
END


SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

PEDALMDETSURT

Borrar Fila

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
```

```
go
-- =============================================
-- Autor:                Carlos David Servìn
-- Creado:               23/08/2011
-- Desc:    Impide borrar artículos o modificarlos si la caja existe, y actualiza cantidades en caja
-- =============================================

ALTER TRIGGER [tD_PEDALMDETSURT]
   ON  [dbo].[PEDALMDETSURT]   AFTER  DELETE
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --             evitar resultados adicionales que interiferan con consultas select

        SET NOCOUNT ON;
        IF EXISTS (SELECT PC.IdCajaGlob
                        FROM   PEDALMCAJAS PC
                                        INNER JOIN DELETED I
                        ON  PC.NumPed    = I.NumPed
                                                AND PC.IdCajaPed = I.IdCajaPed)
        BEGIN
                RAISERROR ('No se puede modificar artículos individuales, se necesita borrar caja',
16, 1)
            ROLLBACK TRANSACTION
        END
END
```

Insertar Fila

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go
-- =============================================
-- Autor:                Carlos David Servìn
-- Creado:               23/08/2011
-- Desc:    Suma la cantidad total de artículos en caja
-- =============================================

ALTER TRIGGER  [tI_PEDALMDETSURT]
   ON [dbo].[PEDALMDETSURT]   FOR  INSERT
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --             evitar resultados adicionales que interiferan con consultas select
        --SET NOCOUNT ON
IF @@ROWCOUNT = 1
BEGIN
        UPDATE PEDALMCAJAS SET
                                TotPzaSurtXCaja = TotPzaSurtXCaja + I.CantPzasSurt,
                                TotPaqSurtCli   = TotPaqSurtCli   + I.CantPaqSurtCli,
                                TotPaqSurtAlm   = TotPaqSurtAlm   +      I.CantPaqSurtAlm,
                                TotArtXCaja     = TotArtXCaja     + 1
   FROM  PEDALMCAJAS, Inserted I
   WHERE PEDALMCAJAS.NUMPED     = I.NUMPED
   AND   PEDALMCAJAS.IDCAJAPED  = I.IDCAJAPED
END

ELSE
        BEGIN
                UPDATE  PC SET
                                PC.TotPzaSurtXCaja = PC.TotPzaSurtXCaja + CantPzasSurt,
                                PC.TotPaqSurtCli   = PC.TotPaqSurtCli   + CantPaqSurtCli,
                                PC.TotPaqSurtAlm   = PC.TotPaqSurtAlm   + CantPaqSurtAlm,
                                PC.TotArtXCaja     = PC.TotArtXCaja     + Cont
                FROM    PEDALMCAJAS PC INNER JOIN
                                (   SELECT SUM (CantPzasSurt)   As CantPzasSurt,
                                                SUM (CantPaqSurtCli) As CantPaqSurtCli,
                                                SUM (CantPaqSurtAlm) As CantPaqSurtAlm ,
                                                COUNT(1) AS Cont,NUMPED,IDCAJAPED
                                        FROM   Inserted I
                                        GROUP BY
                                                I.NUMPED, I.IDCAJAPED
                                )
```

```
                                    AS  PD
                                    ON  PC.NUMPED     = PD.NUMPED
                                    AND PC.IDCAJAPED  = PD.IDCAJAPED
        END
END
```

Modificar Fila

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

-- ============================================
-- Autor:                Carlos David Servìn
-- Creado:               23/08/2011
-- Desc:    Impide borrar artículos o modificarlos si la caja existe, y actualiza cantidades en caja
-- ============================================

ALTER TRIGGER [tU_PEDALMDETSURT]
   ON  [dbo].[PEDALMDETSURT]   AFTER UPDATE
AS
BEGIN
        -- SET NOCOUNT ON Evita que se devuelva el mensaje que muestra el recuento del número de filas
afectadas por una instrucción o un procedimiento almacenado de Transact-SQL como parte del conjunto de
resultados.
    --             evitar resultados adicionales que interiferan con consultas select

        SET NOCOUNT ON;
        IF EXISTS ( SELECT PC.IdCajaGlob
                        FROM   PEDALMCAJAS PC
                        INNER JOIN INSERTED I ON PC.NumPed = I.NumPed
                        AND PC.IdCajaPed = I.IdCajaPed)
        BEGIN
                RAISERROR ('No se puede modificar artículos individuales, se necesita borrar caja',
16, 1)
            ROLLBACK TRANSACTION
        END
END
```

XIII. INTERPRETACIÓN E ICONOS DE PLAN DE EJECUCIÓN GRÁFICO (SQL SERVER MANAGEMENT STUDIO)

Interpretar la salida del plan de ejecución gráfico

La salida del plan de ejecución gráfico de SQL Server Management Studio se lee de derecha a izquierda y de arriba a abajo. Todas las consultas del lote se analizan y se presentan, incluido el costo de cada consulta como porcentaje del costo total del lote

Cada nodo de la estructura en árbol se representa como un icono que especifica el operador lógico y físico utilizado para ejecutar esa parte de la consulta o instrucción.

Cada nodo está relacionado con un nodo principal. Los nodos secundarios que tienen el mismo nodo principal se dibujan en la misma columna. Sin embargo, no todos los nodos de la misma columna tienen necesariamente el mismo nodo principal. Cada nodo se conecta a su nodo principal mediante reglas con puntas de flecha.

El ancho de la flecha es proporcional al número de filas. Se utiliza el número real de filas cuando está disponible. Si no, se utiliza el número estimado de filas.

Cuando la consulta contiene varias instrucciones, se dibujan varios planes de ejecución de la consulta.

Las partes de las estructuras en árbol se determinan por el tipo de instrucción ejecutada.

Para las consultas en paralelo, en las que intervienen varias CPU, las Propiedades de cada nodo en el plan de ejecución gráfico muestran información acerca de los subprocesos del sistema operativo utilizados. Para ver las propiedades de un nodo, haga clic con el botón secundario en el nodo y, a continuación, haga clic en Propiedades. Para obtener más información acerca de consultas en paralelo, vea Procesar una consulta en paralelo.

Icono	Operador	Icono	Operador	Icono	Operador
	Arithmetic Expression		Insert		Remote Update
	Assert		Inserted Scan		RID Lookup
	Bitmap		Iterator Catchall		Row Count Spool
	Bookmark Lookup		Lazy Spool		Segment
	Clustered Index Delete		Log Row Scan		Sequence
	Clustered Index Insert		Merge Interval		SequenceProject
	Clustered Index Scan		Merge Join		Sort
	Clustered Index Seek		Nested Loops		Split
	Clustered Index Update		Nonclustered Index Delete		Spool
	Collapse		Nonclustered Index Insert		Stream Aggregate
	Compute Scalar		Nonclustered Index Scan		Switch
	Concatenation		Nonclustered Index Seek		Table Delete
	Constant Scan		Nonclustered Index Spool		Table Insert
	Delete		Nonclustered Index Update		Table Scan
	Deleted Scan		Online Index Insert		Table Spool
	Eager Spool		Parameter Table Scan		Table Update
	Filter		Remote Delete		Table-valued Function
	Hash Match		Remote Insert		Top
	Hash Match Root		Remote Query		UDX
	Hash Match Team		Remote Scan		Update

Los siguientes iconos del plan de ejecución gráfico representan a los Operadores de Showplan lógicos y físicos de cursores que utiliza SQL Server para ejecutar instrucciones.

Icono	Operador físico de cursor
	Cursor Catchall
	Dynamic
	Fetch Query
	Keyset
	Population Query
	Refresh Query
	Snapshot

Los siguientes iconos del plan de ejecución gráfico representan a los operadores físicos de Parallelism (operador de Showplan) que utiliza SQL Server para ejecutar instrucciones.

Icono	Operador físico de paralelismo
	Distribute Streams
	Repartition Streams
	Gather Streams

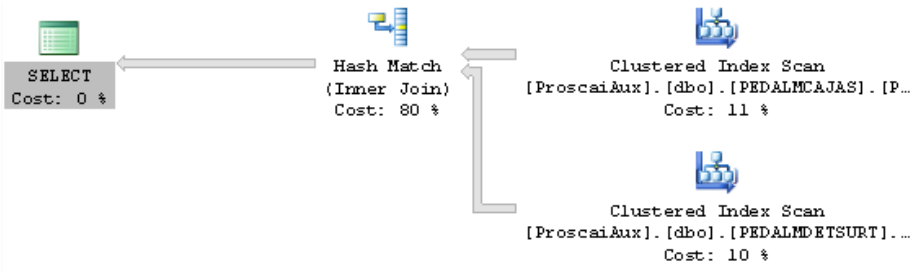
Los siguientes iconos del plan de ejecución gráfico representan a los elementos del lenguaje Transact-SQL que utiliza SQL Server.

Icono	Elemento del lenguaje
	Assign
	Convert
	Declare
	If
	Intrinsic

	Language Element Catchall
	Result
	While

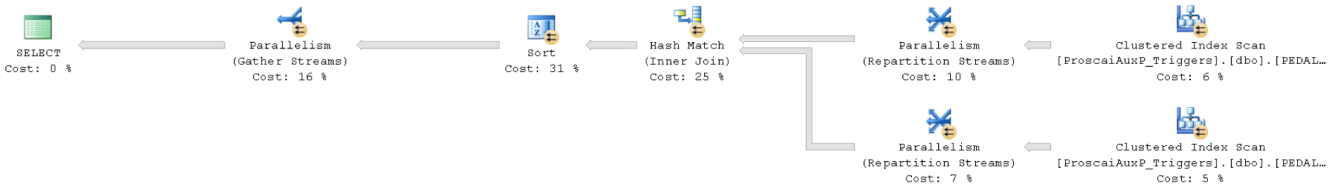
XIV. PLANES DE EJECUCIÓN Y ESTADÍSTICAS DE CONSULTAS

Consulta C-I



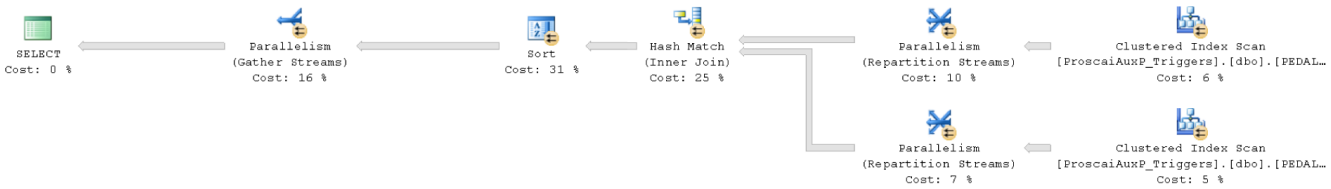

```
-- CAJA SURTIDO

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
```

Consulta C-II




```
-- CAJA SURTIDO  ORDENADO POR PDSU

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
ORDER BY      PDSU.NumPed, PDSU.IdCajaPed
```

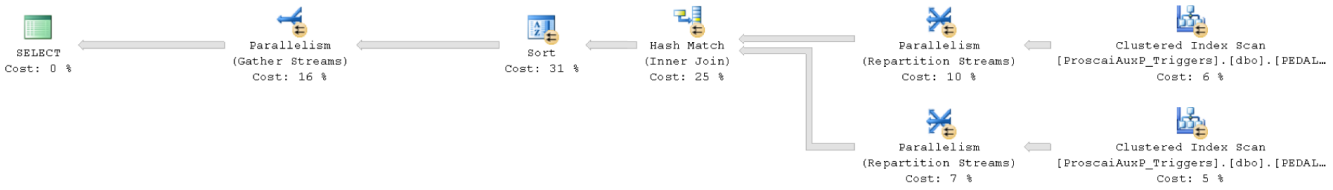
Consulta C-III




```
-- CAJA SURTIDO SURTIDO ORDENADO POR PC

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
```

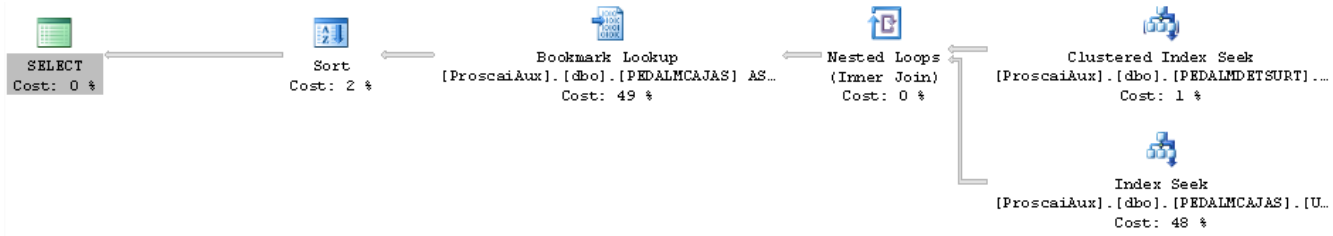
```
WHERE          PDSU.NumPed     = PC.NumPed
AND            PDSU.IdCajaPed = PC.idCajaPed
ORDER BY       PC.NumPed, PC.IdCajaPed
```

Consulta C-IV



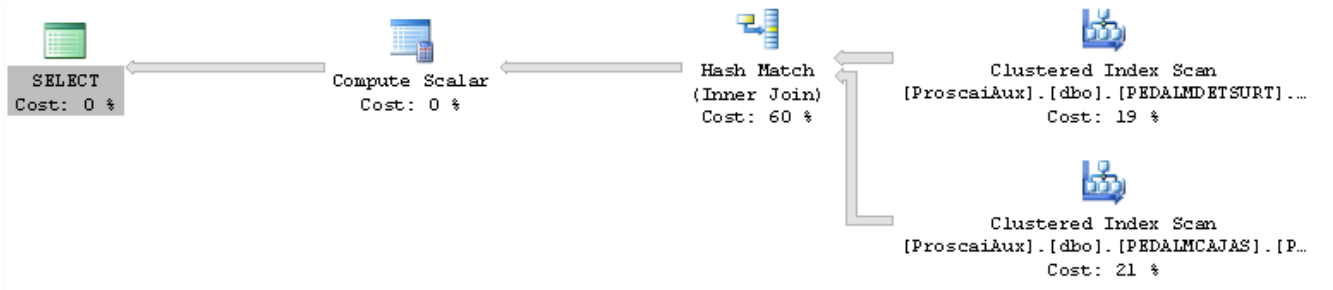

```
-- CAJA SURTIDO SURTIDO ORDENADO POR PDSU, CODIGO ARTICULO

SELECT *
FROM           PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE          PDSU.NumPed     = PC.NumPed
AND            PDSU.IdCajaPed = PC.idCajaPed
ORDER BY       PDSU.NumPed, PDSU.IdCajaPed, PDSU.CODART
```

Consulta C-V




```
--CAJA PEDIDO INDIVIDUAL

SELECT *
FROM           PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE          PDSU.NumPed     = PC.NumPed
AND            PDSU.IdCajaPed = PC.idCajaPed
AND          PC.NUMPED       = 'P50624'
ORDER BY       PC.IdSuc, PC.idCajaSuc, PDSU.CODART
```

Consulta C-VI




```
-- TOTAL ARTÍCULOS SURTIDOS POR CAJA
```

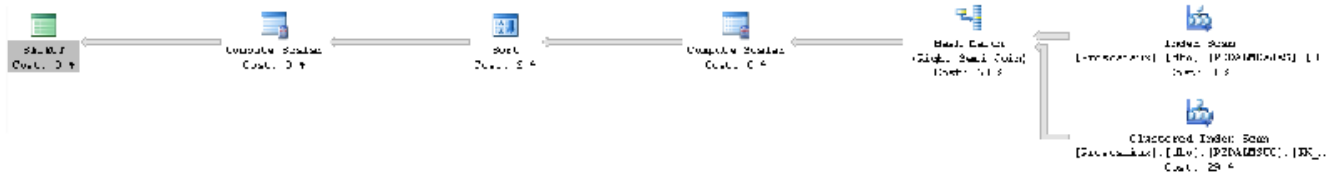
```
SELECT PC.IdCajaGlob, PC.IdCajaPed,  PC.IdCajaEtapa,
              PC.IdCajaSuc,  PC.NumPed,     PC.IdSuc,
              PC.IdEtapa,    PC.NumEmpElab, PC.FecElab,
              PC.idConten,   PC.CajaStatus, PC.Observ,
              SUM(CantPzasSurt)   As TotPzasSurt,
              SUM(CantPaqSurtCli) As TotPaqSurtCli,
              SUM(CantPaqSurtAlm) As TotPaqSurtAlm
FROM   PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PC.NumPed     = PDSU.NumPed
AND           PC.IdCajaPed  = PDSU.IdCajaPed
GROUP BY
              PC.IdCajaGlob, PC.IdCajaPed,  PC.IdCajaEtapa,
              PC.IdCajaSuc,  PC.NumPed,     PC.IdSuc,
              PC.IdEtapa,    PC.NumEmpElab, PC.FecElab,
              PC.idConten,   PC.CajaStatus, PC.Observ
```

Estadísticas							
CONSULTA	**C-I**	**C-II**	**C-III**	**C-IV**	**C-V**	**C-VI**	**Promedio**
Perfil de consulta							
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0
Cantidad de SELECT	3	3	3	3	3	3	3
Filas consultadas por SELECT	99,340	99,344	99,344	99,344	428	24,694	70,416
Número de transacciones	0	0	0	0	0	0	0
Red							
Número de viajes ida y vuelta al servidor	3	3	3	3	3	3	3
Paquetes TDS enviados por cliente	3	3	3	3	3	3	3
Paquetes TDS enviados por servidor	2,571	2,572	2,572	2,572	17	421	1,788
Bytes enviados por cliente	426	534	530	594	596	1,446	688
Bytes recibidos del servidor	10,522,370	10,526,390	10,526,380	10,526,580	59,516	1,716,009	7,312,873
Tiempos (Milisegundos)							
Procesamiento cliente	1,250	1,266	843	1,079	31	282	792
Tiempo total de ejecución	1,468	1,953	1,546	1,875	109	625	1,263
Tiempo de espera respuesta servidor	218	687	703	796	78	343	471

Consulta S-I

```
-- SUCURSALES POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-II

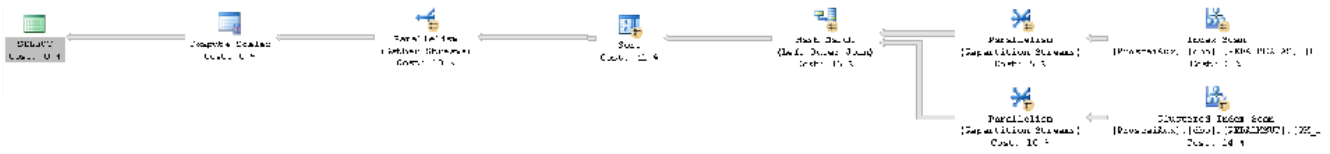


```
-- SUCURSALES SURTIDAS POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
WHERE  PS.IDSUC IN (SELECT IDSUC FROM PEDALMCAJAS WHERE NUMPED = PS.NUMPED)
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-III

```
-- SUCURSALES NO SURTIDAS POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
WHERE  PS.IDSUC NOT IN (SELECT IDSUC FROM PEDALMCAJAS WHERE NUMPED = PS.NUMPED)
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-IV

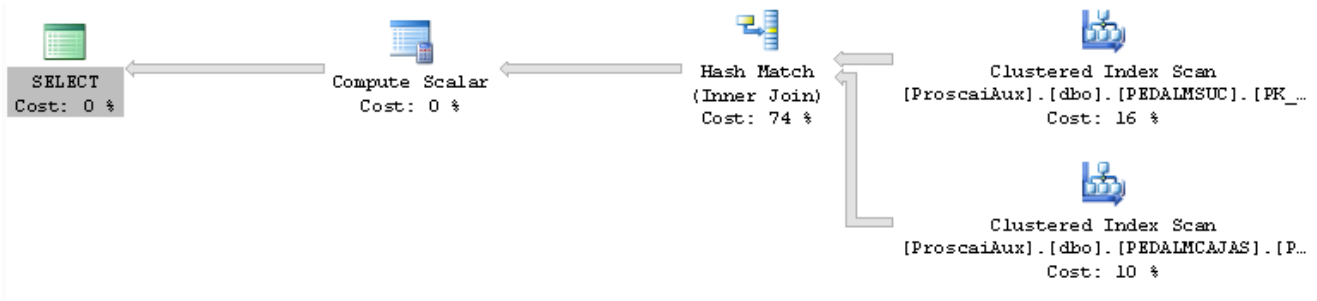


```
SELECT PS.* FROM PEDALMSUC PS, PEDALMCAJAS PC
WHERE PC.NUMPED *= PS.NUMPED
AND   PC.IDSUC  *= PS.IDSUC
ORDER BY PS.IDSUC
```

Consulta S-V

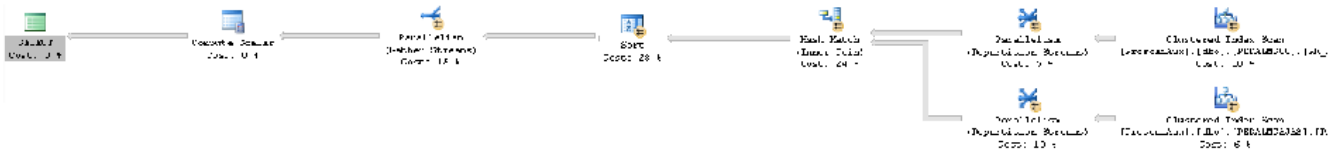
```
-- SUCURSAL  (COTADOR CAJAS)

SELECT PS.NUMPED, COUNT(PC.IDCAJASUC) AS TotalCajasXPedido
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
GROUP BY PS.NUMPED
ORDER BY PS.NUMPED
```

Consulta S-VI




```
-- SUCURSAL Y CAJA (GENERAL)

SELECT *
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
```

Consulta S-VII



```
-- SUCURSAL Y CAJA (ORDENADO POR IDCAJASUC)

SELECT *
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
ORDER BY PS.NUMPED,  PS.IDSUC, PC.IDCAJASUC
```

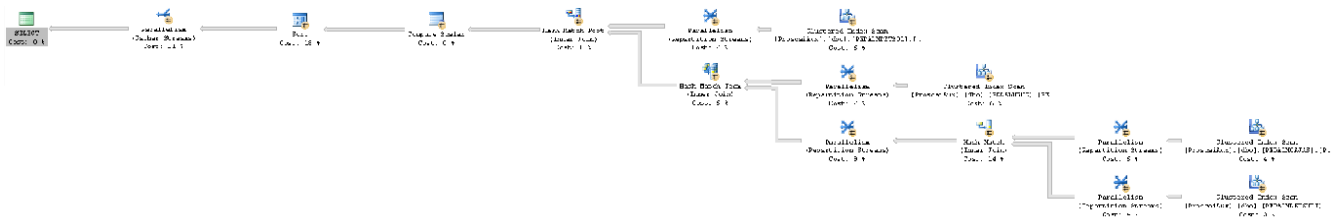
Consulta S-VIII

```
-- SUCURSAL Y CAJA (COTADOR CAJAS)

SELECT PS.NUMPED,PS.IDSUC, COUNT(PC.IDCAJASUC) AS TotalCajasXSucursal
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
GROUP BY PS.NUMPED,  PS.IDSUC
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-IX

```
-- SUCURSAL, CAJA Y PIEZAS SURTIDAS (GENERAL - TotPzasSurtXCaja)

SELECT PS.NUMPED,PC.idCajaPed, PS.IDSUC, PC.IDCAJASUC,
          SUM(CantPzasSurt)   As TotPzasSurtXCaja,
          SUM(CantPaqSurtCli) As TotPaqSurtClitXCaja,
          SUM(CantPaqSurtAlm) As TotPaqSurtAlmtXCaja
FROM   PEDALMSUC PS, PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
AND    PDSU.NumPed    = PC.NumPed
AND       PDSU.IdCajaPed = PC.idCajaPed
GROUP BY PS.NUMPED,PC.idCajaPed, PS.IDSUC, PC.IDCAJASUC
ORDER BY PS.NUMPED,  CONVERT(INT, PS.IDSUC), PC.IDCAJASUC
```

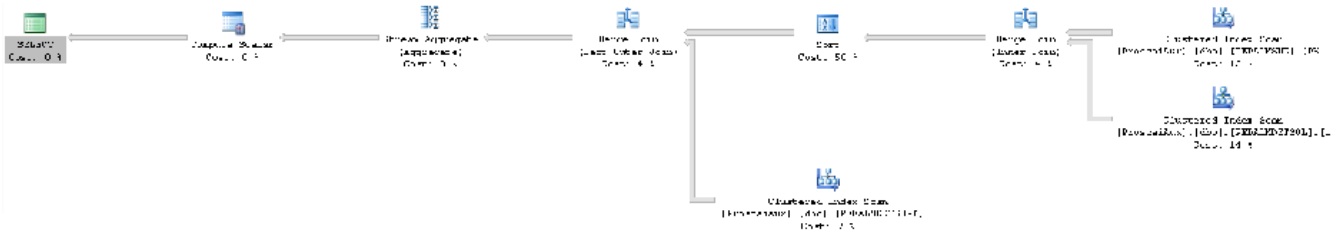
Consulta S-X



```
-- SUCURSAL, CAJA Y PIEZAS SURTIDAS (DETALLE)

SELECT PS.NUMPED, PC.*,PDSO.DESCAR, PDSU.*
```

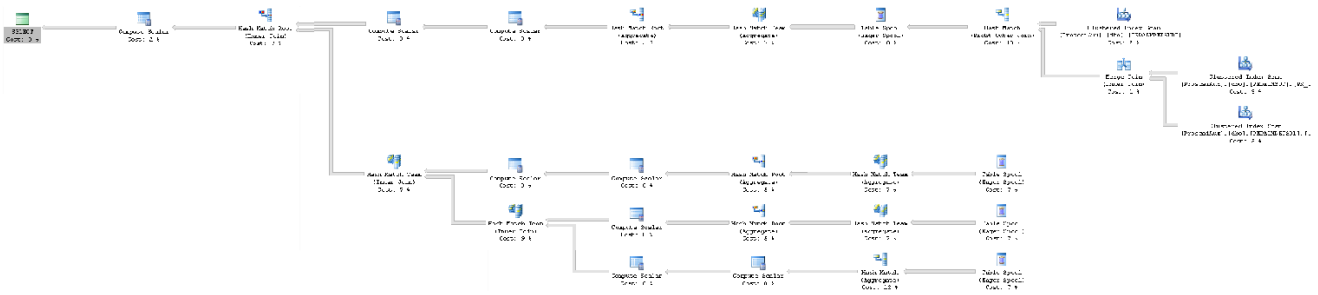
```
FROM    PEDALMSUC PS, PEDALMCAJAS PC,
          PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE    PS.NUMPED       = PC.NUMPED
AND      PS.IDSUC        = PC.IDSUC
AND      PDSU.NumPed     = PC.NumPed
AND        PDSU.IdCajaPed = PC.idCajaPed
AND           PDSO.CodArt     = PDSU.CodArt
AND           PS.NumPed       =  PDSO.NumPed
AND           PS.IdSuc        =  PDSO.IdSuc
ORDER BY PS.NUMPED,  CONVERT(INT, PC.IDSUC), PC.IDCAJASUC, PDSU.CODART
```

Consulta S-XI



```
-- SUCURSAL,  PIEZAS SOLICITADAS Y SURTIDAS

SELECT PS.NumPed, PS.idSuc, PS.NomSuc,
             PDSO.CODART,
             SUM(PDSO.CantPzasSol)    As TotPzasSol,
             SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
             SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
             SUM(CantPzasSurt)        As TotPzasSurt,
             SUM(CantPaqSurtCli)      As TotPaqSurtCli,
             SUM(CantPaqSurtAlm)      As TotPaqSurtAlm

FROM   PEDALMSUC PS, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PS.NumPed      *= PDSU.NumPed
AND           PS.IdSuc       *= PDSU.IdSuc
AND           PDSO.CodArt    *= PDSU.CodArt
AND           PS.NumPed      =  PDSO.NumPed
AND           PS.IdSuc       =  PDSO.IdSuc

GROUP BY
             PS.NumPed, PS.idSuc, PS.NomSuc, PDSO.CODART
```

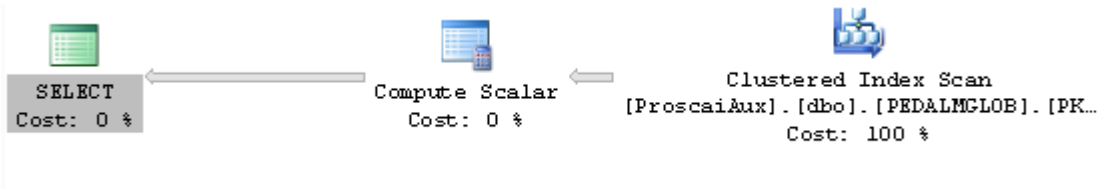
Consulta S-XII



```
-- SUCURSAL,  PIEZAS SOLICITADAS Y SURTIDAS (GENERAL)

SELECT PS.NumPed, PS.idSuc, PS.NomSuc,

              SUM(DISTINCT(PDSO.CantPzasSol))   As TotPzasSol,
              SUM(DISTINCT(PDSO.CantPaqSolCli)) As TotPaqSolCli,
              SUM(DISTINCT(PDSO.CantPaqSolAlm)) As TotPaqSolAlm,
              SUM(CantPzasSurt)       As TotPzasSurt,
              SUM(CantPaqSurtCli)     As TotPaqSurtCli,
              SUM(CantPaqSurtAlm)     As TotPaqSurtAlm

FROM   PEDALMSUC PS, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PS.NumPed             *= PDSU.NumPed
AND            PS.IdSuc      *= PDSU.IdSuc
AND            PDSO.CodArt   *= PDSU.CodArt
AND            PS.NumPed      = PDSO.NumPed
AND            PS.IdSuc       = PDSO.IdSuc
GROUP BY
              PS.NumPed, PS.idSuc, PS.NomSuc, PDSO.CODART
```

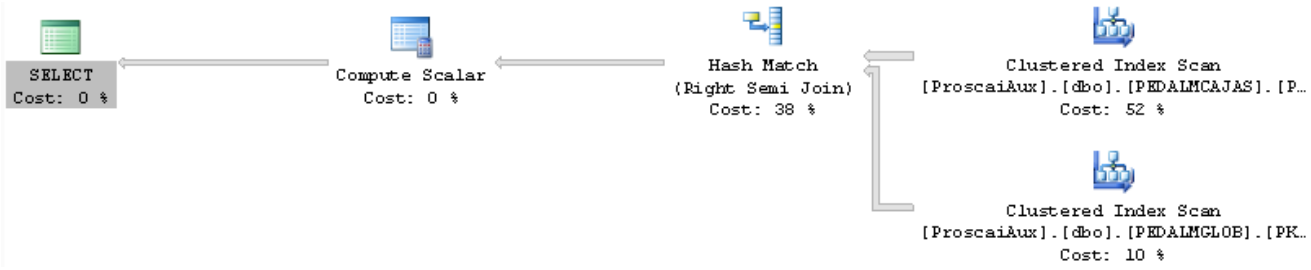
CONSULTA	S-I	S-II	S-III	S-IV	S-V	S-VI	S-VII	S-VIII	S-IX	S-X	S-XI	S-XII
Perfil de consulta												
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0	0	0	0	0	0
Cantidad de SELECT	1	1	1	1	1	1	1	1	1	1	1	1
Filas consultadas por SELECT	83,452	83,451	83,452	83,451	1	92,053	4,827	92,053	92,053	83,451	24,689	99,336
Número de transacciones	0	0	0	0	0	0	0	0	0	0	0	0
Red												
Número de viajes ida y vuelta al servidor	1	1	1	1	1	1	1	1	1	1	1	1
Paquetes TDS enviados por cliente	1	1	1	1	1	1	1	1	1	1	1	1
Paquetes TDS enviados por servidor	1,172	1,172	1,172	1,172	1	1,296	17	2,388	2,388	392	224	3,403
Bytes enviados por cliente	266	380	266	380	394	254	464	280	396	568	1,154	998
Bytes recibidos del servidor	4,799,537	4,799,478	4,799,537	4,799,478	204	5,308,241	67,797	9,780,279	9,780,288	1,604,763	914,371	13,937,550
Tiempos (Milisegundos)												
Procesamiento cliente	391	360	219	453	16	500	16	969	907	312	94	1,657
Tiempo total de ejecución	562	1,250	390	1,296	453	1,156	484	1,156	1,750	2,046	1,062	9,578
Tiempo de espera respuesta servidor	171	890	171	843	437	656	468	187	843	1,734	968	7,921

Consulta P-I




```
----  PEDIDO

SELECT *
FROM   PEDALMGLOB PG
ORDER BY PG.NUMPED
```

Consulta P-II




```
-- PEDIDO SURTIDO

SELECT *
FROM   PEDALMGLOB PG
WHERE  NUMPED IN (SELECT NUMPED FROM PEDALMCAJAS WHERE NUMPED = PG.NUMPED AND
CAJASTATUS <> 1 )
```

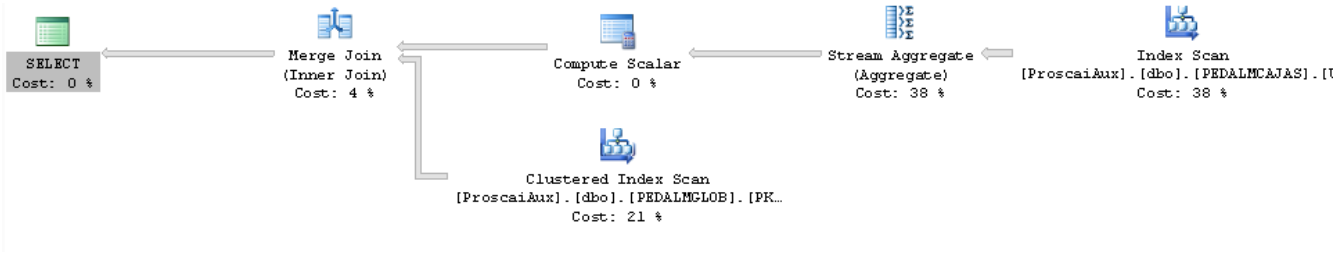
Consulta P-III


```
-- PEDIDO NO SURTIDO

SELECT *
FROM   PEDALMGLOB PG
WHERE  NUMPED NOT IN (SELECT NUMPED FROM PEDALMCAJAS WHERE NUMPED = PG.NUMPED )
```

Consulta P-IV

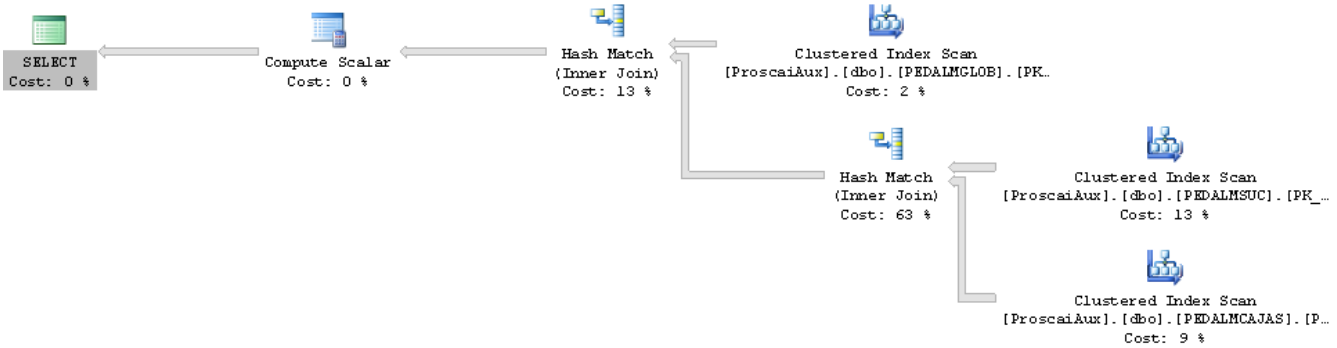
```
-- PEDIDO (CONTADOR DE SUCURSALES)

SELECT   PG.NUMPED, COUNT(DISTINCT (PC.IDSUC)) AS TotSucSurt
FROM     PEDALMGLOB PG, PEDALMCAJAS PC
WHERE    PG.NUMPED    = PC.NUMPED
GROUP BY PG.NUMPED
ORDER BY PG.NUMPED
```

Consulta P-V



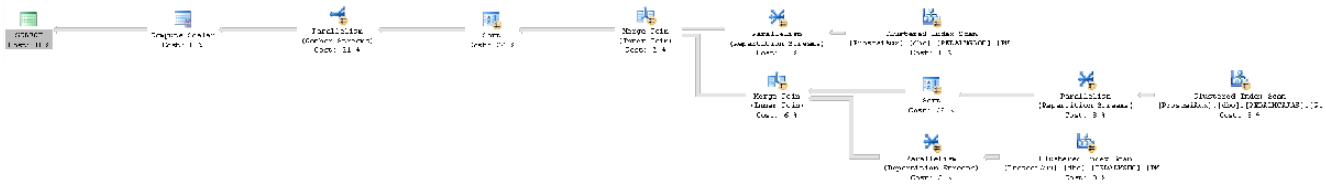

```
-- PEDIDO (CONTADOR DE CAJAS)

SELECT PG.NUMPED, COUNT(*) AS TotCajasSurt
FROM   PEDALMGLOB PG, PEDALMCAJAS PC
WHERE  PG.NUMPED   = PC.NUMPED
GROUP BY PG.NUMPED
ORDER BY PG.NUMPED
```

Consulta P-VI




```
-- PEDIDO, SUCURSAL Y CAJA (GENERAL)

SELECT *
FROM   PEDALMGLOB PG, PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PG.NUMPED = PC.NUMPED
AND    PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
```

Consulta P-VII

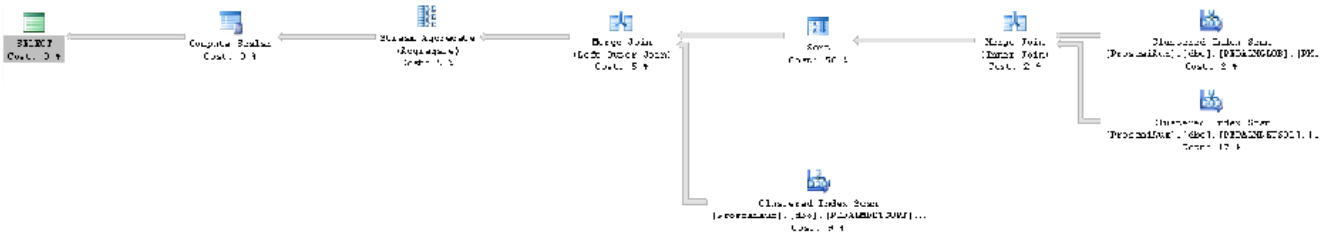


```
-- PEDIDO, SUCURSAL Y CAJA (ORDENADO POR IDCAJASUC)

SELECT *
FROM   PEDALMGLOB PG, PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PG.NUMPED = PC.NUMPED
AND    PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
ORDER BY PG.NUMPED,  PS.IDSUC, PC.IDCAJASUC
```

Consulta P-VIII

```
--PEDIDO, PIEZAS SURTIDAS (GENERAL - TotPzasSurtXCaja)

SELECT PG.NUMPED,PC.idCajaPed, PC.IDSUC, PC.IDCAJASUC,
          SUM(CantPzasSurt)   As TotPzasSurtXCaja,
          SUM(CantPaqSurtCli) As TotPaqSurtClitXCaja,
          SUM(CantPaqSurtAlm) As TotPaqSurtAlmtXCaja
FROM   PEDALMGLOB PG, PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PG.NUMPED      = PC.NUMPED
AND    PDSU.NumPed    = PC.NumPed
AND     PDSU.IdCajaPed = PC.idCajaPed
GROUP BY PG.NUMPED,PC.idCajaPed, PC.IDSUC, PC.IDCAJASUC
ORDER BY PG.NUMPED,  CONVERT(INT, PC.IDSUC), PC.IDCAJASUC
```

Consulta P-IX

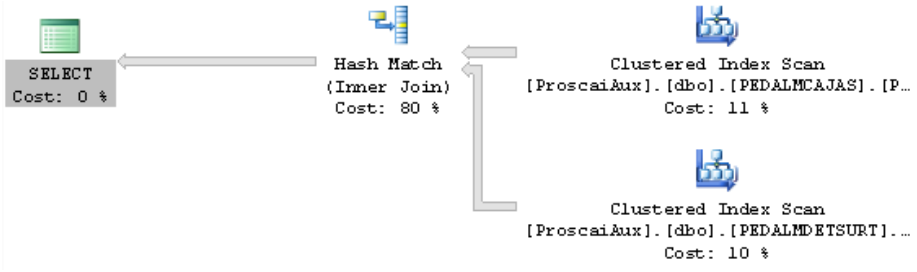


```
--PEDIDO,  PIEZAS SOLICITADAS Y SURTIDAS
```

```
SELECT PG.NumPed,
              PDSO.CODART,PDSO.DESCAR,
              SUM(PDSO.CantPzasSol)    As TotPzasSol,
              SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
              SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
              SUM(CantPzasSurt)        As TotPzasSurt,
              SUM(CantPaqSurtCli)      As TotPaqSurtCli,
              SUM(CantPaqSurtAlm)      As TotPaqSurtAlm

FROM   PEDALMGLOB PG, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PG.NumPed      *= PDSU.NumPed
AND      PDSO.IDSUC       *= PDSU.IDSUC
AND            PDSO.CodArt    *= PDSU.CodArt
AND            PG.NumPed      =  PDSO.NumPed
GROUP BY
              PG.NumPed, PDSO.idSuc, PDSO.CODART,PDSO.DESCAR
```

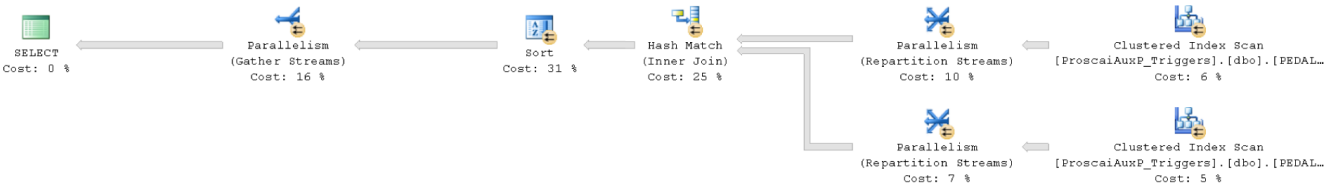
CONSULTA	I	II	III	IV	V	VI	VII	VII	IX	PROMEDIO
Perfil de consulta										
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0	0	0	0
Cantidad de SELECT	3	3	3	3	3	3	3	3	3	3
Filas consultadas por SELECT	4,831	1,732	6	4,835	4,833	92,060	92,066	24,704	67,251	32,480
Número de transacciones	0	0	0	0	0	0	0	0	0	0
Red										
Número de viajes ida y vuelta al servidor	3	3	3	3	3	3	3	3	3	3
Paquetes TDS enviados por cliente	3	3	3	3	3	3	3	3	3	3
Paquetes TDS enviados por servidor	143	50	4	20	20	5,190	5,191	229	1,299	1,350
Bytes enviados por cliente	314	420	394	546	492	504	620	1,198	1,360	650
Bytes recibidos del servidor	574,590	195,182	5,168	72,476	71,416	21,246,630	21,252,720	928,128	5,312,016	5,517,592
Tiempos (Milisegundos)										
Procesamiento cliente	31	47	0	0	265	1,266	1,672	125	828	470
Tiempo total de ejecución	62	109	125	2,500	343	1,562	5,187	953	1,453	1,366
Tiempo de espera respuesta servidor	31	62	125	2,500	78	296	3,515	828	625	896

Consulta C-I



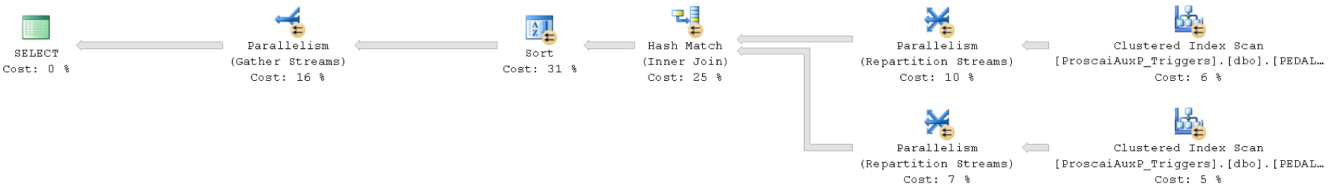

```
-- CAJA SURTIDO

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
```

Consulta C-II



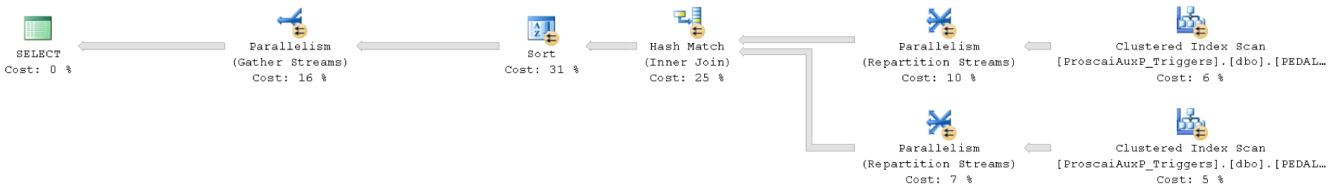

```
-- CAJA SURTIDO  ORDENADO POR PDSU

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
ORDER BY      PDSU.NumPed, PDSU.IdCajaPed
```

Consulta C-III



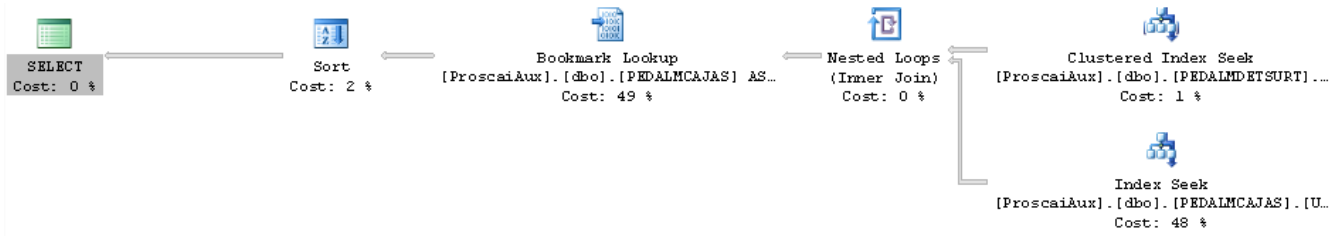

```
-- CAJA SURTIDO ORDENADO POR PC

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
ORDER BY      PC.NumPed, PC.IdCajaPed
```

Consulta C-IV



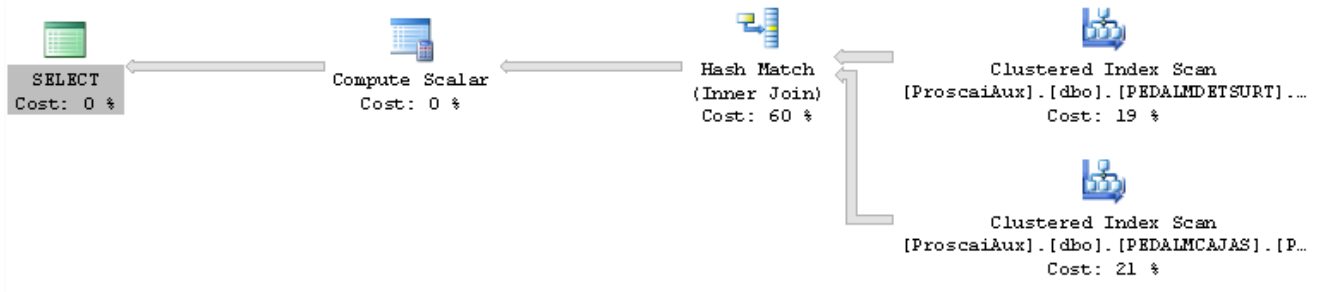

```
-- CAJA SURTIDO ORDENADO POR PDSU, CODIGO ARTICULO

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
ORDER BY      PDSU.NumPed, PDSU.IdCajaPed, PDSU.CODART
```

Consulta C-V




```
--CAJA PEDIDO INDIVIDUAL

SELECT *
FROM          PEDALMDETSURT  PDSU,PEDALMCAJAS PC
WHERE         PDSU.NumPed    = PC.NumPed
AND           PDSU.IdCajaPed = PC.idCajaPed
AND          PC.NUMPED       = 'P50624'
ORDER BY       PC.IdSuc, PC.idCajaSuc, PDSU.CODART
```

Consulta C-VI




```
-- TOTAL ARTÍCULOS SURTIDOS POR CAJA

SELECT PC.IdCajaGlob, PC.IdCajaPed,  PC.IdCajaEtapa,
             PC.IdCajaSuc,  PC.NumPed,     PC.IdSuc,
             PC.IdEtapa,    PC.NumEmpElab, PC.FecElab,
             PC.idConten,   PC.CajaStatus, PC.Observ,
             SUM(CantPzasSurt)   As TotPzasSurt,
             SUM(CantPaqSurtCli) As TotPaqSurtCli,
```

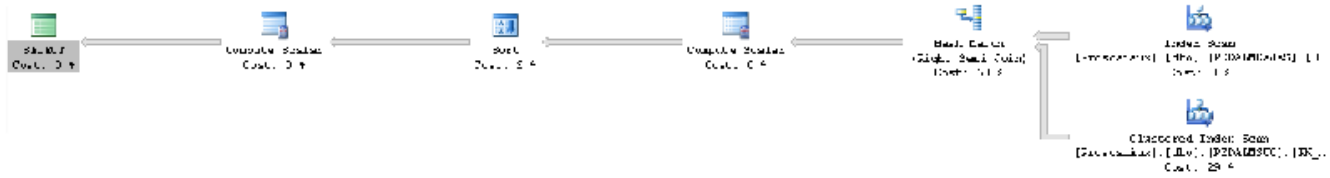
```
            SUM(CantPaqSurtAlm) As TotPaqSurtAlm
FROM   PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PC.NumPed     = PDSU.NumPed
AND            PC.IdCajaPed  = PDSU.IdCajaPed
GROUP BY
               PC.IdCajaGlob, PC.IdCajaPed,  PC.IdCajaEtapa,
               PC.IdCajaSuc,  PC.NumPed,     PC.IdSuc,
               PC.IdEtapa,    PC.NumEmpElab, PC.FecElab,
               PC.idConten,   PC.CajaStatus, PC.Observ
```

Estadísticas							
CONSULTA	**C-I**	**C-II**	**C-III**	**C-IV**	**C-V**	**C-VI**	**Promedio**
Perfil de consulta							
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0
Cantidad de SELECT	3	3	3	3	3	3	3
Filas consultadas por SELECT	99,340	99,344	99,344	99,344	428	24,694	70,416
Número de transacciones	0	0	0	0	0	0	0
Red							
Número de viajes ida y vuelta al servidor	3	3	3	3	3	3	3
Paquetes TDS enviados por cliente	3	3	3	3	3	3	3
Paquetes TDS enviados por servidor	2,571	2,572	2,572	2,572	17	421	1,788
Bytes enviados por cliente	426	534	530	594	596	1,446	688
Bytes recibidos del servidor	10,522,370	10,526,390	10,526,380	10,526,580	59,516	1,716,009	7,312,873
Tiempos (Milisegundos)							
Procesamiento cliente	1,250	1,266	843	1,079	31	282	792
Tiempo total de ejecución	1,468	1,953	1,546	1,875	109	625	1,263
Tiempo de espera respuesta servidor	218	687	703	796	78	343	471

Consulta S-I

```
-- SUCURSALES POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-II

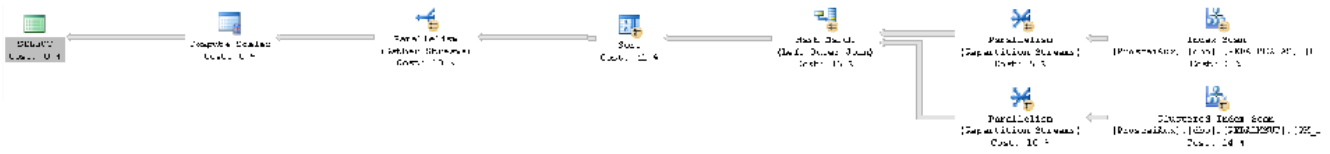


```
-- SUCURSALES SURTIDAS POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
WHERE  PS.IDSUC IN (SELECT IDSUC FROM PEDALMCAJAS WHERE NUMPED = PS.NUMPED)
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-III

```
-- SUCURSALES NO SURTIDAS POR PEDIDO

SELECT *
FROM   PEDALMSUC PS
WHERE  PS.IDSUC NOT IN (SELECT IDSUC FROM PEDALMCAJAS WHERE NUMPED = PS.NUMPED)
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-IV

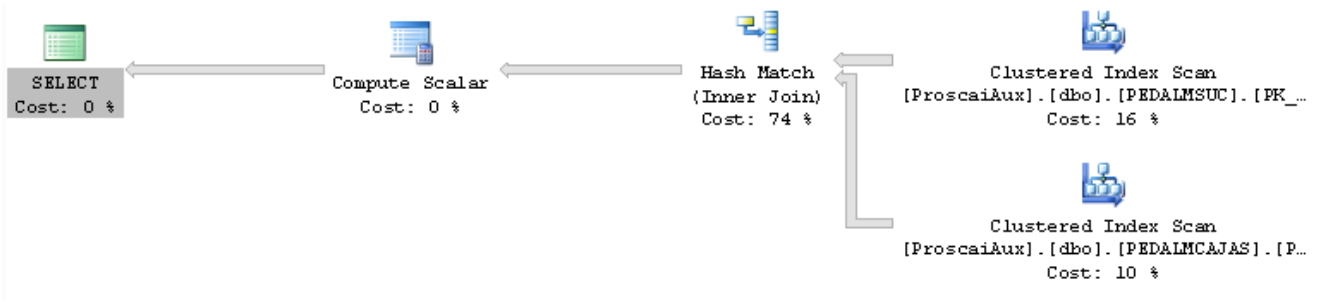


```
SELECT PS.* FROM PEDALMSUC PS, PEDALMCAJAS PC
WHERE PC.NUMPED *= PS.NUMPED
AND   PC.IDSUC  *= PS.IDSUC
ORDER BY PS.IDSUC
```

Consulta S-V

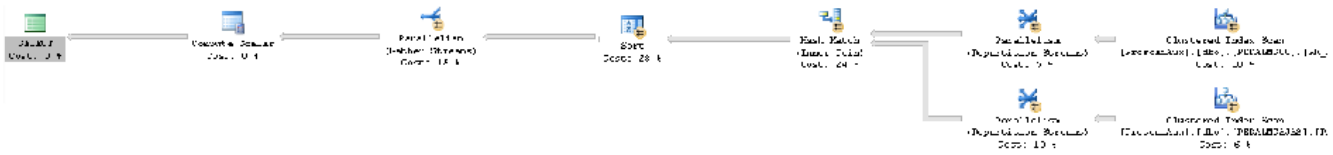
```
-- SUCURSAL  (COTADOR CAJAS)

SELECT PS.NUMPED, COUNT(PC.IDCAJASUC) AS TotalCajasXPedido
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
GROUP BY PS.NUMPED
ORDER BY PS.NUMPED
```

Consulta S-VI




```
-- SUCURSAL Y CAJA (GENERAL)

SELECT *
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
```

Consulta S-VII



```
-- SUCURSAL Y CAJA (ORDENADO POR IDCAJASUC)

SELECT *
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
ORDER BY PS.NUMPED,  PS.IDSUC, PC.IDCAJASUC
```

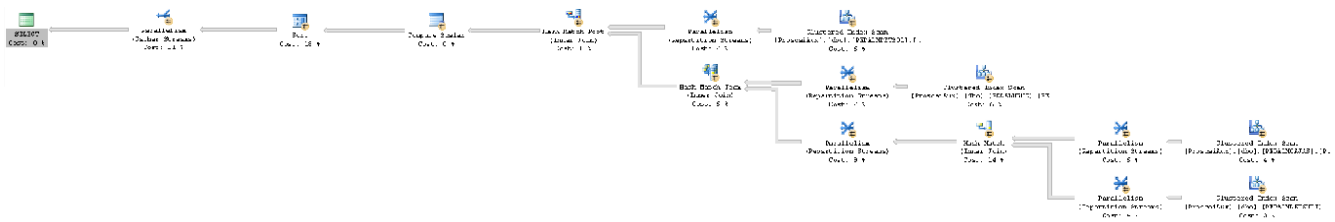
Consulta S-VIII

```
-- SUCURSAL Y CAJA (COTADOR CAJAS)

SELECT PS.NUMPED,PS.IDSUC, COUNT(PC.IDCAJASUC) AS TotalCajasXSucursal
FROM   PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
GROUP BY PS.NUMPED,  PS.IDSUC
ORDER BY PS.NUMPED, CONVERT(INT, PS.IDSUC)
```

Consulta S-IX

```
-- SUCURSAL, CAJA Y PIEZAS SURTIDAS (GENERAL - TotPzasSurtXCaja)

SELECT PS.NUMPED,PC.idCajaPed, PS.IDSUC, PC.IDCAJASUC,
          SUM(CantPzasSurt)   As TotPzasSurtXCaja,
          SUM(CantPaqSurtCli) As TotPaqSurtClitXCaja,
          SUM(CantPaqSurtAlm) As TotPaqSurtAlmtXCaja
FROM   PEDALMSUC PS, PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PS.NUMPED   = PC.NUMPED
AND    PS.IDSUC    = PC.IDSUC
AND    PDSU.NumPed    = PC.NumPed
AND       PDSU.IdCajaPed = PC.idCajaPed
GROUP BY PS.NUMPED,PC.idCajaPed, PS.IDSUC, PC.IDCAJASUC
ORDER BY PS.NUMPED,  CONVERT(INT, PS.IDSUC), PC.IDCAJASUC
```

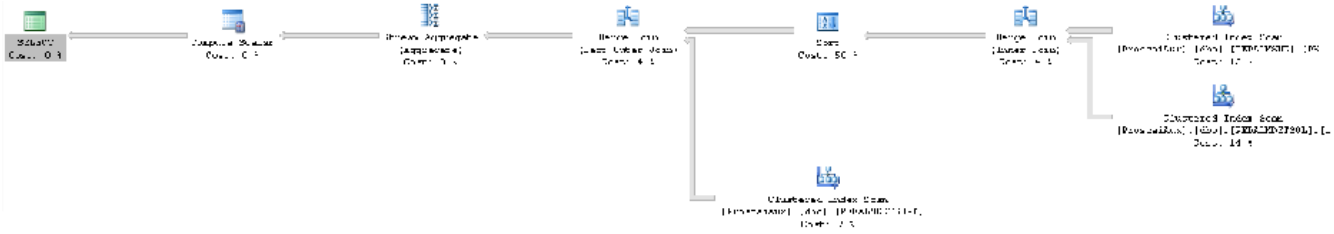
Consulta S-X



```
-- SUCURSAL, CAJA Y PIEZAS SURTIDAS (DETALLE)

SELECT PS.NUMPED, PC.*,PDSO.DESCAR, PDSU.*
FROM   PEDALMSUC PS, PEDALMCAJAS PC,
          PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE    PS.NUMPED      = PC.NUMPED
```

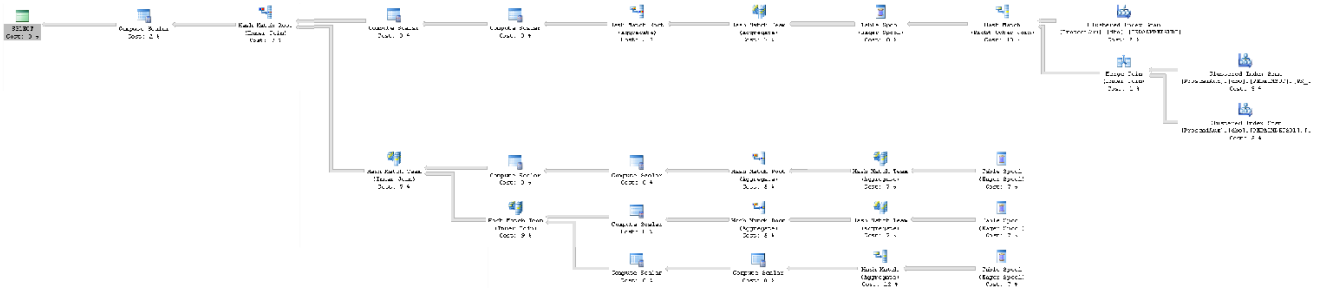
```
AND      PS.IDSUC        = PC.IDSUC
AND      PDSU.NumPed     = PC.NumPed
AND         PDSU.IdCajaPed = PC.idCajaPed
AND            PDSO.CodArt    = PDSU.CodArt
AND            PS.NumPed      =  PDSO.NumPed
AND            PS.IdSuc       =  PDSO.IdSuc
ORDER BY PS.NUMPED,  CONVERT(INT, PC.IDSUC), PC.IDCAJASUC, PDSU.CODART
```

Consulta S-XI



```
-- SUCURSAL,  PIEZAS SOLICITADAS Y SURTIDAS

SELECT PS.NumPed, PS.idSuc, PS.NomSuc,
               PDSO.CODART,
               SUM(PDSO.CantPzasSol)    As TotPzasSol,
               SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
               SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
               SUM(CantPzasSurt)        As TotPzasSurt,
               SUM(CantPaqSurtCli)      As TotPaqSurtCli,
               SUM(CantPaqSurtAlm)      As TotPaqSurtAlm

FROM   PEDALMSUC PS, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PS.NumPed      *= PDSU.NumPed
AND            PS.IdSuc       *= PDSU.IdSuc
AND            PDSO.CodArt    *= PDSU.CodArt
AND            PS.NumPed      =  PDSO.NumPed
AND            PS.IdSuc       =  PDSO.IdSuc

GROUP BY
               PS.NumPed, PS.idSuc, PS.NomSuc, PDSO.CODART
```

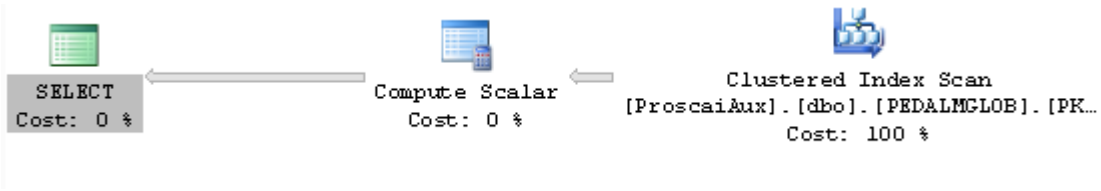
Consulta S-XII



```
-- SUCURSAL,  PIEZAS SOLICITADAS Y SURTIDAS (GENERAL)

SELECT PS.NumPed, PS.idSuc, PS.NomSuc,
            SUM(DISTINCT(PDSO.CantPzasSol))   As TotPzasSol,
            SUM(DISTINCT(PDSO.CantPaqSolCli)) As TotPaqSolCli,
            SUM(DISTINCT(PDSO.CantPaqSolAlm)) As TotPaqSolAlm,
            SUM(CantPzasSurt)       As TotPzasSurt,
            SUM(CantPaqSurtCli)     As TotPaqSurtCli,
            SUM(CantPaqSurtAlm)     As TotPaqSurtAlm
FROM   PEDALMSUC PS, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PS.NumPed     *= PDSU.NumPed
AND            PS.IdSuc      *= PDSU.IdSuc
AND            PDSO.CodArt   *= PDSU.CodArt
AND            PS.NumPed     =  PDSO.NumPed
AND            PS.IdSuc      =  PDSO.IdSuc
GROUP BY
            PS.NumPed, PS.idSuc, PS.NomSuc, PDSO.CODART
```

CONSULTA	S-I	S-II	S-III	S-IV	S-V	S-VI	S-VII	S-VIII	S-IX	S-X	S-XI	S-XII
Perfil de consulta												
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0	0	0	0	0	0
Cantidad de SELECT	1	1	1	1	1	1	1	1	1	1	1	1
Filas consultadas por SELECT	83,452	83,451	83,452	83,451	1	92,053	4,827	92,053	92,053	83,451	24,689	99,336
Número de transacciones	0	0	0	0	0	0	0	0	0	0	0	0
Red												
Número de viajes ida y vuelta al servidor	1	1	1	1	1	1	1	1	1	1	1	1
Paquetes TDS enviados por cliente	1	1	1	1	1	1	1	1	1	1	1	1
Paquetes TDS enviados por servidor	1,172	1,172	1,172	1,172	1	1,296	17	2,388	2,388	392	224	3,403
Bytes enviados por cliente	266	380	266	380	394	254	464	280	396	568	1,154	998
Bytes recibidos del servidor	4,799,537	4,799,478	4,799,537	4,799,478	204	5,308,241	67,797	9,780,279	9,780,288	1,604,763	914,371	13,937,550
Tiempos (Milisegundos)												
Procesamiento cliente	391	360	219	453	16	500	16	969	907	312	94	1,657
Tiempo total de ejecución	562	1,250	390	1,296	453	1,156	484	1,156	1,750	2,046	1,062	9,578
Tiempo de espera respuesta servidor	171	890	171	843	437	656	468	187	843	1,734	968	7,921

Consulta P-I

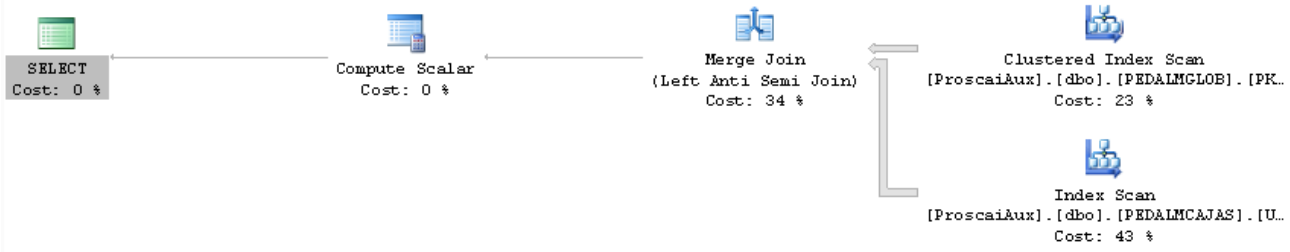



```
----  PEDIDO

SELECT *
FROM   PEDALMGLOB PG
ORDER BY PG.NUMPED
```

Consulta P-II


```
-- PEDIDO SURTIDO

SELECT *
FROM   PEDALMGLOB PG
WHERE  NUMPED IN (SELECT NUMPED FROM PEDALMCAJAS WHERE NUMPED = PG.NUMPED AND
CAJASTATUS <> 1 )
```

Consulta P-III

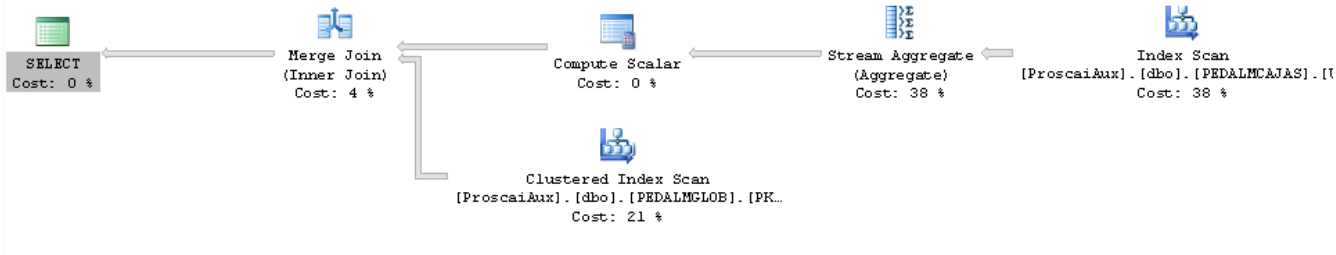



```
-- PEDIDO NO SURTIDO

SELECT *
FROM   PEDALMGLOB PG
WHERE  NUMPED NOT IN (SELECT NUMPED FROM PEDALMCAJAS WHERE NUMPED = PG.NUMPED )
```

Consulta P-IV

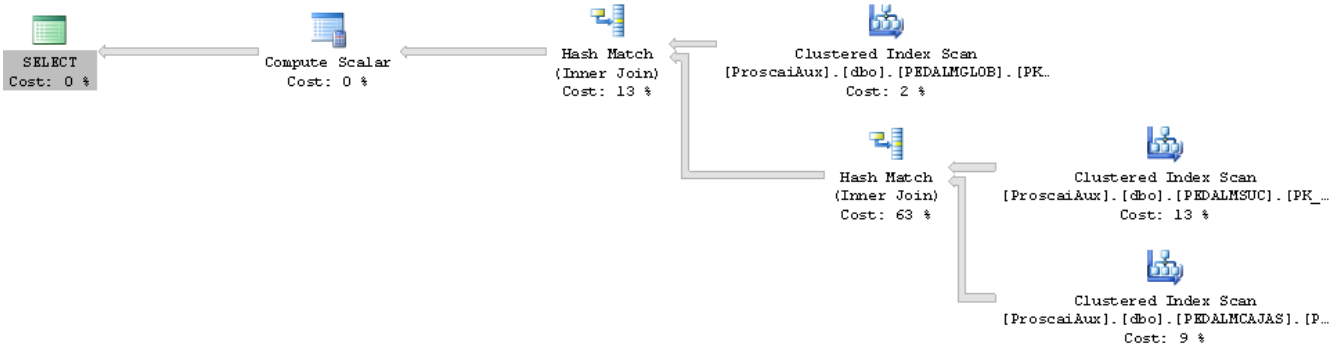
```
-- PEDIDO (CONTADOR DE SUCURSALES)

SELECT   PG.NUMPED, COUNT(DISTINCT (PC.IDSUC)) AS TotSucSurt
FROM     PEDALMGLOB PG, PEDALMCAJAS PC
WHERE    PG.NUMPED   = PC.NUMPED
GROUP BY PG.NUMPED
ORDER BY PG.NUMPED
```

Consulta P-V



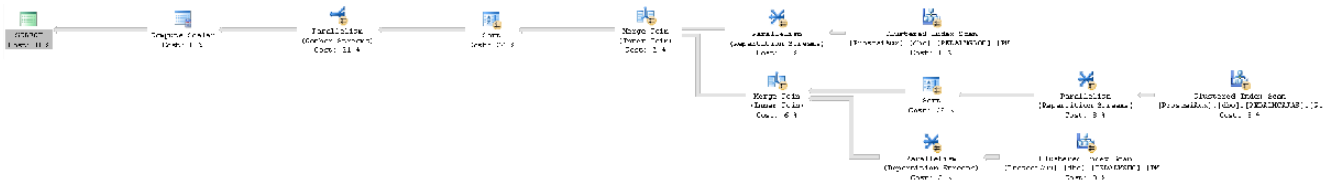

```
-- PEDIDO (CONTADOR DE CAJAS)

SELECT PG.NUMPED, COUNT(*) AS TotCajasSurt
FROM   PEDALMGLOB PG, PEDALMCAJAS PC
WHERE  PG.NUMPED   = PC.NUMPED
GROUP BY PG.NUMPED
ORDER BY PG.NUMPED
```

Consulta P-VI




```
-- PEDIDO, SUCURSAL Y CAJA (GENERAL)

SELECT *
FROM   PEDALMGLOB PG, PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PG.NUMPED = PC.NUMPED
AND    PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
```

Consulta P-VII

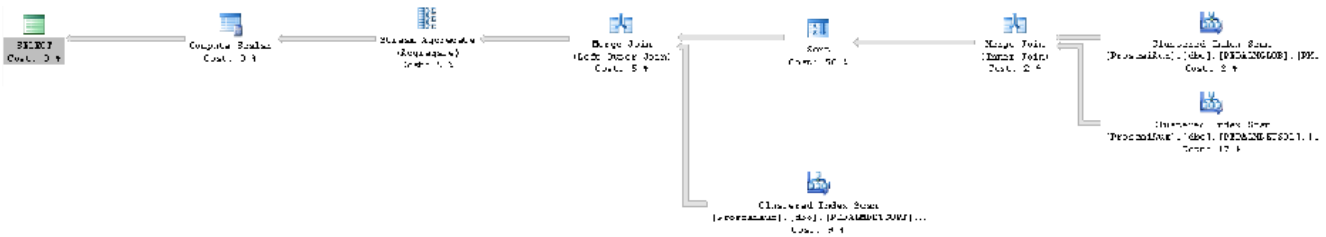


```
-- PEDIDO, SUCURSAL Y CAJA (ORDENADO POR IDCAJASUC)

SELECT *
FROM   PEDALMGLOB PG, PEDALMSUC PS, PEDALMCAJAS PC
WHERE  PG.NUMPED = PC.NUMPED
AND    PS.NUMPED = PC.NUMPED
AND    PS.IDSUC  = PC.IDSUC
ORDER BY PG.NUMPED,  PS.IDSUC, PC.IDCAJASUC
```

Consulta P-VIII

```
--PEDIDO, PIEZAS SURTIDAS (GENERAL - TotPzasSurtXCaja)

SELECT PG.NUMPED,PC.idCajaPed, PC.IDSUC, PC.IDCAJASUC,
          SUM(CantPzasSurt)    As TotPzasSurtXCaja,
          SUM(CantPaqSurtCli) As TotPaqSurtClitXCaja,
          SUM(CantPaqSurtAlm) As TotPaqSurtAlmtXCaja
FROM   PEDALMGLOB PG, PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PG.NUMPED      = PC.NUMPED
AND    PDSU.NumPed    = PC.NumPed
AND     PDSU.IdCajaPed = PC.idCajaPed
GROUP BY PG.NUMPED,PC.idCajaPed, PC.IDSUC, PC.IDCAJASUC
ORDER BY PG.NUMPED,  CONVERT(INT, PC.IDSUC), PC.IDCAJASUC
```

Consulta P-IX



```
--PEDIDO,  PIEZAS SOLICITADAS Y SURTIDAS

SELECT PG.NumPed,
             PDSO.CODART,PDSO.DESCAR,
```

```
            SUM(PDSO.CantPzasSol)   As TotPzasSol,
            SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
            SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
            SUM(CantPzasSurt)       As TotPzasSurt,
            SUM(CantPaqSurtCli)     As TotPaqSurtCli,
            SUM(CantPaqSurtAlm)     As TotPaqSurtAlm

FROM   PEDALMGLOB PG, PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE  PG.NumPed     *= PDSU.NumPed
AND      PDSO.IDSUC    *= PDSU.IDSUC
AND           PDSO.CodArt   *= PDSU.CodArt
AND           PG.NumPed     =  PDSO.NumPed
GROUP BY
            PG.NumPed, PDSO.idSuc, PDSO.CODART,PDSO.DESCAR
```

CONSULTA	I	II	III	IV	V	VI	VII	VII	IX	PROMEDIO
Perfil de consulta										
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0	0	0	0	0	0
Cantidad de SELECT	3	3	3	3	3	3	3	3	3	3
Filas consultadas por SELECT	4,831	1,732	6	4,835	4,833	92,060	92,066	24,704	67,251	32,480
Número de transacciones	0	0	0	0	0	0	0	0	0	0
Red										
Número de viajes ida y vuelta al servidor	3	3	3	3	3	3	3	3	3	3
Paquetes TDS enviados por cliente	3	3	3	3	3	3	3	3	3	3
Paquetes TDS enviados por servidor	143	50	4	20	20	5,190	5,191	229	1,299	1,350
Bytes enviados por cliente	314	420	394	546	492	504	620	1,198	1,360	650
Bytes recibidos del servidor	574,590	195,182	5,168	72,476	71,416	21,246,630	21,252,720	928,128	5,312,016	5,517,592
Tiempos (Milisegundos)										
Procesamiento cliente	31	47	0	0	265	1,266	1,672	125	828	470
Tiempo total de ejecución	62	109	125	2,500	343	1,562	5,187	953	1,453	1,366
Tiempo de espera respuesta servidor	31	62	125	2,500	78	296	3,515	828	625	896

Consulta I

```
SELECT *      FROM   PEDALMGLOB PG  WHERE  PG.NUMPED = 'P50624'
SELECT *      FROM   PEDALMSUC  PS  WHERE  PS.NUMPED = 'P50624'
SELECT *
FROM          PEDALMDETSOL PDSO, PEDALMDETSURT PDSU
WHERE         PDSO.NumPed   *= PDSU.NumPed
AND           PDSO.IDSUC    *= PDSU.IDSUC
AND           PDSO.CodArt   *= PDSU.CodArt
AND           PDSO.NumPed    = 'P50624'
```

Consulta II

```
SELECT *      FROM PEDALMGLOB PG WHERE  PG.NUMPED = 'P50624'
SELECT *      FROM PEDALMSUC PS  WHERE  PS.NUMPED = 'P50624'
SELECT PDSO.NumPed, PDSO.IDSUC,
              PDSO.CODART,PDSO.DESCAR,
              SUM(PDSO.CantPzasSol)   As TotPzasSol,
              SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
              SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
              SUM(CantPzasSurt)       As TotPzasSurt,
              SUM(CantPaqSurtCli)     As TotPaqSurtCli,
              SUM(CantPaqSurtAlm)     As TotPaqSurtAlm
FROM          PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE         PDSO.IDSUC    *= PDSU.IDSUC
AND           PDSO.CodArt   *= PDSU.CodArt
AND           PDSO.NumPed   *= PDSU.NumPed
AND           PDSO.NumPed    = 'P50624'
GROUP BY
              PDSO.NumPed, PDSO.idSuc, PDSO.CODART,PDSO.DESCAR
```

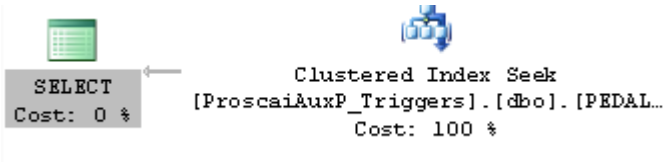
Consulta III

```
SELECT *      FROM PEDALMGLOB PG WHERE  PG.NUMPED = 'P50624'
SELECT PDSO.NumPed,
              PDSO.CODART,PDSO.DESCAR,
              SUM( PDSO.CantPzasSol)   As TotPzasSol,
              SUM(PDSO.CantPaqSolCli) As TotPaqSolCli,
              SUM(PDSO.CantPaqSolAlm) As TotPaqSolAlm,
              SUM(CantPzasSurt)       As TotPzasSurt,
              SUM(CantPaqSurtCli)     As TotPaqSurtCli,
              SUM(CantPaqSurtAlm)     As TotPaqSurtAlm
FROM          PEDALMDETSURT PDSU, PEDALMDETSOL PDSO
WHERE         PDSO.IDSUC    *= PDSU.IDSUC
AND           PDSO.CodArt   *= PDSU.CodArt
AND           PDSO.NumPed   *= PDSU.NumPed
AND           PDSO.NumPed    = 'P50624'
GROUP BY
              PDSO.NumPed,  PDSO.CODART,PDSO.DESCAR
order   by PDSO.CODART
```

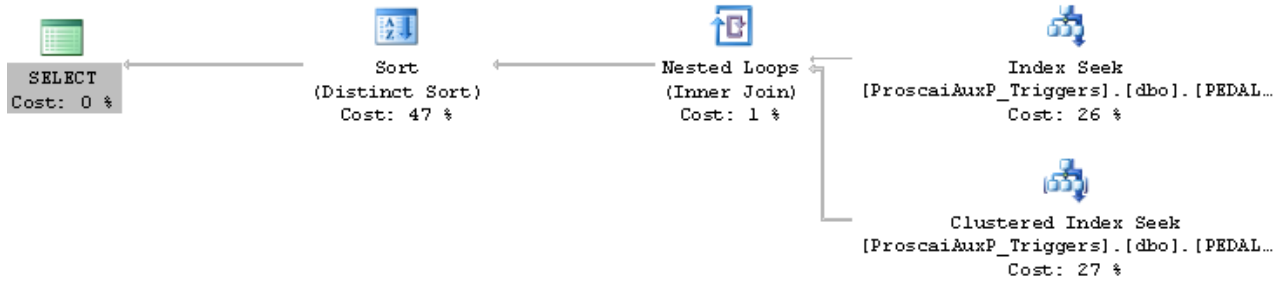
Consulta IV

```
SELECT *
FROM   PEDALMGLOB PG, PEDALMSUC PS, PEDALMCAJAS PC, PEDALMDETSURT PDSU
WHERE  PG.NUMPED      = PC.NUMPED
AND    PS.NUMPED      = PC.NUMPED
AND    PS.IDSUC       = PC.IDSUC
AND    PC.NumPed      = PDSU.NumPed
AND       PC.IdCajaPed  = PDSU.IdCajaPed
AND    PG.NUMPED = 'P50624'
ORDER BY PG.NUMPED,  PS.IDSUC, PC.IDCAJASUC
```

CONSULTA	C-I	C-II	C-III	C-IV	PROMEDIO
Perfil de consulta					
INSERT, DELETE ó UPDATE en consulta	0	0	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0	0	0
Cantidad de SELECT	9	9	6	3	7
Filas consultadas por SELECT	704	659	35	432	458
Número de transacciones	0	0	0	0	0
Red					
Número de viajes ida y vuelta al servidor	3	3	3	3	3
Paquetes TDS enviados por cliente	3	3	3	3	3
Paquetes TDS enviados por servidor	27	18	7	35	22
Bytes enviados por cliente	754	1,500	1,384	790	1,107
Bytes recibidos del servidor	100,394	64,378	16,974	133,098	78,711
Tiempos (Milisegundos)					
Procesamiento cliente	31	62	15	109	54
Tiempo total de ejecución	46	62	46	109	66
Tiempo de espera respuesta servidor	15	0	31	0	12




```
SELECT * FROM PEDALMART WHERE    NUMPED = 'P47789'
```
SELECT * FROM PEDALMART WHERE NUMPED = 'P47791'




```
SELECT CodArt,CodArtP, Sku,DescAr,NumPzaXPaqCli, NumPzaXPaqAlm
FROM PEDALMART
WHERE NUMPED
```

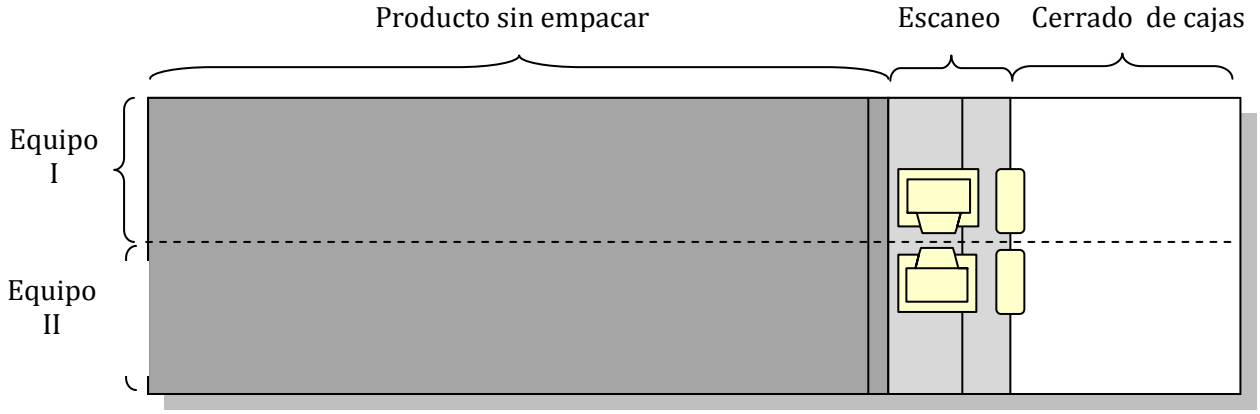
```
IN (SELECT NUMPED FROM PEDALMGLOB WHERE NUMPEDP = 'P47789')
```
group by CodArt,CodArtP, Sku,DescAr,NumPzaXPaqCli, NumPzaXPaqAlm

CONSULTA	**I**	**II**	**PROM**
Perfil de consulta			
INSERT, DELETE ó UPDATE en consulta	0	0	0
Filas afectadas por INSERT, DELETE, ó UPDATE	0	0	0
Cantidad de SELECT	2	1	1.5
Filas consultadas por SELECT	53	43	48
Número de transacciones	0	0	0
Red			
Número de viajes ida y vuelta al servidor	1	1	1
Paquetes TDS enviados por cliente	1	1	1
Paquetes TDS enviados por servidor	2	1	1.5
Bytes enviados por cliente	212	450	331
Bytes recibidos del servidor	5027	3575	4301
Tiempos (Milisegundos)			
Procesamiento cliente	15	0	7.5
Tiempo total de ejecución	15	46	30.5
Tiempo de espera respuesta servidor	0	46	23

XV. METODOLOGÍA PARA EL PREPARADO DE UN PEDIDO

Al prepara un pedido el jefe de almacén entrega las guías de preparación por mesa y jornada de trabajo con el total de piezas que se tienen que surtir. Cada uno de los responsables tiene la obligación verificar que el total de la mercancía se encuentre acomodada en la mesa. Este es el primer filtro para evitar un error en el surtido, ya que al finalizar el empaquetado no debe de faltar ni sobrar mercancía en la mesa.

Existen 2 equipos y 3 zonas por la mesa de trabajo. Las zonas por mesa son una de *producto sin empacar* en esta zona se acomoda la mercancía a surtir por mesa. Una *zona de escaneo* de mercancía donde se encuentra la computadora, el escáner y la impresora, posteriormente se encuentra una zona de *cerrado de cajas preparadas*.



Generalmente un equipo de trabajo está constituido por tres personas, Un responsable, el cual selecciona las sucursales a surtir, verifica los totales, y escanea la mercancía., otro empleado está encargado de acomodar la mercancía sin empacar y facilitarle los artículos al responsable anticipándosele. Por último un empleado que ayuda a empacar la mercancía en las cajas, ciérralas cajas y pegarles sus etiquetas respectivas.

Una vez que toda la mercancía se encuentra en la mesa el empleado encargado de la mercancía acerca la mercancía solicitada por sucursal al responsable . Este solamente puede tomar un paquete a la vez con una sola mano, escanea la mercancía, verificar que se actualizó y con la misma mano tiene que introducir la mercancía a la caja que se encuentra del lado contrario. El encargado de la caja acomoda la mercancía. *Los empleados solamente deben tocarla una vez*. El proceso se repite hasta que se llena la caja o se surte la sucursal. Entonces el responsable imprime la etiqueta. El encargado de cerrar las cajas, la cierra con una cinta inviolable, le acerca una nueva caja al que responsable del pedido, pone la etiqueta a la caja preparada y la acomoda en un carro de transporte. Al mismo tiempo el encargado de la mercancía tiene que visualizar los artículos solicitados por la nueva sucursal he irle acomodando de nueva cuenta los artículos que se requieren al responsable del equipo.

FIG. 80 RESPONSABLE Y ENCARGADO DE CAJAS

XVI. CONSULTA DE RESULTADOS A LA BASE DE DATOS

```
SELECT COUNT(DISTINCT (NUMCLI)) AS TOT_CLIENTES FROM PEDALMGLOB
TOT_CLIENTES
------------
176

(1 filas afectadas)

SELECT COUNT(NUMPED) FROM PEDALMGLOB
-----------
19779
(1 filas afectadas)

SELECT  MAX(IDGLOB)AS MaxIdCaja FROM PEDALMCAJAS
MaxIdCaja
-----------
492824
(1 filas afectadas)

 SELECT  SUM(NumSol) FROM PEDALMDET
-----------
28632078
(1 filas afectadas)

SELECT DATEPART(YEAR,FecIni), COUNT(NUMPED)
FROM PEDALMGLOB
AND  DATEPART(YEAR,FecIni)  <> '2012'
GROUP BY DATEPART(YEAR,FecIni)
-----------
2008 3
2009 926
2010 5359
2011 10383
 (4 filas afectadas)

SELECT NumPed, OrdCli, NumCli, NumEmp ,NumCajas, NumPzas, NumPaqCli,
FecElabIni,FecElabFin
 FROM PEDALMGLOB WHERE NUMPED = 'P58273'
```

NumPed	OrdCli	NumCli	NumEmp	NumCajas	NumPzas	NumPaqCli	FecElabIni	FecElabFin
P58273	300455681	CMX001	3131	293	240480	40080	27/04/2012 13:12:00	2012-04-20 13:11:00

```
(1 filas afectadas)
```

En la siguiente consulta se muestran el máximo de piezas surtidas por día con 6 máquinas trabajando.

```
SELECT  CONVERT(varchar(12),FecElab),
        SUM(NumPzas) As totNumPzas,
        COUNT(ContCaja)
FROM    PEDALMDET
GROUP BY CONVERT(VARCHAR(12),FecElab)
ORDER BY totNumPzas
```

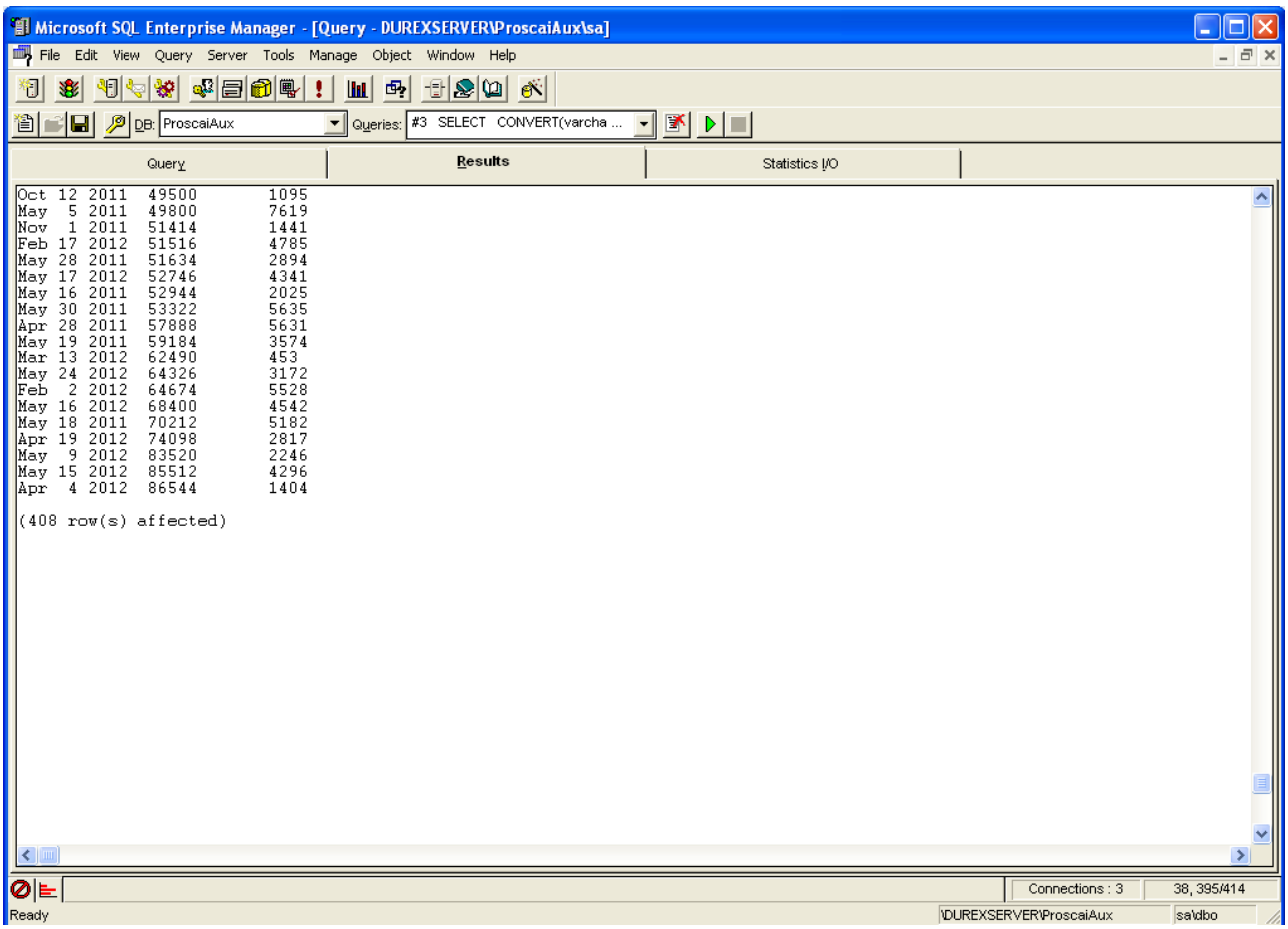


FIG. 81 CONSULTA DE PIEZAS SURTIDAS DIARIAMENTE

XVII. MANUAL DE PROVEEDORES COMERCIAL MEXICANA ENTREGA DE MERCANCÍA CENTROS DE DISTRIBUCIÓN.

c) Proveedores que entregan mercancía por Centro de Distribución

Cuando realice la entrega de mercancía en un centro de distribución, para suministrar a cada una de nuestras sucursales, deberá cubrir con los siguientes requisitos:

1. Se deberá generar folio de puerta para todas las entregas del Centro de Distribución vía I, II, III.

2. Por ningún motivo el transportista se podrá retirar de la sucursal sin el folio de puerta o acuse de recibo, al entregar mercancía por cualquier vía.

3. Deberá de entregar Original de la factura y 2 copias para las entregas que realice por la vía I y II.

4. Las cantidades facturas deben coincidir con la orden de compra.

5. Facture a nombre de la compañía que indica la orden de compra (Tiendas Comercial Mexicana, S.A. de C.V. y Districomex, S.A. de C.V.), indicando la vía a la que corresponde la mercancía.

6. Se le entregará como comprobante copia de su factura y acuse de recibo con las cantidades recibidas para la vía I y II.

7. Se deberá de entregar por vía, no deberá de mezclar la mercancía en los pallets que entregue.

8. Entregue copia de nuestro pedido de Compras o la orden de Compras (vía I y II).

9. Las entregas que se realizan para la vía III, se deberán facturar a cada sucursal de acuerdo a la orden de compra:

- Deberá enviar Original de la factura y 2 copias.
- Facture a nombre de Tiendas Comercial Mexicana, S.A. de C.V. y consignado a la Sucursal de entrega.
- Toda la mercancía debe venir palletizada.
- Deberá de enviar su mercancía dentro de la vigencia del pedido, respetando la fecha del mismo.
- Deberá de identificar el destino de la mercancía a través de la etiqueta diseñada por el CEDIS.
- Nuestro sistema tomará como base para el pago la fecha de entrega en sucursal, así como las condiciones del pedido vigente con la que fue recibida la mercancía.
- Se publicará en Provecomer el acuse de recibo y número de pallet de la mercancía entregada a la sucursal.
- Asegúrese de que el empaque resista el peso de la mercancía que contiene el pallet.
- Su factura la deberá de introducir en una caja la cual marcará con letra de gran tamaño y color visible “Aquí va la factura”.

10. . Nuestro sistema tomará como base del recibo el pedido vigente para la generación de su pago (Vía I y II).

11. La facturación es por unidad de transporte completa, una unidad puede contener varias ordenes de compras.

12. Cada proveedor será responsable de mantener actualizados los datos logísticos de sus productos con el CEDIS a través del Departamento de logística.

13. Sus pedidos consolidados serán publicados en la página de Provecomer los cuales identificará con la Sucursal 280

· Respetar la vigencia que indica el pedido.
· Coordinación en la vigencia de entrega al CEDIS y CEDIS a Sucursal
· Entrega al 100% de la mercancía en fecha estipulada
· Entregas parciales siempre y cuando estén en vigencia.

14. El cálculo de los requerimientos de las Sucursales se realiza sistemáticamente a través de modelos estadísticos, tomando en consideración la vía que entregue (I, II y III):

- Rol de entrega
- Días de inventario objetivos
- Demanda estimada
- Tiempo de entrega CEDIS
- Nivel de Servicio

15. Las entregas de mercancía en CEDIS, serán por citas programadas, las cuales deberán solicitar al área de citas:

- Citas fijas, establecidas por la Gerencia de Operaciones del CEDIS.
- Citas Variables, se deberán solicitar de Lunes a Sábado de las 8:00 A.M. a 17:00 P.M., en los teléfono 5270 9815, 5270 9821 y 5270 9831.
- No se recibirá mercancía ni vehículos sin cita o fuera de ella.
- Se deberá entregar el código identificador de la cita.

16. El proveedor será responsable de realizar la maniobra de descarga en el CEDIS.

17. Toda la mercancía se entregará *paletizada:*

- Las tarimas a utilizar deben ser CHEP o SMART y las tarimas deben ser de 4o x 48".
- Las tarimas las recibirá CEDIS hasta 800 Kg.
- Las tarimas deberán tener 3 vueltas de *emplaye* cubriendo el producto y la base.
- Colocar la etiqueta EAN y contenido del embalaje en una caja visible, de acuerdo a la vía de entrega (verificar manual de operación de CEDIS).

18. El cargo por concepto logístico será determinado por área Logística del CEDIS aplicándose en los pagos próximos de sus facturas.

19. No se aceptan pagos de fletes en los centros de distribución.

20. Se aplicará un cargo por penalización en incumplimiento de entregas, atraso en citas, camas con hueco, cajas c/*sku´s* distintos y cajas incompletas.

21. En el caso de que existiera alguna anomalía en el proceso de abasto podrá comunicarse

al teléfono 5270 9848 Atención Denia Morán

XVIII. EJEMPLOS DE SOAP Y WSDL

Petición SOAP

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com">
      <manufacturer>K2</manufacturer>
      <model>Fatbob</model>
    </m:GetEndorsingBoarder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Respuesta SOAP

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetEndorsingBoarderResponse xmlns:m="http://namespaces.snowboard-info.com">
      <endorsingBoarder>Chris Englesmann</endorsingBoarder>
    </m:GetEndorsingBoarderResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Descripción WSDL

```
<?xml version="1.0"?>

<!-- root element wsdl:definitions defines set of related services -->
<wsdl:definitions name="EndorsementSearch"
  targetNamespace="http://namespaces.snowboard-info.com"
  xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
  xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

  <!-- wsdl:types encapsulates schema definitions of communication types;
here using xsd -->
  <wsdl:types>

    <!-- all type declarations are in a chunk of xsd -->
    <xsd:schema targetNamespace="http://namespaces.snowboard-info.com"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema">

      <!-- xsd definition: GetEndorsingBoarder [manufacturer string, model
string] -->
      <xsd:element name="GetEndorsingBoarder">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="manufacturer" type="string"/>
            <xsd:element name="model" type="string"/>
        </xsd:sequence>
```

```
      </xsd:complexType>
      </xsd:element>

      <!-- xsd definition: GetEndorsingBoarderResponse [...
endorsingBoarder string ...] -->
      <xsd:element name="GetEndorsingBoarderResponse">
      <xsd:complexType>
        <xsd:all>
          <xsd:element name="endorsingBoarder" type="string"/>
        </xsd:all>
      </xsd:complexType>
      </xsd:element>

      <!-- xsd definition: GetEndorsingBoarderFault [... errorMessage
string ...] -->
      <xsd:element name="GetEndorsingBoarderFault">
      <xsd:complexType>
        <xsd:all>
          <xsd:element name="errorMessage" type="string"/>
        </xsd:all>
      </xsd:complexType>
      </xsd:element>

    </xsd:schema>
  </wsdl:types>

  <!-- wsdl:message elements describe potential transactions -->

  <!-- request GetEndorsingBoarderRequest is of type GetEndorsingBoarder --
>
  <wsdl:message name="GetEndorsingBoarderRequest">
    <wsdl:part name="body" element="esxsd:GetEndorsingBoarder"/>
  </wsdl:message>

  <!-- response GetEndorsingBoarderResponse is of type
GetEndorsingBoarderResponse -->
  <wsdl:message name="GetEndorsingBoarderResponse">
    <wsdl:part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
  </wsdl:message>

  <!-- wsdl:portType describes messages in an operation -->
  <wsdl:portType name="GetEndorsingBoarderPortType">

    <!-- the value of wsdl:operation eludes me -->
    <wsdl:operation name="GetEndorsingBoarder">
      <wsdl:input message="es:GetEndorsingBoarderRequest"/>
      <wsdl:output message="es:GetEndorsingBoarderResponse"/>
      <wsdl:fault message="es:GetEndorsingBoarderFault"/>
    </wsdl:operation>
  </wsdl:portType>

  <!-- wsdl:binding states a serialization protocol for this service -->
  <wsdl:binding name="EndorsementSearchSoapBinding"
                type="es:GetEndorsingBoarderPortType">
```

```
    <!-- leverage off soap:binding document style @@@(no wsdl:foo pointing
at the soap binding) -->
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>

    <!-- semi-opaque container of network transport details classed by
soap:binding above @@@ -->
    <wsdl:operation name="GetEndorsingBoarder">

      <!-- again bind to SOAP? @@@ -->
      <soap:operation soapAction="http://www.snowboard-
info.com/EndorsementSearch"/>

      <!-- furthur specify that the messages in the wsdl:operation
"GetEndorsingBoarder" use SOAP? @@@ -->
      <wsdl:input>
        <soap:body use="literal"
                namespace="http://schemas.snowboard-
info.com/EndorsementSearch.xsd"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"
                namespace="http://schemas.snowboard-
info.com/EndorsementSearch.xsd"/>
      </wsdl:output>
      <wsdl:fault>
        <soap:body use="literal"
                namespace="http://schemas.snowboard-
info.com/EndorsementSearch.xsd"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <!-- wsdl:service names a new service "EndorsementSearchService" -->
  <wsdl:service name="EndorsementSearchService">
    <wsdl:documentation>snowboarding-info.com Endorsement
Service</wsdl:documentation>

    <!-- connect it to the binding "EndorsementSearchSoapBinding" above -->
    <wsdl:port name="GetEndorsingBoarderPort"
                binding="es:EndorsementSearchSoapBinding">

      <!-- give the binding an network address -->
      <soap:address location="http://www.snowboard-
info.com/EndorsementSearch"/>
    </wsdl:port>
  </wsdl:service>

 </wsdl:definitions>
```

XIX. ÍNDICE DE DOCUMENTOS

XML Cadenas

```
<?xml version="1.0" encoding="utf-8" ?>

- <CADENAS>
-   <Cadena name="Walmart" ID="WMX">
        <Logo>walmart_logo2.gif</Logo>
        <TipoArchivos>PWMX*.prn</TipoArchivos>
        <MaxArtCaja>12</MaxArtCaja>
        <RepTipo>GEN</RepTipo>
        <RepEsp>FALSE</RepEsp>
        <CedisSel>true</CedisSel>
        <Depart>FALSE</Depart>
        <Depart>FALSE</Depart>
       <TipoPed>1</TipoPed>
        <TipoPrep>2</TipoPrep>
    </Cadena>
-   <Cadena name="Chedrahui" ID="CHE">
        <Logo>loginbck_r2_c2.gif</Logo>
        <TipoArchivos>PCHE*.prn</TipoArchivos>
       <MaxArtCaja>0</MaxArtCaja>
        <RepTipo>CHE</RepTipo>
        <RepEsp>TRUE</RepEsp>
        <CedisSel>FALSE</CedisSel>
        <Depart>FALSE</Depart>
        <TipoPed>1</TipoPed>
        <TipoPrep>1</TipoPrep>
    </Cadena>
-   <Cadena name="Suburbia" ID="SUB">
        <Logo>suburbia.bmp</Logo>
        <TipoArchivos>PSUB*.prn</TipoArchivos>
        <MaxArtCaja>0</MaxArtCaja>
        <RepTipo>SUB</RepTipo>
        <RepEsp>TRUE</RepEsp>
        <CedisSel>TRUE</CedisSel>
        <Depart>TRUE</Depart>
        <TipoPed>1</TipoPed>
        <TipoPrep>1</TipoPrep>
    </Cadena>
-   <Cadena name="COMERSA" ID="CMX">
        <Logo>comer2.jpg</Logo>
        <TipoArchivos>PCMX*.prn</TipoArchivos>
        <MaxArtCaja>0</MaxArtCaja>
        <RepTipo>CMX</RepTipo>
        <RepEsp>TRUE</RepEsp>
        <CedisSel>TRUE</CedisSel>
        <Depart>TRUE</Depart>
        <TipoPed>2</TipoPed>
        <TipoPrep>2</TipoPrep>
    </Cadena>
</CADENAS>
```

InicialesCadena

```
Public Sub InicialesCadena(ByVal NomCad As String, ByRef Cad As StrucCadena, ByVal Carpeta As
String)
        'Subrutina encargada de ingresar valores del Cliente y/o Cadena
        Try

        Cad.Nombre = NomCad

        ' Usando LINQ
        Dim CadXML As XElement = XElement.Load(Carpeta & NomCadXml)
        Dim BNodo As XElement
            Dim query = From r As XElement In CadXML...<Cadena>
                        Where r.@<name> = NomCad
            BNodo = query.FirstOrDefault()
            Cad.Logo = BNodo.<Logo>.Value
            Cad.TipoArch = BNodo.<TipoArchivos>.Value
            Cad.ArtXCaja = BNodo.<MaxArtCaja>.Value
            Cad.RepTipo = BNodo.<RepTipo>.Value
            Cad.RepEsp = BNodo.<RepEsp>.Value
            Cad.CedisSel = BNodo.<CedisSel>.Value
            Cad.Depart = BNodo.<Depart>.Value

        Catch ex As Exception

            If ex.GetType.ToString = "System.NullReferenceException" Then

                MsgBox("La Cadena: " & NomCad & ", no esta dada de alta." & vbNewLine &
                "Informar Sistemas", MsgBoxStyle.Critical, "InicialesCadena")
                Environment.Exit(0)

            Else
                MsgBox(ex.Message)
            End If

        End Try


    End Sub
```

Código

```
Imports System.Data.SqlClient

Public Class FrmBase
    Friend Login As Boolean
    Public Sub New()
        Dim strLinea As String
        'strLinea = "CONTROL.INI"
        'strLinea = "Suburbia"
        'strLinea = "WALMART"
        'strLinea = "CYA.Ini"
        strLinea = "COMERSA"
        'strLinea = "PALACIO.INI"
        'strLinea = "Chedrahui"
        Inicio(strLinea)

    End Sub
    Public Sub New(ByVal strLinea As String)

        Inicio(Command())

    End Sub
    Public Sub Inicio(ByVal Cadena As String)
        'CARGA VALORES INICIALES DE LA APLICACIÓN
```

```
        InicialesCarga(Ini)
        InicialesCadena(Cadena, Cad, Ini.Inicial)
        InicialesBase(ConIni)

        'Inicializa componentes
        InitializeComponent()
        ValIniFrmBase()

        Show()
    End Sub
    Private Sub ValIniFrmBase()
        'Define los valores iniciales que se despliegan en la FrmBase.
        Try

            Me.Text = "Salidas de A.P.T - " & Cad.Nombre
            Me.EtiCadNom.Text = Cad.Nombre
            DespliegaImagen(ImgCad, Ini.Logos & Cad.Logo)
            DepliegaPedidos()

        Catch e As Exception
            MsgBox(e.Message, MsgBoxStyle.Critical, "Error en la carga de DefValores")
            Application.Exit()
        End Try

    End Sub
    Private Sub DepliegaPedidos()
        'DESPLIEGA PEDIDOS EN "PedList, UTILIZANDO LA CLASE System.IO "

        'Referencia al directorio
        Dim Dir As New IO.DirectoryInfo(Ini.Pedidos)
        Dim InfArchivos As IO.FileInfo() = Dir.GetFiles(Cad.TipoArch)
        Dim InfArch As IO.FileInfo
        PedList.Items.Clear()

        'Muestra la lista de todos los archivos del directorio.
        For Each InfArch In InfArchivos
            Me.PedList.Items.Add(InfArch)
        Next

    End Sub

    Private Sub PedList_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles PedList.Click
        'SELECCIONA PEDIDO PULSANDO  "PedList"

        'Limpia Valores
        Limpiar(True)

        If MsgBox("¿Desea   cargar lista de empaque?" & vbNewLine & "Pedido:  " & PedList.Text, vbYesNo) = vbYes
Then

            LimpiaPedSuc(True)
            Dim ArtFalt As New ClsArtículos
            If CargaPedido(PedList.Text, F1Gen) = True Then

                BarraHerr.Visible = True
                If Cad.Depart Then
                    CargaDepto(Ped.NumCli)
                End If

                BarraHerr.Visible = True

                PedidoValIni()
                Dim d As String = Ped.NumCli


            Else

                If ArtFalt.TotArt <> 0 Then

                    If MsgBox("Existen artículos con incidencias" & vbNewLine & "¿Desea visualizar artículos
faltantes?", vbYesNo, "Artículos Faltantes") = vbYes Then

                        Dim obj As New FrmFaltantes(ArtFalt)
                        obj.Show()
```

```
                End If

            Else
                MsgBox("Error en la carga", vbCritical)
            End If
        End If


    Else

        TxtPedido.Text = ""
        If TxtPedido.CanFocus Then TxtPedido.Focus()

    End If

End Sub

Private Sub CargaDepto(ByVal NumCli As String)
    Dim strSQL As String
    Dim DR As Data.SqlClient.SqlDataReader = Nothing
    Dim Conn As Data.SqlClient.SqlConnection = Nothing
    Try
        strSQL = "SELECT *  FROM  DEPARTCLI DC, DEPARTDUR DD" & vbNewLine
        strSQL = strSQL & "WHERE     DC.IDDEPDUR = DD.IDDEPDUR" & vbNewLine
        strSQL = strSQL & "AND       NUMCLIP = '" & NumCli & "' " & vbNewLine
        strSQL = strSQL & "AND       DD.IDDEPDUR = '" & Mid(Ped.ArtGen(1).CodArtP, 2, 1) & "'"

        Conn = ConnDr(DR, strSQL)

        While DR.Read()
            GBDepto.Visible = True
            TxtDepto.Text = DR("IDDEPCLI")
            TxtDeptoDesc.Text = DR("DESCRIPCION")
            Exit Sub
        End While
        GBDepto.Visible = False
    Catch Ex As Exception

        MsgBox(Ex.Message, MsgBoxStyle.Critical, "Error Carga Departamento")

    Finally
        Conn.Close()
    End Try

End Sub
Private Sub TxtPedido_KeyPress(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs)
Handles TxtPedido.KeyPress
    'Rutina homologa retorno de carro de los diferentes tipos de escaners
    If Asc(e.KeyChar) = 13 Then
        '    'KeyAscii = 0
        My.Computer.Keyboard.SendKeys("{TAB}")
    End If

End Sub


Private Sub TxtPedido_LostFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TxtPedido.LostFocus
    'SELECCIONA PEDIDO AL SALIR DE "TxtPedido"

    Dim I As Integer
    Dim ContL As Integer

    'Verifica tamano del archivo.
    If Len(TxtPedido.Text) <> 6 Then

        If Len(TxtPedido.Text) <> 0 Then MsgBox("El tamaño de pedido no concuerda.")
        TxtPedido.Text = ""
        Exit Sub

    End If

    'Verifica que el pedido este dado de alta.
    ContL = PedList.Items.Count
```

```
        For I = 0 To PedList.Items.Count - 1

            If CStr(Mid(PedList.Items(I).ToString, 8, 6)) = UCase(sender.Text) Then
                If ContL <> I Then
                    PedList.SelectedIndex = I
                    Me.PedList_Click(PedList.Items(I).ToString, e)

                End If


                TxtPedido.Text = ""


                Exit Sub

            End If


        Next I


        MsgBox("El pedido no se encuentra capturado")
        TxtPedido.Text = ""


    End Sub
#Region "Limpieza"
    Public Sub Limpiar(Optional ByVal LimpiaGrid As Boolean = False)

        'ContA = 0
        'ReDim ArtF(0)

        'Ped = New ClsPedido
        'Ped.NumPed = "P46000"
        'NumEmp = "000001"
        If NumEmp <> "" Then ControlUsuarios(ConSelect, NumEmp, Ped.NumPed)
        NumEmp = ""
        BarraHerr.Visible = False
        If Cad.Depart Then GBDepto.Visible = False
        LimpiaPedSuc(False)
        If LimpiaGrid = True Then F1Gen.Rows.Clear()

    End Sub

    Public Sub LimpiaPedSuc(ByVal Visible As Boolean)
        'Dependiedio del estado del pedido prepara a TxtIdSuc ó  TxtPedido para que se le asigne un valor
        GBSuc.Visible = Visible
        TxtIdSuc.Clear()
        TxtPedido.Clear()

        If Visible = True Then

            TxtIdSuc.Focus()


        Else

            TxtPedido.Focus()

        End If

    End Sub

    Private Sub BtnRefrescar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnRefrescar.Click
        'Botón de refrescar
        DepliegaPedidos()
        Limpiar(True)

    End Sub

#End Region

    Public Sub TestEncoding()
        Dim password As String = InputBox("Enter the password:")
```

```
        Dim wrapper As New Simple3Des(password)
        Dim plainText As String = InputBox("Enter the plain text:")
        Dim cipherText As String = wrapper.EncryptData(plainText)
        MsgBox("The cipher text is: " & cipherText)
    End Sub
    Private Sub TxtIdSuc_KeyPress(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs)
Handles TxtIdSuc.KeyPress
        'Rutina homologa retorno de carro de los diferentes tipos de escaners
        If Asc(e.KeyChar) = 13 Then
            '    'KeyAscii = 0
            My.Computer.Keyboard.SendKeys("{TAB}")
        End If
    End Sub
    Private Sub TxtIdSuc_LostFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TxtIdSuc.LostFocus
        'VERIFICA EL NÚMERO DE SUCURSAL Y REDIRECCIONA
        Select Case Len(TxtIdSuc.Text)

            Case 0
                Exit Sub
            Case Is > 3
                If IsNumeric(Mid(TxtIdSuc.Text, 4, Len(TxtIdSuc.Text) - 3)) = True _
                   And (Mid(TxtIdSuc.Text, 1, 3)).ToUpper = "SUC" Then
                    DespliegaSuc()
                    TxtIdSuc.Text = ""
                    Exit Sub
                End If

        End Select

        MsgBox("El número de sucursal: " & TxtIdSuc.Text & " no es valido." & vbNewLine & "Favor de verificar.",
vbInformation, "Carga de Sucursal.")
        TxtIdSuc.Text = ""
        TxtIdSuc.Focus()

    End Sub
    Private Sub BtnSurtir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnSurtir.Click
        DespliegaSuc()
    End Sub
    Private Sub DespliegaSuc()
        'PROCESO DE DESPLEGADO DE SUCURSALES
        'Valida el usuario y dependiendo de tipo de surtido (Múltiple o individual),
        'asigna rutina correspondiente.

        'Dim I As Integer

        'Abortar = False
        'Ped.ArtCantPedF()

        'ModifCajas = True
        ''ContCajas = 0

        If NumEmp = "" Then
            'Me.AddOwnedForm(FrmLogin)
            'Dim FormLogin As New FrmLogin
            If FrmLogin.ShowDialog = DialogResult.No Then
                Exit Sub
            End If



        End If

        'CargaCantArtIni(False)


        If Len(TxtIdSuc.Text) = 0 Then

            CargaSucMult()

        Else

            CargaSucInd()
```

```
        End If

    End Sub
    Private Sub CargaSucInd()
        'DESPLIEGA SUCURSAL INDIVIDUAL EN FrmDetCajas
        Dim I As Integer

        For I = 0 To F1Gen.RowCount - 1

            If F1Gen.Item(1, I).Value = Mid(TxtIdSuc.Text, 4, Len(TxtIdSuc.Text) - 3) Then
                CargaArtDetCajas(I)
                Exit Sub

            End If

        Next I

        MsgBox("El número de sucursal no es valido." & vbNewLine & "Favor de verificar.", vbCritical)
        TxtIdSuc.Text = ""
        TxtIdSuc.Focus()

    End Sub

    Private Sub CargaSucMult()
        Dim I As Integer

        'LimpiaNumSuc(False)
        Abortar = False
        For I = 0 To F1Gen.RowCount - 1

            If F1Gen.Item(0, I).Value = True Then

                CargaArtDetCajas(I)

            End If

            If Abortar = True Then Exit Sub

        Next I

        If Ped.SucCantSurt <> 0 Then

            'ImprimeReporteGlob()

        End If

        'LimpiaNumSuc(False)

        'LimpiarGrid(F1Gen)

    End Sub
    Private Sub CargaArtDetCajas(ByVal I As Integer)

        If CStr(F1Gen.Item(4, I).Value) = "N" Then

            FrmDetCajas.CargaSucursal(F1Gen.Item(1, I).Value, F1Gen.Item(3, I).Value, 1, True, I)

        Else

            FrmDetCajas.CargaSucursal(F1Gen.Item(1, I).Value, F1Gen.Item(3, I).Value, 1, False, I)

        End If
        FrmDetCajas.ShowDialog()
        FrmDetCajas = Nothing

    End Sub




    Public Sub PedidoValIni()

        Dim strSQL As String
        Dim DR As Data.SqlClient.SqlDataReader = Nothing
        Dim Conn As Data.SqlClient.SqlConnection = Nothing
```

```
        strSQL = "SELECT FecElabIni FROM PEDALMGLOB " & vbNewLine
        strSQL = strSQL & "WHERE NumPed = '" & Ped.NumPed & "' " & vbNewLine
        Conn = ConnDr(DR, strSQL)


        While DR.Read()
            'Ped.CantCajas = Rs.Fields("NumCajas")
            'Ped.CantPzas = Rs.Fields("NumPzas")

            'Ped.NumPaqCli = Rs.Fields("NumPaqCli")
            'Ped.NumPaqAlm = Rs.Fields("NumPaqAlm")
            'Ped.CanSucSur = Rs.Fields("NumSuc")

            Ped.FecElabIni = DR("FecElabIni")
            ' VacioGlob = False

        End While
        Conn.Close()
        DR.Close()
        strSQL = "SELECT IDSUC, NUMCAJAS FROM PEDALMSUC " & vbNewLine
        strSQL = strSQL & "WHERE NUMPED = '" & Ped.NumPed & "' "
        'Clipboard.Clear()
        'Clipboard.SetText(strSQL)
        Conn = ConnDr(DR, strSQL)

        While DR.Read()
            For DGCont As Integer = 0 To F1Gen.RowCount - 1

                If F1Gen.Item(1, DGCont).Value = DR("IDSUC").ToString Then
                    F1Gen.Item(3, DGCont).Value = DR("NUMCAJAS")
                    F1Gen.Item(4, DGCont).Value = "N"
                    F1Gen.Rows(DGCont).DefaultCellStyle.BackColor = Color.DarkSlateBlue
                    F1Gen.Rows(DGCont).DefaultCellStyle.ForeColor = Color.White
                    F1Gen.Item(0, DGCont).Value = False
                    Exit For
                End If

            Next

        End While
        Conn.Close()


    End Sub
    Private Sub F1Gen_CellClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles F1Gen.CellClick
        '   F1Gen.Item(0, 1).Value = Not (F1Gen.Item(0, 1).Value = False)
    End Sub

    Private Sub F1Gen_CellContentClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles F1Gen.CellContentClick

    End Sub

    Private Sub F1Gen_KeyPress(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyPressEventArgs)
Handles F1Gen.KeyPress

        If Asc(e.KeyChar) = 13 Then

            F1Gen.Item(0, F1Gen.CurrentCell.RowIndex - 1).Value = True
        End If
    End Sub


    Private Sub F1Gen_RowEnter(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles F1Gen.RowEnter

        'F1Gen.Item(0, e.RowIndex).Value = (Not (F1Gen.Item(0, e.RowIndex).Value)

    End Sub

    Private Sub BtnTodoLimp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnTodoLimp.Click
        For Each Ren In F1Gen.Rows
```

```
            F1Gen.Item(0, Ren.index).Value = False

        Next
    End Sub

    Private Sub BtnTodoSel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnTodoSel.Click
        For Each Ren In F1Gen.Rows
            F1Gen.Item(0, Ren.index).Value = True

        Next
    End Sub

    Private Sub PedList_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
PedList.SelectedIndexChanged

    End Sub

    Private Sub TxtPedido_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TxtPedido.TextChanged

    End Sub
End Class
```

FrmDetCajas

```
Imports System.Data.SqlClient

Public Class FrmDetCajas

    Private bindingSource1 As New BindingSource()

#Region "Despliegue"

    Public Sub CargaSucursal(ByVal IdSuc As String, ByVal NumCajas As Integer, ByVal NvoIdCaja As Integer, ByVal
SucSurt As Boolean, ByVal IdF1 As Integer, Optional ByVal Cancelada As Boolean = False)

        IdFCont = IdF1
        TotCajas = NumCajas
        IdCaja = NvoIdCaja
        SucSurtida = SucSurt
        Cancel = Cancelada

        ValoresIni(IdSuc)
        TxtCajas.Text = IdCaja & " / " & TotCajas
        StatusCaja(SucSurtida)

    End Sub
    Private Sub ValoresIni(ByVal IdSuc As Integer)
        Dim I As Integer
        Dim CodArt As String

        EncabezadoIni(IdSuc)

        Ped.Suc(TxtIdSuc.Text).CajaBorraTodas()
        If SucSurtida = False Then
            CargaCaja(1)
            F1DetArt.DataSource = Nothing
            F1DetArt.Rows.Clear()
            With Ped.Suc(TxtIdSuc.Text)

                F1DetArt.RowCount = .TotArt
                For I = 1 To .TotArt
                    CodArt = .Art(I).CodArt
                    F1DetArt.Item(0, I - 1).Value = Ped.ArtGen(I).CodArtP
                    F1DetArt.Item(1, I - 1).Value = CodArt
                    F1DetArt.Item(2, I - 1).Value = Ped.ArtGen(I).Desc
                    F1DetArt.Item(3, I - 1).Value = 0
                    F1DetArt.Item(4, I - 1).Value = 0
                    F1DetArt.Item(5, I - 1).Value = Ped.ArtGen(I).NumPzaXPaqAlm
                    F1DetArt.Item(6, I - 1).Value = Ped.ArtGen(I).NumPzaXPaqCli
                    F1DetArt.Item(7, I - 1).Value = 0
                    F1DetArt.Item(8, I - 1).Value = .Art(I).CantPzasSol
                    F1DetArt.Item(9, I - 1).Value = .Art(I).CantPzasSol
```

```
                Next I

            End With
            F1DetArt.DefaultCellStyle.BackColor = System.Drawing.Color.Maroon
        Else
            DespliegaArtículosSurtidos()
        End If
    End Sub
    Private Sub EncabezadoIni(ByVal IdSuc As Integer)
        On Error GoTo Errores

        TxtIdSuc.Text = IdSuc
        TxtNumPed.Text = Ped.NumPed
        TxtNomSuc.Text = Ped.Suc(IdSuc).Desc
        TxtNomArt.Text = ""

        Exit Sub

Errores:


    End Sub
    Private Sub CargaCaja(ByVal IdCaja As Integer)

        Dim Caja As New ClsCaja
        Caja.CajaIdSuc = IdCaja
        Caja.ColCompArt = Ped.Suc(TxtIdSuc.Text).ColCompArt
        Ped.Suc(TxtIdSuc.Text).CajaIngresa = Caja

    End Sub
    Private Sub StatusCaja(ByVal Val As Boolean)
        'DESPLIEGA EN EL ESTATUS DE LA CAJA

        If Cancel = False Then
            TxtCodArt.Enabled = Not (Val)
            TxtCodArt.Visible = True
            If TxtCodArt.CanFocus Then
                TxtCodArt.Enabled = Not (Val)

            End If
            If Val Then

                EtiStatus.ForeColor = Color.Green ' = &H40C0&
                EtiStatus.Text = "Cargado"
                TxtIdSuc.Focus()
            Else
                EtiStatus.ForeColor = Color.Green
                EtiStatus.Text = "Abierto"
                TxtCodArt.Focus()
            End If

        Else

            EtiStatus.ForeColor = Color.Red
            EtiStatus.Text = "Reempaque"
            TxtCodArt.Focus()
        End If




        'F1DetArt.Enabled = Not (Val)



    End Sub

#Region "Base de Datos"

    Private Sub DespliegaArtículos()
        Dim strSQL As String
        Try
            ' Set up the DataGridView.
            With F1DetArt
                ' Automatically generate the DataGridView columns.
```

```
                .AutoGenerateColumns = True

                ' Set up the data source.
                strSQL = "SELECT * FROM  PEDALMDET       " & vbNewLine
                strSQL = strSQL & "WHERE    NUMPED     = '" & Ped.NumPed & "'    " & vbNewLine
                strSQL = strSQL & "AND      IDSUC      = '" & TxtIdSuc.Text & "'" & vbNewLine
                strSQL = strSQL & "AND      NUMPZAS <> 0 " & vbNewLine
                strSQL = strSQL & "Order By ContCaja, Consec"
                bindingSource1.DataSource = ConnData(strSQL)
                .DataSource = bindingSource1

                .AutoResizeColumns()
                .BorderStyle = BorderStyle.Fixed3D


                .EditMode = DataGridViewEditMode.EditOnEnter
            End With
        Catch ex As SqlException
            MsgBox(ex.Message, "Error", vbCritical)
            System.Threading.Thread.CurrentThread.Abort()
        End Try
    End Sub
    Private Sub DespliegaArtículosSurtidos()
        Dim strSQL As String
        Try
            With F1DetArt
                .DataSource = Nothing
                .Rows.Clear()
                .DefaultCellStyle.BackColor = System.Drawing.Color.DodgerBlue
                .AutoGenerateColumns = False
                strSQL = "SELECT PD.CODART AS CODART, DESCAR, NUMPZAS FROM  PEDALMDET PD, MARTIC MA   " &
vbNewLine
                strSQL = strSQL & "WHERE    NUMPED     = '" & Ped.NumPed & "'" & vbNewLine
                strSQL = strSQL & "AND      IDSUC      = '" & TxtIdSuc.Text & "'       " & vbNewLine
                strSQL = strSQL & "AND      MA.CODART  = PD.CODART " & vbNewLine
                strSQL = strSQL & "AND      NUMPZAS <> 0 " & vbNewLine
                strSQL = strSQL & "Order By ContCaja, Consec"

                bindingSource1.DataSource = ConnData(strSQL)
                .DataSource = bindingSource1
                .Columns(0).DataPropertyName = "CODARTP"
                .Columns(1).DataPropertyName = "CODART"
                .Columns(2).DataPropertyName = "DESCAR"
                .Columns(3).DataPropertyName = "NUMPZAS"

                '  .ClearSelection()
            End With

        Catch ex As SqlException
            MsgBox(ex.Message, "Error", vbCritical)
            System.Threading.Thread.CurrentThread.Abort()
        End Try
    End Sub

#End Region 'Base de Datos

#End Region 'Despliegue

#Region "Salida"

    Private Sub GuardaImprimeValores()

        If IdCaja = TotCajas Then

            If CargaClases() = False Then Exit Sub

        End If
        If VerifVal() = False Then Exit Sub
        If EtiStatus.Text = "Abierto" Then
            If TipoPrep <> PrepEtapas Then

                If GuardaValores = False Then Exit Sub

            Else
```

```
            If GuardaValoresInd = False Then Exit Sub

        End If
    ElseIf Cancel = True Then

        'ReemplazaValores()

    End If
    CantArt = 0
    ImprimeReportes()
    Me.Close()

    'ModFrmF1()
    'NumSuc = NumSuc + 1
End Sub
Private Function CargaClases() As Boolean

    Dim I As Integer
    Dim CantPaqSurtCli As Integer
    Dim CantPaqSurtDur As Integer
    Dim CantPzasSurt As Integer
    Dim CodArt As String
    Dim CodArtP As String

    If CantArt = 0 And SucSurtida = False Then

        MsgBox("LA CANTIDAD DE ARTÍCULOS CAPTURADOS ES 0", vbInformation, "Verificar")
        CargaClases = False
        Exit Function

    End If
    CargaClases = VerCanEmp()
    If CargaClases = False Then Exit Function

    For I = 0 To F1DetArt.RowCount - 1
        CantPzasSurt = F1DetArt.Item(4, I).Value
        If CantPzasSurt <> 0 Then
            CodArtP = F1DetArt.Item(0, I).Value
            CodArt = F1DetArt.Item(1, I).Value
            CantPaqSurtDur = F1DetArt.Item(4, I).Value
            CantPaqSurtCli = CantPaqSurtDur * F1DetArt.Item(5, I).Value / F1DetArt.Item(6, I).Value

            With Ped.Suc(TxtIdSuc.Text).Caja(IdCaja).Art(CodArt) 'Artìculos

                .CantPaqSurtDur = CantPaqSurtDur
                .CantPaqSurtCli = CantPaqSurtCli
                .CantPzasSurt = CantPzasSurt

            End With

            With Ped.ArtGen(CodArtP) 'Artículos Generales

                .CantPaqSurtDur = .CantPaqSurtDur + CantPaqSurtDur
                .CantPaqSurtCli = .CantPaqSurtCli + CantPaqSurtCli
                .CantPzasSurt = .CantPzasSurt + CantPzasSurt

            End With
        End If

        F1DetArt.Item(3, I).Value = 0
    Next I

End Function
Private Function VerifVal() As Boolean
    VerifVal = False
    If Ped.Suc(TxtIdSuc.Text).CantCajasSinArt <> 0 Then
        Exit Function
    End If
    If Ped.Suc(TxtIdSuc.Text).Caja(IdCaja).TotArt = 0 Then
        MsgBox("EL NÚMERO DE ARTÍCULOS EN LA CAJA: " & IdCaja & vbNewLine & "DEBE DE SER MAYOR A CERO")
        Exit Function
    End If
    VerifVal = True
End Function
Private Function VerCanEmp() As Boolean
```

```
        'Verifica que la cantidad de piezas surtidas en la caja sea divisible
        'entre las piezas por empaque solicitadas por el cliente.

        Dim I As Integer
        For I = 0 To F1DetArt.RowCount - 1
            If F1DetArt.Item(9, I).Value = "1" Then
                VerCanEmp = False
                MsgBox("LA CANTIDAD DE EMPAQUES CAPTURADOS" & vbNewLine & "DEBE DE SER MULTIPLO DE LA UNIDAD" &
vbNewLine & "DE EMPAQUE DEL CLIENTE." & vbNewLine & vbNewLine & "VER LINEAS ROSAS.", vbInformation, "Verificar")
                Exit Function
            End If
        Next I
        VerCanEmp = True

    End Function

#Region "Base de Datos"

    Private Function GuardaValores() As Boolean
        Dim Color As Integer
        Dim Cont As Integer
        Dim I As Integer
        Dim J As Integer

        Dim strsql As String
        On Error GoTo Mensaje_Error

        GuardaValores = True


        strsql = "BEGIN TRAN" & vbNewLine
        strsql = strsql & "DECLARE @NumCaja INT" & vbNewLine
        strsql = strsql & "DECLARE @IdConten varchar(20)" & vbNewLine
        strsql = strsql & "DECLARE @NumCajaXEtapa INT" & vbNewLine & vbNewLine

        'PedAlmSuc    ----------------------------------------
        InsertaSucursal(Ped.Suc(TxtIdSuc.Text), strsql)
        '-------------------------------------------------------
        For I = 1 To Ped.Suc(TxtIdSuc.Text).CajaCont
            'PedAlmDet -------------------------------------------------
            InsertaDetalle(Ped.Suc(TxtIdSuc.Text).Caja(I), I, strsql)
            '----------------------------------------------------------
            'PedAlmCajas -----------------------------------------------
            InsertaCajaCont(Ped.NumCli, TxtIdSuc.Text, strsql, I)
            '----------------------------------------------------------
        Next I
        'PedAlmGlob ------------------------------------------------------------
        strsql = strsql & vbNewLine
        strsql = strsql & "EXEC  sp_InsertaPedGlobalEtapas" & vbNewLine
        strsql = strsql & "          /*NUMPED   */  '" & Ped.NumPed & "',                    " & vbNewLine
        strsql = strsql & "          /*ORDCLI   */  '" & Ped.OrdCli & "',                    " & vbNewLine
        strsql = strsql & "          /*NUMCLI   */  '" & Ped.NumCli & "',                    " & vbNewLine
        strsql = strsql & "          /*NUMEMP   */  '" & NumEmp & "',                        " & vbNewLine
        strsql = strsql & "          /*NUMART   */   " & Ped.ArtTotPzasSurt & ",             " & vbNewLine
        strsql = strsql & "          /*FECINI   */  '" & Format(Ped.FecIni, "yyyy/MM/dd") & "'," & vbNewLine
        strsql = strsql & "          /*FECVEN   */  '" & Format(Ped.FecCan, "yyyy/MM/dd") & "'," & vbNewLine
        strsql = strsql & "          /*FECELABI */  '" & Format(Ped.FecElabIni, "yyyy/MM/dd HH:mm") & "', " &
vbNewLine
        strsql = strsql & "          /*OBSER    */  '',                " & vbNewLine
        strsql = strsql & "          /*TIPOPED  */  " & TipoPed & " ,   " & vbNewLine
        strsql = strsql & "          /*TIPOPREP */  " & TipoPrep & " ,  " & vbNewLine
        strsql = strsql & "          /*AGENTE   */  ''                 " & vbNewLine
        '-------------------------------------------------------------------------------------------
        strsql = strsql & "IF @@ERROR <> 0        " & vbNewLine
        strsql = strsql & "     BEGIN             " & vbNewLine & vbNewLine

        strsql = strsql & "         ROLLBACK TRAN " & vbNewLine
        strsql = strsql & "         Print 'Error en la carga de valores.'" & vbNewLine & vbNewLine

        strsql = strsql & "     END               " & vbNewLine
        strsql = strsql & "ELSE                   " & vbNewLine
        strsql = strsql & "     BEGIN             " & vbNewLine & vbNewLine

        strsql = strsql & "         COMMIT TRAN   " & vbNewLine
        strsql = strsql & "         Print 'Carga exitosa.'" & vbNewLine & vbNewLine
```

```
        strsql = strsql & "END"
        'Print #1, strsql
        'Close #1

        Clipboard.Clear()
        Clipboard.SetText(strsql)
        ConnSinDr(strsql)

        Exit Function

Mensaje_Error:
        MsgBox(Err.Number & " " & Err.Description, vbCritical, "Error en la carga. GuardaValores")
        GuardaValores = False
    End Function
    Private Function GuardaValoresInd() As Boolean
        Dim Color As Integer
        Dim Cont As Integer
        Dim ContCaja As Integer

        Dim I As Integer
        Dim J As Integer
        Dim NumLin As Integer

        Dim strsql As String
        'Dim Rs As New ADODB.Recordset

        On Error GoTo Mensaje_Error

        GuardaValoresInd = True


        ContCaja = Ped.Suc(TxtIdSuc.Text).CajaCont

        strsql = "BEGIN TRAN" & vbNewLine
        strsql = strsql & "DECLARE @NumCaja       INT" & vbNewLine
        strsql = strsql & "DECLARE @NumCajaXEtapa INT" & vbNewLine
        strsql = strsql & "DECLARE @IdConten varchar(20)" & vbNewLine
        strsql = strsql & "DECLARE @ExistGlob     INT" & vbNewLine & vbNewLine
        'PedAlmGlob              ----------------------------------------------------------------
        strsql = strsql & "SELECT @ExistGlob = COUNT(NumPed) FROM PEDALMGLOB WHERE NUMPED = '" & Ped.NumPed &
"'" & vbNewLine
        strsql = strsql & "IF @ExistGlob = 0          " & vbNewLine
        strsql = strsql & "     BEGIN " & vbNewLine
        strsql = strsql & "         INSERT INTO PEDALMGLOB (NumPed,OrdCli, NumCli, NumEmp, NumSuc, NumCajas,
NumPzas, NumPaqCli, NumPaqAlm, NumArts, FecIni, FecVen, FecElabIni, FecElabFin,Status, StatusFac, TipoPedido,
TipoPrep ) VALUES ('" & Ped.NumPed & "','" & Ped.OrdCli & "','" & Ped.NumCli & "','" & NumEmp &
"',0,0,0,0,0,0,'" & Format(Ped.FecIni, "YYYY/MM/DD") & "','" & Format(Ped.FecCan, "YYYY/MM/DD") &
"',GETDATE(),'1/1/1900',1, " & SinFact & ", " & TipoPed & ", " & TipoPrep & "  )" & vbNewLine
        strsql = strsql & "     END   " & vbNewLine
        strsql = strsql & "ELSE       " & vbNewLine
        strsql = strsql & "     BEGIN " & vbNewLine
        strsql = strsql & "         UPDATE PEDALMGLOB SET Status = " & PedGlobEnElab & "  WHERE NumPed = '" &
Ped.NumPed & "'" & vbNewLine
        strsql = strsql & "     END   " & vbNewLine
        'PedAlmSuc    ----------------------------------------
        InsertaSucursal(Ped.Suc(TxtIdSuc.Text), strsql)
        '-------------------------------------------------------
        'PedAlmDet    ----------------------------------------
        InsertaDetalle(Ped.Suc(TxtIdSuc.Text).Caja(ContCaja), ContCaja, strsql)
        '-------------------------------------------------------
        'PedAlmCajas  ----------------------------------------
        InsertaCajaCont(Ped.NumCli, TxtIdSuc.Text, strsql, ContCaja)
        '-------------------------------------------------------

        strsql = strsql & "IF @@ERROR <> 0          " & vbNewLine
        strsql = strsql & "     BEGIN               " & vbNewLine & vbNewLine

        strsql = strsql & "         ROLLBACK TRAN   " & vbNewLine
        strsql = strsql & "         Print 'Error en la carga de valores.'" & vbNewLine & vbNewLine

        strsql = strsql & "     END                 " & vbNewLine
        strsql = strsql & "ELSE                     " & vbNewLine
        strsql = strsql & "     BEGIN               " & vbNewLine & vbNewLine

        strsql = strsql & "         COMMIT TRAN     " & vbNewLine
```

```
        strsql = strsql & "         Print 'Carga exitosa.'" & vbNewLine & vbNewLine
        strsql = strsql & "END"

        'Print #1, strsql
        'Close #1


        Clipboard.Clear()
        Clipboard.SetText(strsql)
        ConnSinDr(strsql)
        'MConecta.CommandTimeout = 300

        'MConecta.Execute(strsql, , adExecuteNoRecords)
        Vacio = False
        'MaxCajasElab = ContCaja


        Exit Function

Mensaje_Error:
        MsgBox(Err.Number & " " & Err.Description, vbCritical, "Error en la carga. GuardaValores")
        GuardaValoresInd = False
    End Function
    Private Sub InsertaSucursal(ByVal Suc As ClsSucursal, ByRef strsql As String)
        'Inserta Sucursal
        strsql = strsql & "/* Inserta Sucursal -------- */ " & vbNewLine
        If Not (SucSurtida) Then

            strsql = strsql & "INSERT INTO PEDALMSUC  (NumPed, idSuc, NumEmp, NumCajas, NumPzas, NumPaqCli
,NumPaqAlm, NumArts, Obser)   " & vbNewLine
            strsql = strsql & "VALUES   (/*NUMPED  */ '" & Ped.NumPed & "',     " & vbNewLine
            strsql = strsql & "          /*SUC     */ '" & TxtIdSuc.Text & "', " & vbNewLine
            strsql = strsql & "          /*NUMEMP  */ '" & NumEmp & "',        " & vbNewLine
            strsql = strsql & "          /*SUCCAJA */  " & TotCajas & ",       " & vbNewLine
            strsql = strsql & "          /*SUCPZAS */  " & Suc.TotPzasSurt & "," & vbNewLine
            strsql = strsql & "          /*NUMPAQCLI*/ " & Suc.TotPaqCli & ",  " & vbNewLine
            strsql = strsql & "          /*NUMPAQALM*/ " & Suc.TotPaqDur & ",  " & vbNewLine
            strsql = strsql & "          /*NUMARTS */  " & CantArt & ",        " & vbNewLine
            strsql = strsql & "          /*OBSER   */ '')                      " & vbNewLine & vbNewLine
        Else
            strsql = strsql & "UPDATE  PEDALMSUC    " & vbNewLine
            strsql = strsql & "SET     NUMCAJAS  =  " & TotCajas & ",       " & vbNewLine
            strsql = strsql & "        NUMPZAS   =  " & Suc.TotPzasSurt & "," & vbNewLine
            strsql = strsql & "        NUMPAQALM =  " & Suc.TotPaqDur & ",  " & vbNewLine
            strsql = strsql & "        NUMPAQCLI =  " & Suc.TotPaqCli & ",  " & vbNewLine
            strsql = strsql & "        NUMARTS   =  " & CantArt & vbNewLine
            strsql = strsql & "WHERE   NUMPED    = '" & Ped.NumPed & "'     " & vbNewLine
            strsql = strsql & "AND     IDSUC     = '" & TxtIdSuc.Text & "'  " & vbNewLine & vbNewLine


        End If
    End Sub
    Private Sub InsertaDetalle(ByVal Caja As ClsCaja, ByVal ContCaja As Integer, ByRef strsql As String)
        Dim ContArt As Integer
        Dim ConsecArt
        'Inserta Artículos Pedido X caja
        ConsecArt = 1
        strsql = strsql & "/* Inserta Artículos -------- */ "

        For ContArt = 1 To Caja.TotArt

            With Caja.Art(ContArt)

                If .CantPzasSurt <> 0 Then
                    strsql = strsql & "INSERT INTO PEDALMDET"
                    strsql = strsql & "(NUMPED, IDSUC, CONTCAJA, CONSEC, FECELAB, CODART, CODARTP, NUMSOL,
NUMPZAS, NUMPAQCLI, NUMPAQALM, NumPzaXPaqCli, NumPzaXPaqAlm, OrdCli, IdEtapa)         "
                    strsql = strsql & "VALUES('" & TxtNumPed.Text & "',"        'NUMPED
                    strsql = strsql & "'" & TxtIdSuc.Text & "', "               'IDSUC
                    strsql = strsql & ContCaja & ","                            'CONTCAJA
                    strsql = strsql & ConsecArt & ","                           'CONSEC
                    strsql = strsql & "GETDATE(),"                              'FECELAB
                    strsql = strsql & "'" & .CodArt & "',"                      'CODART
                    strsql = strsql & "'" & .CodArtP & "',"                     'CODARTP
                    strsql = strsql & .CantPzasSol & ","                        'NUMSOL
```

```
                    strsql = strsql & .CantPzasSurt & ","                       'NUMPZAS
                    strsql = strsql & .CantPaqSurtCli & ","                     'NUMPAQCLI
                    strsql = strsql & .CantPaqSurtDur & ","                     'NUMPAQALM
                    strsql = strsql & Ped.ArtGen(.CodArtP).NumPzaXPaqCli & ","  'NumPzaXPaqCli
                    strsql = strsql & Ped.ArtGen(.CodArtP).NumPzaXPaqAlm & ",'" 'NumPzaXPaqAlm
                    strsql = strsql & .OrdCli & "',"                            'ORDCLI
                    strsql = strsql & Ped.IdEtapa & ")" & vbNewLine             'IdEtapa
                    ConsecArt = ConsecArt + 1

                End If

            End With

        Next ContArt


    End Sub


#End Region 'Base de Datos

#Region "Reportes"
    Private Sub ImprimeReportes()
        'Dim Color As Integer
        Dim Cont As Integer
        'Dim Fun As String

        Dim Cls As New ClsRepEtiCaja

        Me.Cursor = Cursors.WaitCursor
        'Printer.ScaleMode = 1
        'CodBar.PrinterScaleMode = Printer.ScaleMode
        If TipoPrep = PrepEtapas Then
            CallByName(Cls, "EtiquetaCaja" & Cad.RepTipo, vbMethod, TxtIdSuc.Text, TxtNomSuc.Text, 4, TotCajas,
"CodBar", Ped.Suc(TxtIdSuc.Text), Vacio)
        Else
            For Cont = 1 To TotCajas
                CallByName(Cls, "EtiquetaCaja" & Cad.RepTipo, vbMethod, TxtIdSuc.Text, TxtNomSuc.Text, Cont,
TotCajas, "CodBar", Ped.Suc(TxtIdSuc.Text), Vacio)
            Next Cont

        End If

        Me.Cursor = Cursors.AppStarting

    End Sub
#End Region 'Reportes

#End Region 'Salida

#Region "Texto"

    Private Sub TxtCodArt_LostFocus(ByVal sender As Object, ByVal e As System.EventArgs) Handles
TxtCodArt.LostFocus
        Dim I As Integer


        If TxtCodArt.Text = "" Then Exit Sub
        TxtNomArt.Text = ""
        With F1DetArt

            Select Case Len(TxtCodArt.Text)

                Case 5
                    If SucSurtida = False Then

                        If UCase(TxtCodArt.Text) = "(IMP(" Then
                            TxtCodArt.Text = ""
                            BtnImprimir_Click()

                        End If

                    End If

                Case 12, 13
```

```
                    For I = 0 To .RowCount - 1

                        If .Item(1, I).Value = TxtCodArt.Text Then

                            If CantArt < Cad.ArtXCaja Or .Item(5, I).Value <> 0 Or Cad.ArtXCaja = 0 Then

                                TxtNomArt.Text = .Item(1, I).Value & "   " & .Item(2, I).Value

                                If .Item(7, I).Value = .Item(8, I).Value Then

                                    TotArt = TotArt + 1

                                End If

                                PoneValores(I)



                            Else

                                MsgBox("La cantidad de artículos debe de ser <= " & Cad.ArtXCaja & vbNewLine &
"Favor de verificar.", vbExclamation, "Tope máximo de artículos")

                            End If

                            Exit For

                        End If

                    Next I

                Case Else

                    MsgBox("Verificar código" & vbNewLine, vbExclamation)

            End Select

            If .RowCount = I Then

                MsgBox("El artículo no se encuentra en el pedido." & vbNewLine & "Favor de verificar.",
vbExclamation, "Artículo inexistente")

            End If

        End With

        If TxtCodArt.Text <> "" Then
            Limpiar()

        End If

    End Sub

#End Region 'Texto

#Region "Controles"

    Private Sub BtnAbortar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnAbortar.Click
        Abortar = True
        Me.Close()
    End Sub
    Private Sub BtnSalir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnSalir.Click
        Me.Close()
    End Sub
    Private Sub BtnImprimir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnImprimir.Click
        GuardaImprimeValores()
    End Sub
    Private Sub BtnImprimir_Click()
        Throw New NotImplementedException
    End Sub
```

```
    Private Sub BtnLimpiar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnLimpiar.Click
        Limpiar()
    End Sub
    Private Sub BtnCajaSum_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnCajaSum.Click
        '*********************************************************************
        If (Cancel = True) Or (Vacio = False And TipoPrep <> PrepEtapas) Then
            MsgBox("No se puede aumentar el nùmero de cajas," & vbNewLine & "en reempaque", vbExclamation, "")
            Exit Sub

        End If
        IdCaja = TotCajas

        If MsgBox("¿Desea pasar a la caja " & TotCajas + 1 & "?" & vbNewLine & "No se va poder modificar la
caja:" & IdCaja, vbYesNo) = vbNo Then Exit Sub
        If EtiStatus.Text = "Abierto" Then
            If CargaClases() = False Then Exit Sub
        End If
        '************************************
        If EtiStatus.Text = "Abierto" And TipoPrep = PrepEtapas Then
            'If GuardaValoresInd = False Then Exit Sub

            'If TotCajas >= Ped.Suc(TxtIdSuc.Text).CajaCont Then

            '    CargaCaja(TotCajas + 1)

            'End If

            'ImprimeReportes(False)
            'CantArt = 0

        Else

            If TotCajas >= Ped.Suc(TxtIdSuc.Text).CajaCont Then
                CargaCaja(TotCajas + 1)
            End If

        End If



        TotCajas = TotCajas + 1
        IdCaja = TotCajas
        TxtCajas.Text = IdCaja & " / " & TotCajas

        '************************************

        If Ped.Suc(TxtIdSuc.Text).Caja(IdCaja).TotPzasSurt = 0 Then
            StatusCaja(True)
        Else
            StatusCaja(False)
        End If



        '  CargaGrid(IdCaja)
        'TxtCodArt.Enabled = True
        'TxtCodArt.Focus()
        '  PoneCeros_Click
        With F1DetArt
        End With
        StatusCaja(True)
        '    BtnImp.Visible = True

    End Sub

#End Region 'Controles

#Region "F1DetArt"

    Private Sub PoneValores(ByVal Ren As Integer)
        Dim DifEmp As Double
```

```
        With F1DetArt

            If .Item(3, Ren).Value = 0 Then CantArt = CantArt + 1


            Select Case .Item(9, Ren).Value

                Case Is < .Item(5, Ren).Value

                    MsgBox("La cantidad marcada es mayor a la cantidad pedida." & vbNewLine & "Favor de
verificar.")
                    Exit Sub

                Case .Item(5, Ren).Value
                    .Rows(Ren).DefaultCellStyle.BackColor = Color.Green

                    ' .Item(7, Ren).Value = 0

                Case Else

                    DifEmp = (.Item(9, Ren).Value - .Item(5, Ren).Value) / .Item(6, Ren).Value
                    If Fix(DifEmp) - DifEmp = 0 Then

                        .Rows(Ren).DefaultCellStyle.BackColor = Color.DarkBlue
                        .Item(10, Ren).Value = 0

                    Else

                        .Rows(Ren).DefaultCellStyle.BackColor = Color.Violet
                        .Item(10, Ren).Value = 1


                    End If

            End Select
            .Item(3, Ren).Value = .Item(3, Ren).Value + 1
            .Item(4, Ren).Value = .Item(5, Ren).Value * .Item(3, Ren).Value
            .Item(7, Ren).Value = .Item(7, Ren).Value + .Item(6, Ren).Value
            .Item(9, Ren).Value = .Item(8, Ren).Value - .Item(7, Ren).Value

            'F1DetArt.ClearSelection()
            If Ren <= 17 Then
                .FirstDisplayedCell = .Item(5, 0)
            Else
                '.TopRow = 1
                .FirstDisplayedCell = .Item(2, Ren - 3)
            End If
        End With

    End Sub
    Private Sub F1DetArt_SelectionChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles
F1DetArt.SelectionChanged
        If F1DetArt.SelectedCells.Count > 0 Then
            F1DetArt.ClearSelection()
        End If
        If SucSurtida Then
            TxtIdSuc.Focus()
        Else
            TxtCodArt.Focus()
        End If
    End Sub

#End Region 'F1DetArt

#Region "Limpieza"

    Private Sub Limpiar()
        TxtCodArt.Text = ""
        TxtCodArt.Focus()
    End Sub

#End Region 'Limpieza


End Class
```

XX. PROPUESTA DE TESIS

TITULO: Sistema De Control Y Procesamiento De Información Para Salida de Productos Terminados En Una Empresa Textil: Caso Real.

OBJETIVO: Mejorar la eficiencia en la salida de productos terminados, mediante el análisis y evaluación de los procesos administrativos y productivos de la empresa, implementando una base de datos a partir de la información recopilada.

TEMARIO

INTRODUCCION

Una empresa textil deseaba mejorar su proceso de salida de los productos del almacén de producto terminado. Se busca mediante el análisis de datos hacer una propuesta, desarrollar el proyecto, implementar la solución, y hacer una evaluación de resultados obtenidos.

DEFINICIÓN DEL PROBLEMA

No existen mecanismos de vinculación y control de procesos que permitan relacionar las salidas de producto terminado a la facturación, envío, he inventariado. Dificultando el control de dichos procesos así como el análisis de información.

Las salidas del almacén de producto terminado se realizan de forma manual, generando muchos tiempos muertos, procesos duplicados, así como devoluciones por errores. Ocasionándole una pérdida considerable de recursos a la empresa.

Durante años la empresa ha tratado de vincular y sistematizar sin éxito sus procesos. Se busca mediante la recopilación de información, hacer un sistema de bases de datos que permita resolver dicha problemática.

MARCO TEÓRICO

Definir una base de datos relacional, diseño conceptual, modelo de diseño entidad-relación, diseño lógico, y diseño físico, que nos permita analizar la información

Implementar la base de datos mediante un Sistema gestor de Bases de Datos SGBD, para el caso particular del sistema Microsoft SQL 2000 server.

Establecer los mecanismos de comunicación ActiveX Data Objects (ADO) y Open Data Base Conectivity (ODBC) para posibilitar el acceso a la información.

Realizar las funciones específicas, algoritmos de búsqueda y ordenamiento, si es necesario.

Modelo de clases.

HERRAMIENTAS QUE SE UTILIZARÁN

Sistemas operativos: WINDOWS 2003 SERVER, WINDOWS XP

Manejador de bases de datos: WINDOWS SQL SERVER 2000

Lenguajes de programación VB6

Generador de reportes: Crystal Reports

Excel

Código libre

Interfaz que conecte la base de datos con Excel

IMPLEMENTACIÓN DEL SISTEMA

Base de datos: Ejecución de la base de Datos ProscaiAux en el servidor DurexServer, con una capacidad inicial de 400 MB, en 2 archivos de 200 MB cada uno, ProscaiAux_Data.MDF donde se albergan todo los datos almacenados y ProscaiAux_Log.LDF donde se almacenan las transacciones. Asignación de seguridad y roles.

Conexión a la base de Datos: Configuración DSN para Microsoft SQL Server para usuarios mediante Open Data Base Connectivity (ODBC). Así como los procedimientos ActiveX Data Objects ADO para comunicarse con la base de datos. Generar archivos de inicio. *.INI (variables de conexión y función de conexión a base de datos…)

Queries: (Consulta Sucursal, Almacenamiento de pedidos Modificación de valores en el pedidos).

CONCLUSIONES

BIBLIOGRAFIA

TRABAJOS FUTUROS

Integración del sistema de salidas al modelo comercial, haciendo modificaciones en facturación, he inventarios de forma automática.

Extender el sistema de salida tanto al proceso productivo, como Departamento de Envíos para generar un control y seguimiento del producto desde materia prima hasta la entrega al cliente.

Medición y mejoramiento de la productividad

Control estadístico de procesos

Inventario de materias necesarias para realizar el seminario:

ALGORITMOS Y ESTRUCTURAS DE DATOS

BASES DE DATOS

LENGUAJES DE COMPUTACION

INGENIERIA DE SOFTWARE

SISTEMAS OPERATIVOS

BIBLIOGRAFÍA

Almería, U. d. (16 de 12 de 2011). *Web de Bases de datos (ITIG) .* Recuperado el 16 de 12 de 2011, de Web de Bases de datos (ITIG): http://indalog.ual.es/mtorres/BD/teoria.php

CHEN, P. P.-S. (1976). *The Entity-Relationship Model-Toward a Unified View of Data.* Massachusetts: Massachusetts Institute of Technology.

Codd, E. E. (1990). *The Relational Model for Database Management: V2 .* Reading, Massachusetts: Addison-Wesley.

Codd, E. F. (1985). Does Your DBMS Run by The Rules. *Computer World* .

Codd, E. F. (1985). Is Your DBMS Really Relational. *Computer World* .

Costa, D. C. (2001). *El modelo relacional y el algebra relacional.* Barcelona: UOC.

Date, C. J. (2001). *Introducción a los sistemas de bases de datos.* Pearson Educación.

Elmasri, R., & Navathe, S. B. (2002). *Sistemas de base de datos (Conceptos fundamentales).* DF: Addison Wesley.

Jardine, D. A. (1976). *The ANSI/SPARC DBMS model.* Montreal, Canada: North-Holland Pub. Co.

Paré, R. C. (2005). *Bases de datos.* Barcelona: Universitat Oberta de Catalunya.

Ramez Elmasri, S. B. (2003). *Fundamentals of database systems.* Boston, MA: Pearson Education.

Ricardo, C. M. (2004). *Databases illuminated.* Sudbury, MA: Jones & Bartlett Learning.

Short, S. (2002). *Building XML Web Services for The Microsoft NET Platform.* Washington: Microsoft Pres.

Stephens, R. (2009). *Fundamentos diseño de bases de datos.* Anaya Multimedia.

T. Teorey, D. Y. (1986 Vol 18 No. 2). A logical design methodology for relational databases using the extended E-R model. *ACM Computing Surveys* , 197-221.

Ullman, J. D. (1999). *Introducción a los sistemas de base de datos.* Prentice Hall.

Wes, M. (2010). *Developing High Quality Data (European Process Industries STEP Technical Liaison Executive).* London: Shell International Limited.

West, M. (2010). *Developing High Quality Data Models.* Burlington, MA: Morgan Kaufmann.