UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS PROFESIONALES "ARAGÓN"

"EXPERIENCIA PROFESIONAL EN SISTEMAS CONTABLES BANCARIOS"

T E S I S

QUE PARA OBETENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A:
R I C A R D O R U I Z J U Á R E Z

ASESOR:
M. en I. ARCELIA BERNAL DÍAZ.

FES Aragón

BOSQUES DE ARAGÓN, ESTADO DE MÉXICO MAYO 2015

Agradecimientos.

Es difícil encontrar el orden del listado de las personas a quienes quieres agradecer el apoyo para que se publicara este trabajo. Quiero hacer énfasis que el triunfo de uno es el triunfo de todos.

Agradezco el esfuerzo de mis padres para darme una Licenciatura, hoy veo ese esfuerzo al darles educación a mis dos hijos. Magdalena, donde quiera que te encuentres…gracias, mil. Francisco muchas gracias.

A mi esposa Adriana por todos los momentos de apoyo desde que nos convertimos en amigos. No solo me has dado dos hijos, sino dos esperanzas.

A mis hijos David y Luis, ambos son mi infancia perdida. Ustedes me enseñan más de lo que yo, a ustedes.

A Arcelia Bernal, por creer en este proyecto. Siempre recuerdo con agrado tus clases…pero recuerdo con más agrado tu amistad.

A mi gran amigo Carlos Alberto; sin tu amistad la carrera no habría sido la misma.

A Farha, por tus consejos y amistad.

A todos mis amigos de la Preparatoria en especial a Israel y Ulises. Y de la Universidad Jonathan, Claudia, Pablo, Jorge, Ricardo y Oswaldo; de cada uno aprendí mucho.

A mi familia política por su apoyo incondicional.

A la Preparatoria No. 3 “Justo Sierra”, a la ahora FES “Aragón” y a la Universidad Nacional Autónoma de México, por dar educación pública de calidad. Me han dado el conocimiento necesario para sobresalir profesionalmente.

Índice

Introducción

Uno de los objetivos del presente texto es explicar mi experiencia adquirida en el área de Sistemas Contables Bancarios; en seis años de trabajo dentro de una empresa dedicada al sector financiero y comercial, donde día a día he analizado procesos y flujos de programas, desarrollado y mejorado aplicaciones; además de tomar decisiones y administrado recursos para la entrega de productos terminados. Sin embargo, el principal objetivo de este escrito es ser utilizado para que generaciones presentes y futuras de la licenciatura de Ingeniería en Computación tenga un panorama más amplio del medio laboral de un Ingeniero en el área de desarrollo de Sistemas Contables Bancarios.

La licenciatura de Ingeniería en Computación esta direccionada, entre muchos otros sentidos, a la programación de Sistemas de Cómputo, la administración y explotación, diseño y análisis de las Bases de Datos; pero también contiene materias adicionales que nutren el desarrollo del alumno como "Administración, Contabilidad y Costos". Esta última asignatura enfocada para que el futuro Ingeniero en Computación tenga las herramientas necesarias para desarrollarse en el área contable.

En el actual campo laboral un Ingeniero en Computación no debe conformarse únicamente con el conocimiento de un lenguaje de programación, un nuevo sistema operativo o programa de cómputo. Un Ingeniero en Computación debe ser interdisciplinario y capacitarse en otra asignatura como Administración, Finanzas, Mecánica o Pedagogía por mencionar algunas; tener firmes conocimientos de las necesidades de una empresa y como resolverlas mejora los procesos en los que un Ingeniero en Computación intervine.

Desarrollar Sistemas Contables y trabajar en ellos ha sido una gran experiencia dentro de mi carrera profesional, ya que no solamente he adquirido conocimientos de programación de Sistemas sino también de Contabilidad. Los Sistemas Contables son altamente utilizados desde pequeñas empresas hasta grupos de empresas donde revisar sus estados financieros hace colocar comas y apostrofes en las cifras numéricas para leerlas correctamente. Día a día se convive con estos sistemas, en el depósito o retiro en un banco, en la compra de la despensa en el supermercado, en la adquisición de mercancías de una empresa o en el pago de impuestos, etc. Existe un gran número de oportunidades para un Ingeniero en Computación dentro de este ramo.

La distribución de los capítulos dentro de este trabajo está relacionada con los puestos laborales que he tenido dentro de la empresa:

El Capítulo 1 corresponde al puesto de Analista Interfaz Contable, donde conocí el sistema *AS/400*[1] e inicie mi carrera en los Sistemas Contables dando

[1] AS/400-*Aplication Sistem* por sus siglas en inglés, es un Sistema propio de IBM denominado de medio rango.

mantenimiento a un Sistema propio de la empresa. En la parte técnica mejoré exponencialmente mis conocimientos en Bases de Datos, especialmente en la explotación de la información a través de sentencias *SQL*[2] ya que proveía de información a las Áreas Contables.

El Capítulo 2 describe el puesto de Desarrollador de Sistemas AS/400, donde desarrollé programas en el robusto lenguaje *RPG*[3] en su versión *ILE*[4] y Lenguaje de Control (*CL*[5]). Di mantenimiento a procesos transaccionales y *batch*; donde la programación es completamente estructurada. En este puesto adquirí los conocimientos fundamentales de programación, es decir, desde la estructura de un programa hasta minimizar lo más posible los recursos dedicados a este. Asimismo trabajé brevemente con desarrollos en *PL-SQL*[6] en *ORACLE*[7], pretendiendo migrar los procesos del AS/400 a procesos en PL-SQL para la transición de tecnología.

Mi actual puesto es colocado en el Capítulo 3 Líder de Proyecto de Sistemas. En este capítulo describo mi cambio de área, de la parte bancaria a la parte comercial de la empresa, donde realizó un proyecto de un Sistema Fiscal, en este proyecto no he dejado de programar, sin embargo mi prioridad es la administración de recursos y la entrega de requerimientos en tiempo y forma; además definir a mi equipo de trabajo los procesos de negocio que se tienen que realizar. La información contable que hay que explotar se encuentra en *SAP*[8], por lo tanto he tenido la necesidad de adecuarme al lenguaje de programación *ABAP*[9] para la creación de procesos.

[2] SQL- *Structured Query Language* por sus siglas en inglés. Lenguaje Estructurado de Consulta. Lenguaje de programación orientado a las Bases de Datos.
[3] RPG-Report Program Generator. Lenguaje de programación desarrollado por IBM para generar reportes comerciales.
[4] ILE- Integrated Language Environment. Diseño de IBM para la integración con procedimientos escritos en otros lenguajes de programación.
[5] CL- Control Language. Lenguaje para interacción con el sistema AS/400.
[6] PLSQL-*Procedural Language/SQL*. Lenguaje Procedimental/SQL. Lenguaje de programación orientado a las Bases de Datos.
[7] ORACLE-Sistema de gestión de Base de Datos.
[8] SAP-Software para aplicaciones de negocios.
[9] ABAP-Lenguaje de programación orientado a aplicaciones SAP.

Capitulo 1. Analista Interfaz Contable

"La caminata más larga comienza con un paso"

Dentro de la empresa a la que pertenezco el puesto de Analista en el Área de Sistemas es el escalafón más bajo, en este se encuentran personas recién egresadas, no necesariamente egresadas de una carrera orientada a la computación, por ser una empresa dedicada a los servicios financieros y comerciales pueden encontrarse Contadores, Capturistas de Datos, Asistentes de Dirección entre otros.

1.1 Primer empleo

Lo primero que hay que hacer para trabajar dentro de una empresa es realizar una entrevista con el área de Recursos Humanos o con el responsable de la vacante. Para muchos jóvenes recién egresados aprobar la entrevista no es fácil, para otros, por cuestiones que me gustaría llamarlo suerte, pasar la entrevista es más fácil. En la segunda empresa a la que toque puertas fui contratado.

Una de las políticas que me gustaría agradecer y que generalmente utiliza la empresa es la contratación de jóvenes recién egresados y la forma de contratarlos, en mi caso, no me solicitaron tener un alto conocimiento de algún Lenguaje de Programación, Base de Datos o manejo de algún Sistema Operativo. El área que solicitaba la vacante me aplicó un examen de lógica de programación, en el cual se siguen flujos, se buscan valores en variables, se enuncian los problemas a resolver de una manera confusa. El resultado del examen supone el grado de abstracción de cada participante, sin embargo años después me di cuenta que aprobar el examen no aseguraba un alto grado de abstracción. Para la contratación de un recurso ya sea con o sin experiencia se requiere de tomar otros factores en cuenta que se logran visualizan en la entrevista; el desarrollo que espera de la empresa, la educación que tiene, el motivo de la incursión en el área a la que quiere pertenecer, etc.

La ventaja que se tienen al contratar personal recién egresado es la oportunidad de capacitarlo a tus necesidades. Se le brinda una oportunidad de hacer una carrera dentro de la empresa y de ser el caso de tomar cursos que posteriormente puede utilizar dentro de su currículum. Sin embargo, la contratación de jóvenes recién egresados tiene algunas desventajas, entre ellas la amplia rotación de personal por falta de entrega de resultados. Algunos jóvenes no logran adaptarse rápidamente ya que se requiere una amplia curva de aprendizaje en negocios complejos. Los resultados de una persona recién

egresada son considerablemente lentos con respecto a otra con dos o tres años de experiencia.

La ventaja de contratar personas con más experiencia es el conocimiento previo que tiene y que puede asociarse a las nuevas actividades dentro de la empresa. Por otro lado cada empresa tiene sus políticas, como por ejemplo horarios, días laborales, forma de pago, exigencia, etc. Un trabajador se acostumbra a estas políticas y al llegar a una nueva empresa puede estar o no de acuerdo con nuevas políticas, con esto, la deserción o los despidos por estas situaciones también se hacen presentes. Otra desventaja es el costo de una persona con experiencia, a veces las plazas con las que se cuentan no alcanzan para cubrir este costo.

1.2 Acercamiento a los Sistemas *iSeries*

Las primeras actividades que debía desempeñar dentro de mi trabajo era utilizando un Sistema *AS/400* ahora llamado Sistemas *iSeries*, este sistema es poco conocido entre los actuales Desarrolladores de Sistemas. Al ser un sistema propio de *IBM* y dedicado a soluciones empresariales no es utilizado como base de enseñanza de programación dentro de las aulas de clase como lo son los lenguajes *C*, *JAVA*, *SQL*, entre otros. El costo de sus equipos y licencias hacen casi imposible este hecho. Sin embargo existen en la red la modalidad de renta de espacios dentro de estos equipos.

Coloco la página de internet donde se puede registrar y posteriormente usar gratuitamente un equipo *iSeries*.

http://www.rzkh.de/

La inscripción gratuita contiene espacio en disco y recursos del equipo muy limitados, para obtener un mejor espacio y rendimiento, existen diferentes tarifas de renta.

Los servidores *iSeries* de *IBM* son servidores denominados de medio rango, son robustos pero que no tienen la capacidad de un mainframe, sin embargo por su arquitectura manejan y almacenan enormes cantidades de información.

El servidor *AS/400* contiene su propio sistema operativo llamado *OS/400*. La seguridad dentro del sistema está basada en la lógica de objetos. Esto significa que todo lo que hay en el sistema (programas, archivos de datos, colas de mensajes) es un objeto. "*Cada objeto tiene dos partes independientes: una parte descriptiva, que define las formas válidas de utilizar los datos; y una parte de datos, que sirve como aspecto funcional del objeto"* (Soltis, 2014). Una vez

definido el objeto las características de este no pueden cambiarse, lo cual hace difícil la creación de virus dentro de *OS/400*.

La mayoría de los *AS/400* ni siquiera tiene el tradicional administrador de bases de datos ya que no lo necesitan. El sistema realiza trabajos para su administración por su cuenta.

El sistema *AS/400* contiene su propia Base de Datos, la cual es *DB2/400*. Dentro de *DB2* se maneja el término de Archivos de la Base de Datos, los cuales son divididos en Archivos Físicos y Archivos Lógicos. "*Los Archivos Físicos contienen los datos reales mientras que los Archivos Lógicos permiten que un usuario tenga acceso a datos en un formato que sea diferente de la manera que se almacena en unos o más archivos físicos. El archivo lógico no contiene ningún expediente de datos. Contiene el número de registro correspondiente del expediente de datos en el archivo físico"* (Bernal, 2015). Con la evolución de la Base de Datos ahora es conocido como tablas e índices.

El lenguaje de programación nativo de los Sistemas *AS/400* es *RPG*. El *RPG* fue creado y diseñado para cubrir las necesidades empresariales. Una de las ventajas de este diseño es la integración con la Base de Datos. "*Si se quiere obtener un registro de una Base de Datos, basta con escribir una sentencia que lea el registro. Incluso si se quiere escribir una sentencia de SQL más compleja en el programa de RPG, se escribe la sentencia y se le indica dónde insertar o devolver datos en los nombres de variables de RPG. Es fácil, algunos lenguajes requieren llamar a un complejo número de funciones de las API para SQL. Otros exigen utilizar números de columnas de SQL ordinales en vez de simples nombres de variables"* (Soule, 2014).

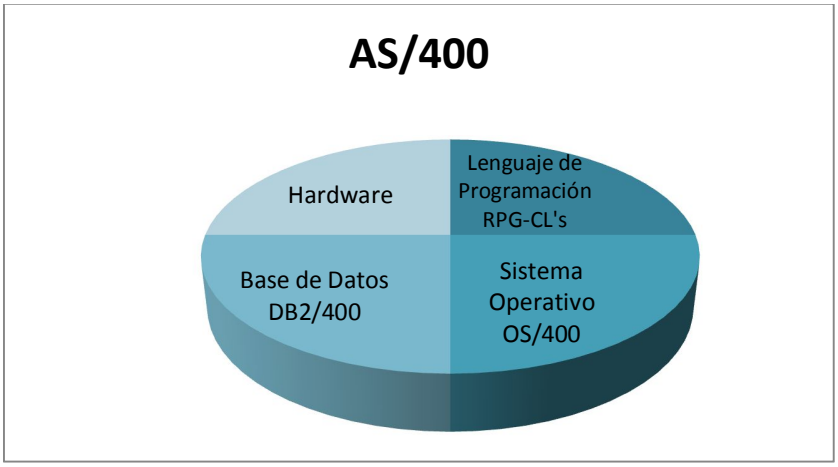

Figura 1.1 Elaboración propia (2014). Segmentos de un sistema AS/400.

Una de las desventajas mínimas de un Sistemas *iSeries* es la interfaz con el usuario, no es nada amigable, para la reducción del uso de recursos la interfaz

está basada en caracteres, no es común el manejo del sistema operativo con el *mouse*. El manejo de sistema operativo es a través de comandos seleccionados o escritos por el teclado. No es un sistema dedicado al usuario final, por lo tanto se requiere la creación de interfaces gráficas para su interacción con aplicaciones *WEB.* Sin embargo, he tenido la oportunidad de ver empresas bancarias y del ramo farmacéutico lo utilizan sin una Interfaz Gráfica intermedia para la atención de sus clientes y venta de mercancías.

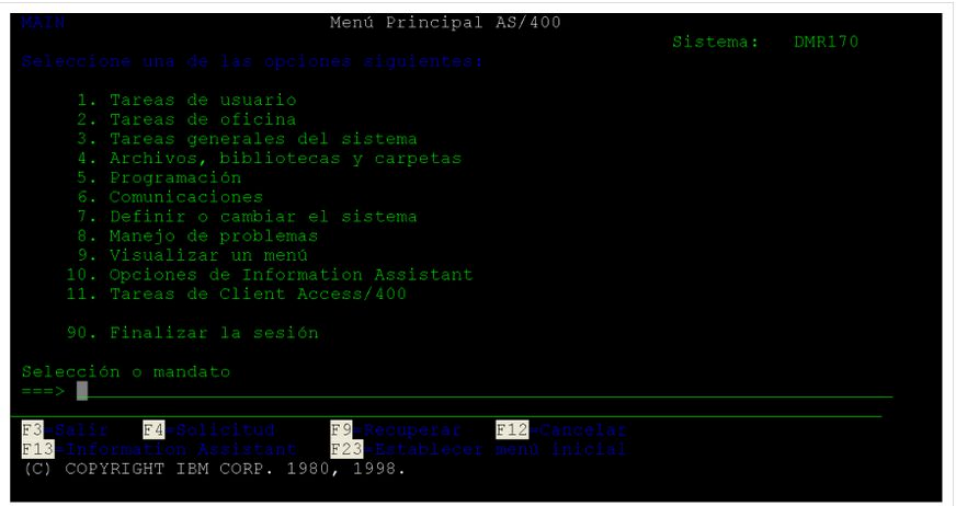

Figura 1.2. INBASCON (2015) Menú de inicio del sistema AS/400. Recuperado del sitio: http://inbascon.es/desarrollo/as400/.

1.3 Manejo de SQL dentro de los sistemas *iSeries*

Una de las herramientas más potentes y de fácil uso que tienen las Bases de Datos es el *SQL* (*Structured Query Language* en sus siglas en inglés o Lenguaje de Consulta Estructurado). El *SQL* es un lenguaje declarativo muy parecido al lenguaje natural, concretamente al inglés. Creado para tener acceso a las bases de datos. "*En 1982 IBM presentó la primera versión de DB2, que incluía soporte para SQL"* (Conte, 2014).

Como comentaba en el tema 1.2 dentro de la base de datos *DB2/400* los datos reales se guardan en Archivos Físicos, sin embargo la tendencia hacia los manejadores de la Base de Datos tradicionales estos se han ido sustituyendo por la creación de Tablas y los Archivos Lógicos por Índices.

El sistema *AS/400* cuentan con herramientas para la explotación de la Base de Datos tales como *STRSQL* (*STAR SQL*) o *RUNQRY* (*RUN QUERY*), y los programas escritos en *RPG-ILE* tienen la capacidad de insertar sentencias *SQL* complejas para optimizar el código. Hasta hace algunos años la utilización de sentencias *SQL* dentro los programas en *RPG-ILE* no era lo más

recomendable porque *SQL* no es el lenguaje nativo dentro del *iSeries*, sin embargo *IBM* ha modificado el Optimizador de la Base de Datos para orientar el acceso a la información un entorno más enfocado a las Bases de Datos tradicionales. El optimizador de una Base de Datos es la herramienta con la cual se interpreta una sentencia *SQL* y se busca el acceso a los datos más eficiente.

Coloco el siguiente ejemplo de una sentencia *SQL*:

SELECT CAMPO1, CAMPO2, CAMPO3 FROM LIBRERÍA1/TABLA1

En la sentencia *SELECT* de *SQL* se están eligiendo 3 campos que pertenecen a la TABLA1 (Archivo) que su vez pertenece a la Librería1 (Esquema). El Optimizador de la base de datos *DB2/400* automáticamente busca la ruta más corta para la lectura de la información; esto en base a los accesos que tienen la tabla así como estadísticas que va grabando de acuerdo a la recurrencia de las sentencias realizadas y a la cantidad de registros que contiene la tabla. El Optimizador tiene la capacidad de reescribir la sentencia, es decir, colocar los campos seleccionados en diferente orden, utilizar los campos llave que se integran en la sentencia o en la cláusula *WHERE,* además de cambiar el orden de las uniones (*join)* entre tablas para minimizar los recursos en la sentencia.

SQL abre la tabla, recupera la información, cierra la tabla y despliega la información. Con *SQL* no es necesario preocuparse por especificar todo lo anterior, simplemente se escribe una sentencia de un renglón y el resultado es desplegado o se puede grabar en otra tabla a través de una sentencia *INSERT*.

No así, si se quiere recuperar la misma información a través de un programa escrito en *RPG*, primero se debe declarar el Archivo del cual se va a leer la información y el que va a recibir la información, este último se debe declarar de actualización y colocar un sobrenombre diferente al formato de registro para poder actualizarlo. Se debe definir si el archivo se abre implícitamente o explícitamente. Se indica desde donde se inicia a leer el archivo que contiene la información, si desde el principio del archivo o a través de una llave en cualquier lugar del archivo y hacia donde se lee, es decir hacia el principio o hacia el final del archivo. Una vez definido esto se hace un ciclo hasta copiar todo lo deseado.

Como se puede observar, la intervención del *SQL* es bastante sobresaliente, con solo una sentencia en *SQL* se puede ahorrar mucho en la programación o consulta de información. Mientras en *RPG* el acceso a los datos es muy marcado, es decir, se debe especificar cada acción a realizar sobre los archivos que contienen los datos.

La diferencia de usar *SQL* a *RPG* es el camino para conseguir la información. Mientras con *SQL* la búsqueda de la información la hace el Optimizador, con *RPG* el programador elige el camino que debe de llevar el

programa para la adquisición de la información. Con *RPG* la programación es más propia del programador.

1.4 Sistemas Paramétricos

Uno de los temas para mi más interesante de este capítulo es la Parametría de Sistemas. En la experiencia que tengo desarrollando Sistemas he notado y comprobado que tener catálogos en la Base de Datos donde se encuentra la información necesaria para que el sistema funcione es más eficiente con respecto a sistemas donde el programa contiene los datos a usar. Con esto no hay necesidad de modificar el código fuente.

Colocaré un ejemplo para explicar el párrafo anterior:

Uno de los cambios que me toco realizar fue el cálculo centralizado de la tasa del Impuesto al Valor Agregado (IVA) para un producto que provee la empresa. Es decir, aunque al día de hoy la tasa del IVA está homologada en todas las zonas del país, en 2010 la tasa del IVA correspondía al 16% en el centro y 11% en las zonas fronterizas. Sin embargo existía otro problema, si un crédito fue surtido antes de 2010, la tasa de IVA se tenía que respetar. Recordemos que al inicio del año 2010 el aumento de IVA se incrementó de 10% y 15% a 11% y 16% respectivamente.

No importando que el producto fuera vendido en la zona fronteriza, este tenía que respetar la tasa centralizada porque el servicio era dado desde el centro del país.

La ventaja más influyente que tenía el Sistema para realizar el cambio era la opción de manejar cálculos dinámicos que se creaban a través de fórmulas que se guardaban en la Base de Datos. Las fórmulas se creaban con sentencias SQL.

Para la creación de cálculos dinámicos era necesario primero obtener los datos estáticos. Defino los datos estáticos a los datos que no cambian dentro del proceso y que son grabados en tablas de la Base de Datos. Por ejemplo, para obtener el importe base y el IVA de un movimiento realizado en el producto centralizado, los cálculos estáticos son el importe del movimiento y la tasa de IVA de la agencia donde se surtió el producto. El proceso primeramente buscaba todos los datos estáticos y posteriormente les colocaba un identificador (id) previamente definido. Por otro lado la fórmula contenía el id de los datos estáticos para posteriormente sustituirlo por el valor de cálculo estático.

En este caso, se debía forzar a que la tasa de IVA siempre fuera de 15% o 16%. En el ejemplo el consecutivo 1 es el importe del movimiento realizado

mientras que el consecutivo 2 es la tasa de IVA. Para obtener la base, la fórmula se vería de la siguiente manera:

1 / (1+ ((CASE WHEN 2 < 15 THEN 2 + 5 ELSE 2 END) /100))

Desglosemos la fórmula:

a) 1, corresponde al primer datos estático.
b) /, es la operación a realizar para obtener el IVA.
c) (1+, se abre paréntesis para dividir las operaciones a realizar.
d) ((CASE WHEN, se puede traducir como “cuando”.
e) 2, corresponde al segundo dato estático.
f) <15, si es menor o igual al valor constante 15.
g) THEN, se puede traducir como “entonces”.
h) 2 + 5, es la suma del dato estático con el valor constante 5.
i) ELSE, se puede traducir como “de lo contrario”.
j) 2, corresponde al segundo dato estático.
k) END, es la palabra reservada para finalizar la sentencia.
l) /100)), es necesaria la división para obtener el porcentaje de la tasa de IVA.

Dentro de la fórmula tenemos un problema, ¿Cómo diferenciar los datos constantes de los datos estáticos, es decir, los cálculos estáticos los colocaré sombreados:

1 / (1+ ((CASE WHEN 2 < 15 THEN 2 + 5 ELSE 2 END) /100))

Para esto, la fórmula se grababa en una tabla que contenía campos dedicados para los id de tipo numérico y otros para las sentencias de tipo carácter:

Calculo1: 1
Sentencia1: / (1+ ((CASE WHEN
Calculo2: 2
Sentencia2: < 15 THEN
Calculo3: 2
Sentencia3: + 5 ELSE
Calculo4: 2
Sentencia4: END)/100))

De esta manera la sustitución de valores era más sencilla. Sustituyendo valores con un crédito surtido en frontera antes de 2010 con un movimiento de $600 se tiene lo siguiente:

600 / (1+ ((CASE WHEN 10 < 15 THEN 10 + 5 ELSE 10 END)/100))

Dentro del programa, esta fórmula se colocaba dentro de la siguiente sentencia *SQL*:

SELECT 600 / (1+ ((CASE WHEN 10 < 15 THEN 10 + 5 ELSE 10 END)/100)) FROM SYSIBM.SYSDUMMY1.

La tabla SYSIBM.SYSDUMMY1 es una tabla propia del sistema que contiene una sola columna con un registro, dedicada para pruebas. Ejecutando la sentencia *SQL* el resultado es: 521.42. Para el cálculo del IVA, la fórmula se extiende:

(1 / (1+ ((CASE WHEN 2 < 15 THEN 2 + 5 ELSE 2 END)))) * ((CASE WHEN 2 < 15 THEN 2 + 5 ELSE 2 END)/100)

Sustituyendo los valores:

(600 / (1+ ((CASE WHEN 10 < 15 THEN 10 + 5 ELSE 10 END)))) * ((CASE WHEN 10 < 15 THEN 10 + 5 ELSE 10 END)/100)

El resultado es 78.26.

Sin la necesidad de hacer un cambio al programa y por medio de la parametría el IVA se calcula con la tasa centralizada.

Tomando en cuenta la explicación de los Sistemas Paramétricos, ahora lo complementaré con las actividades realizadas en mi trabajo.

La modificación de los datos lo hacía a través de Mantenimientos dentro del iSeries. Un mantenimiento es un programa que contiene una vista gráfica donde se despliegan datos, los cuales al cambiar, eliminar o insertar afectan directamente a una o varias tablas. Las vistas gráficas se crean en *RPG* con archivos *SUBFILES*. "*Un SUBFILE permite manipular múltiples registros del mismo tipo en la misma pantalla"* (IBM, 2006).

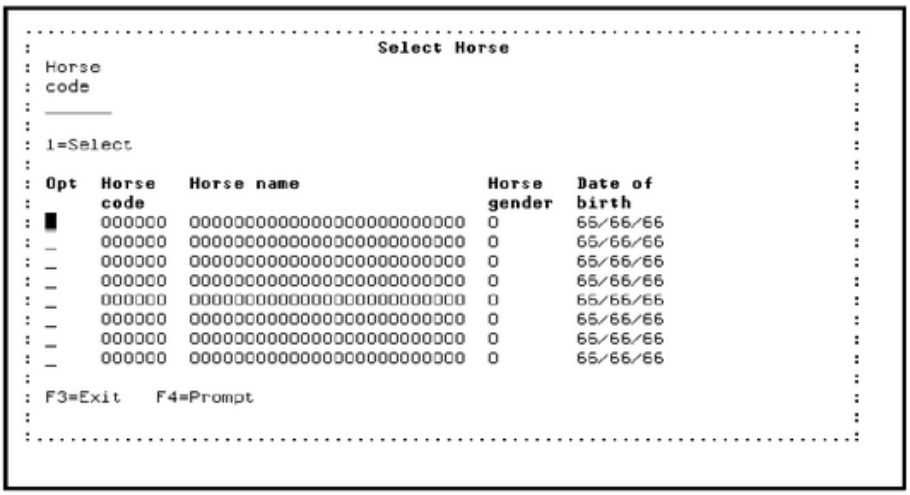

Figura 1.3 SUPPORT (2014).Representación gráfica de un archivo SUBFILE. Recuperado del sitio: https://support.ca.com/cadocs/0/CA%202E%20Release%208%207-ENU/Bookshelf_Files/HTML/Tutorial%20Guide/index.htm?toc.htm?1539943.html?intcmp=searchresultclick&resultnum=752.

El uso de mantenimientos es muy útil y confiable en términos de integridad de los datos. Con los mantenimientos se tiene la capacidad de visualización de datos antes de grabar, condicionar por programa valores de un campo de entrada, alimentar más de una tabla con el mismo valor de entrada, manejo de mensajes informativos, de alerta y de error, textos explicativos dentro del mantenimiento, etc. Sin embargo una de las limitaciones que veo sobre los mantenimientos es el tiempo para hacer modificaciones si se maneja un volumen considerable de información. Por ejemplo, al modificar la forma de contabilizar la entrada de enganches en el Sistema Contable que utilizaba, solo se tenía que agregar una cuenta contable y modificar otra. Para el sistema que mantenía, esto significaba hacerlo una vez y copiarlo alrededor de 40 veces, ya que el manejo de productos que contemplaban enganche era la mayoría. Cada producto tiene sus propias particularidades, esto hace que además de copiar la información se tenía que modificar.

Con la exigencia de los cambios me di a la tarea de encontrar nuevas formas de hacer las modificaciones a los catálogos, la primera experiencia de hacerlo con *SQL* no fue la mejor que pase dentro de mi primer año de trabajo; al desconocer las tablas involucradas que manejaba el Sistema no hice los cambios pertinentes y mande a error un número considerable de asientos contables, lo cual fue motivo de una llamada de atención de mi entonces gerente.

Un segundo intento fue a través de macros que se pueden programar dentro del sistema *AS/400*. Las macros ayudan mucho, sin embargo no dejan de ser grabaciones que se hacen para tareas recurrentes y si por algún motivo el

mantenimiento cambia, se genera un error de entrada de datos o el sistema no responde a la misma velocidad de la macros, la tarea de la macros es interrumpida y los cambios en los datos no se efectúan satisfactoriamente.

Un tercer intento fue combinar *SQL* y macros. Poco a poco fui conociendo las necesidades del sistema y la lógica de programación de este, y fui adquiriendo más conocimientos de Base de Datos, la relación entre las tablas que conformaban los catálogos y los valores mínimos de cada campo para hacer funcionar el sistema. No fue fácil, pero encontré la forma de alimentar los catálogos lo más rápido e íntegro posible a través de SQL.

Otra actividad en la que manejaba *SQL* era la corrección de atributos de las pólizas contables. Cada registro que se manda al libro mayor contiene ciertos atributos. Si alguno de estos atributos se informaba de manera errónea entonces el Sistema lo rechazaba, sin embargo, si el Sistema podía traducir el registro lo tomaba como un registro insuficiente. El registro insuficiente se quedaba dentro del Sistema, por lo tanto debía mandar dos registros para solucionarlo. Uno para eliminar el apunte erróneo y otro con el apunte correcto. Si un registro era rechazado, solo tenía que mandar el registro correcto.

1.5 Introducción a los Sistemas Contables

Como comente en temas anteriores mis primeras actividades se resumían en modificar tablas dentro de la Base de Datos, sin embargo no tenía el conocimiento de que estaba modificando, sabía que estaba grabando o actualizando los atributos que conformaban las cuentas contables, pero no contaba con el conocimiento del significado de cada atributos, el valor que debía contener cada cuenta contable y como obtener dicho valor, además de la naturaleza de las cuentas, si tenía que ser cargada o abonada.

La siguiente actividad a la que fui asignado fue cuadrar apuntes contables, es decir, los apuntes contables que tenían un defecto. Voy a resumir lo más posible a que se refiere el cuadre contable.

La forma del crédito dentro de la empresa en la que laboro es sobre Saldos Globales, esto significa que al inicio del crédito se fincan los Intereses con respecto al Capital del préstamo solicitado, los Intereses no cambian a lo largo de la vida del Crédito, si existe algún atraso por parte del cliente hay penalizaciones, a los que llamaremos intereses moratorios, sin embargo los intereses pactados en un principio no se afectan. No pasa lo mismo con los créditos basados en Saldos Insolutos, donde cada cierto periodo los intereses se recalculan con respecto a la deuda que se tiene en ese momento, no con respecto a la deuda original. Esto es, mientras se disminuya la deuda los intereses también disminuyen.

Por ser saldos globales cada movimiento contable del crédito se debe dividir en capital e intereses, dependiendo de la proporción que tenga cada uno con respecto al total de la deuda, si operativamente existió un movimiento, este se debe ver reflejado dentro de la contabilidad, además de contener los importes correctos para cada cuenta contable. Cada noche se ejecutan procesos *batch* que identifican los movimientos Operativos y se revisan con respecto a los apuntes contables que se generan a lo largo del día transaccionalmente. Si un movimiento del crédito no cumplía con los requisitos de esta regla o mantenía una diferencia considerable entre la suma de los cargos y los abonos no se mandaba a los libros de Mayor, el apunte contable se quedaba pendiente para mí revisión manual y posteriormente se impactaban en el Libro antes del cierre contable mensual.

Además de modificar los Créditos descuadrados manualmente, se tenían que revisar la generación de estos apuntes y de ser necesario modificar los catálogos de parámetros para evitar más apuntes errores dentro de la contabilidad.

1.6 Cierres Contables

Cada principio de mes era necesario tener todos los apuntes contables en el libro mayor correspondientes al mes próximo anterior. Esto para que los contadores pudieran hacer sus reportes para la autoridad así como para reportes internos.

Mi participación dentro de esta actividad iniciaba aproximadamente de 7 a 5 días antes del último día del mes. Se organizaba una junta con todas las áreas involucradas de negocio para programar el calendario del cierre mensual. Varias áreas de sistemas también participaban.

El siguiente paso era reportar el cierre operativo de las agencias. Todas las sucursales tenían que haber mandado transacciones del primer día del siguiente mes. De tener un escenario diferente las acciones a tomar dependían de cada país. Mi participación en el área contable no se limitó solo a México. Participe en la parte contable de Guatemala, Honduras y Panamá.

La autoridad en Guatemala es más rigurosa, por lo tanto el día primero de cada mes todos los apuntes contables tenían que estar impactados en el libro mayor, no importando si el día primero del mes era un fin de semana o un día festivo, para Guatemala no existían días inhábiles. Si una sucursal se atrasaba hablaba vía telefónica con mi contacto en dicho país para que le diera seguimiento y me entregara el estatus. Dependiendo del problema que existía se tomaba la decisión de dejarla fuera del cierre contable, tomar el último día completo o esperar que la agencia mandara las transacciones faltantes.

Para Honduras y Panamá el proceso era menos prioritario, para ambos países el cierre contable podía extenderse hasta el día 7 de cada mes, por lo tanto si una sucursal tenía algún problema este podía pasar desapercibido.

Para México el escenario era completamente diferente, por la cantidad de información que se maneja y la dependencia de procesos; el día del cierre se extendía al segundo día hábil de cada mes. Si alguna sucursal tenía un problema al transmitir se comunicaba a los involucrados y en la mayoría de las ocasiones se decidía tomar el último día completo de la sucursal atrasada para hacer el cierre contable. Existieron diferentes motivos para que una sucursal en México no trasmitiera, fallas eléctricas, fallas de conexión de datos, el servidor fuera de servicios, etc. Pero la que más me sorprendió es por inundaciones que a veces sufren las agencias por los huracanes que azotan el territorio mexicano.

Lo siguiente que tenía que realizar era cargar toda la información pendiente de impactar en el libro de mayor. Al principio todo era manual; se solucionaba incidencia por incidencia. En la parte de Latinoamérica no tenía mucho problema, ya que el nivel de transacciones era mucho menor al de México, lo que minimizaba los errores. Sin embargo la cantidad de información en México es mucho mayor. Al ver la cantidad de tiempo que me tardaba en resolver las incidencias y que no me quedaba tiempo de dedicarme más que solo a eso decidí optar por hacer programas en *RPG-ILE* para resolverlos. Lo platique con quien entonces era mi gerente, al principio me negó el permiso. Sin embargo, me tome la libertad de crear algunos programas y después mostrárselos. Encontraba el denominador común de las incidencias y con un programa los resolvía. Debo admitir que al principio, como todo programador novicio tuve algunos errores, los cuales corregía manualmente o a través de *SQL*.

Una vez que adquirí más conocimiento dentro de la programación con *RPG-ILE* los cierres mensuales se empezaron a ser más sencillos, ejecutaba procesos para reparar las incidencias dejando las actividades manuales en menor proporción.

1.7 Curso de Mejores prácticas en SQL

El único igualador social es la educación.
Gabriel Quadri.

El crecimiento de la empresa se vio reflejado en la necesidad de comprar un equipo *AS/400* con más capacidad. La apertura de nuevos sucursales y la extensión a otros países dio como resultado más transacciones enviadas a central, datos a almacenar y con esto tiempo de procesamiento de los programas.

El equipo productivo anterior quedo como un espejo mientras el nuevo equipo quedo como productivo. Por la compra del nuevo equipo *IBM* otorgo cursos de capacitación a los empleados.

Mi primer curso financiado por la empresa fue de "*SQL Performance. Monotoring, Analysis and Tuning*". El curso fue impartido dentro de las instalaciones de la empresa por personal de *IBM* proveniente de los laboratorios de *Rochester, Minessota*. El curso comenzó con el uso del paquete *System i Navigator;* el cual es utilizado para la navegación entre los objetos de la Base de Datos. Además de visualizar sus atributos y poder hacer cambios a los objetos.

Sin embargo el tema principal de este curso fue el optimizador de la Base de Datos. El instructor comentaba sobre la forma de trabajar del optimizador de la Base de Datos, la creación y utilización de los índices dentro de una tabla; los tipos de índices que pueden ser creados dentro de *DB2*. Las estadísticas que se van guardando dentro del sistema para que el optimizador de la base de datos tome la decisión de uno u otro acceso para la recuperación de los datos dentro de una tabla.

Otro de los temas principales de este curso fue la optimización de consultas mediante la aplicación *Visual Explain* que viene instalado en paquete *iSeries for Windows*. *Visual Explain* esta direccionado a generar el detalle de las consultas *SQL*, medir los tiempos de respuestas, además de la revisión de los procesos internos que hace el optimizador de la base de datos para llegar al resultado.

La representación de las consultas las hace a través de iconos, cada icono representa diferentes acciones de acceso a las tablas e índices, es decir lo que la Base de Datos hace internamente. El caminos a seguir es evitar "*full access*" o acceso completos a una tabla, lo más importante es la recuperación de datos por una llave o un índice.

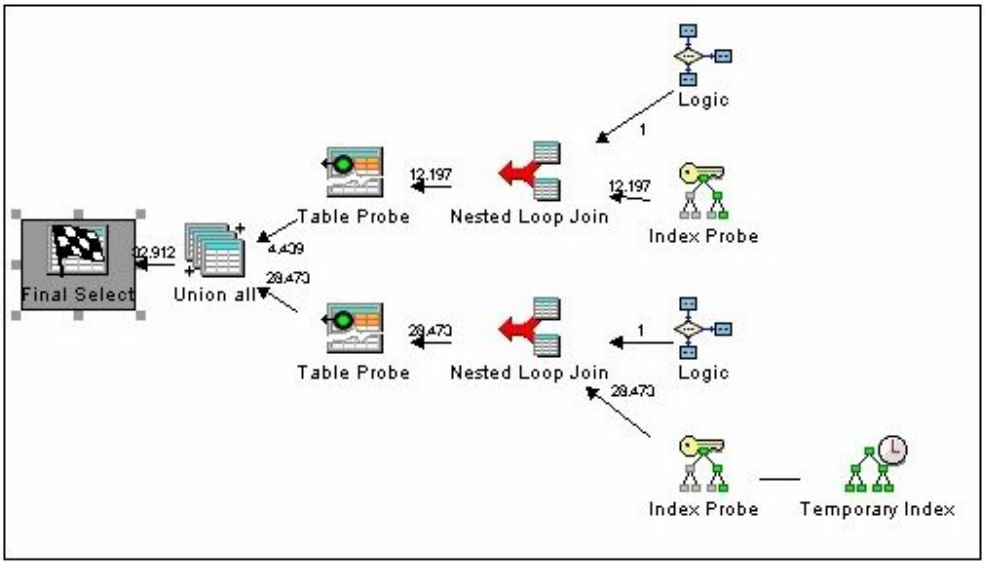

Figura 1.4. Sansoterra Michael (2014). Representación gráfica de sentencias SQL en Visual Explain. Recuperado del sitio: http://www.itjungle.com/fhg/fhg031914-story01.html.

Dentro del siguiente capítulo, mostraré como aplique los conocimientos adquiridos en mi etapa de Analista a la programación y al área contable, conocía la salida de los procesos, ahora debía conocer el cómo el proceso arrojaba los resultados. Tenía las herramientas necesarias para entender los procesos contables además del conocimiento técnico para desarrollar procesos eficientes.

Capitulo 2. Desarrollador de Sistemas AS/400

2.1 Breve Reseña Histórica de los sistemas *iSeries*

"Los componentes de un sistema operativo tienen que ver básicamente con el hardware subyacente. Un sistema operativo debe conocer los detalles del hardware."

El 21 de Junio de 1969 *IBM* anunció el Sistema/3, un sistema que por sus características definió nuevo segmento en el mercado, el de los miniordenadores. Pocos meses después, el 8 de enero de 1970, *Frank G. Soltis*, doctor en Ingeniería por la Universidad de Iowa, presentó por primera vez a la directiva de *IBM Rochester (Minnesota, EUA)* la idea para desarrollar miniordenadores con un novedoso diseño. En 1988, dieciocho años más tarde, dicho diseño, tras pasar por un largo periodo de evolución, se materializó en la plataforma informática *AS/400*, el entonces denominado *Application System/400*.

Tras obtener un doctorado en Ingeniería, especialista en arquitectura de ordenadores y diseño de sistemas operativo en la Universidad de Iowa, *Soltis* comenzó a trabajar a tiempo completo en un revolucionario sistema: el sucesor del Sistema/3 de *IBM*. En la práctica, el desarrollo de la organización de Sistemas avanzados que encabezaba *Soltis* no sustituyó al S/3, que siguió su propio desarrollo con la aparición de sus derivaciones, los Sistemas 32 y 34 en los años de 1975 y 1977, respectivamente. El equipo –que fue creciendo conforme avanzaba la década hasta convertirse en uno formado por cientos de personas– trabajó durante más de un lustro hasta lograr un sistema que cumpliera con los altos estándares que el diseño de *Soltis* requería.

Finalmente, el 24 de octubre de 1978 el nuevo sistema fue anunciado. Recibió el nombre de Sistema/38. La primera parte del trabajo había concluido. El Sistema estaba listo para ser utilizado en las empresas comerciales, una vez superadas las pruebas de confiabilidad y rendimiento teórico por lo menos en principio, ya que los primeros equipos no vieron la luz en el mercado hasta 1980 por graves deficiencias en el rendimiento con aplicaciones reales.

A continuación IBM intento combinar varias de sus tecnologías de rango medio y destruir la incompatibilidad entre ellas, este proyecto fue llamado Fort Knox, seria remplazar el S/36, S/38, Series/1, RS/6000 y pequeños mainframe. El proyecto no fue exitoso, e hizo al S/38 poco competitivo en el mercado.

"Muchos en Rochester –señala Soltis– creímos que [el proyecto Fort Knox] estaba condenado desde el principio, porque estaba tratando de resolver un problema de IBM en lugar de un problema del cliente. En IBM otros lo vieron como técnicamente demasiado grande para completarse: no había manera de hacer converger cinco sistemas diferentes y separados en uno. De igual manera,

otros pensaron que era imposible administrar el proyecto a través de cuatro laboratorios. Por todas estas razones, fracasó…".

El plan de contingencia de *IBM* fue el proyecto Silverlake, el cual tomo partes del S/36 y S/38 y algunos restos del proyecto Fort Knox y fue convertido en el sistema AS/400 que en realidad fue una nueva aplicación del S/38. Un grupo de desarrolladores de Rochester demostró la viabilidad de unir los S/36 y S/38, haciendo correr al primero como un ambiente del segundo. De esta manera, también podían construirse modelos más pequeños para cubrir todo el rango de productos del laboratorio.

Soltis recuerda: *"Una vez más, fue en Rochester donde se mostró a IBM y al mundo que podía hacerse. Después de un ciclo de desarrollo de 28 meses sin precedentes, estuvimos listos para anunciar el sistema de convergencia. Le dijimos al mundo que se trataba de un nuevo sistema y lo llamamos AS/400. Los de dentro, sin embargo, sabíamos que bajo las cubiertas de cada AS/400 se escondía un S/38"* (Soltis, 2014).

"El AS/400 fue un éxito inmediato. No sólo trajo de vuelta la parte de mercado perdida con numerosos competidores en años anteriores, sino que también sobrepasó a todos ellos con rapidez. Con más de 300,000 AS/400 corriendo en negocios de todo el mundo, el AS/400 es el sistema multiusuario más vendido de la historia. El culto al S/38 se ha convertido en una concurrida religión. Incluso para el AS/400 cambió el nombre del sistema operativo al de OS/400". (Blanch, 2014)

2.2 Conocimientos previos de programación

Desde el primer semestre en la carrera me llamo la atención la programación, aunque no tenía conocimiento previo antes de cursar la Universidad, las materias "Introducción a la Programación" y "Programación Orientada a Objetos" me ayudaron a desarrollar mi gusto por la generación de código fuente. A lo largo de la carrera tuve proyectos donde aplique la programación, algunos fueron un rotundo fracaso así como también otros un gran éxito. Es más fácil recordar los que tuvieron un gran éxito. El primero de ellos fue utilizar lenguaje Ensamblador para controlar el puerto paralelo (que ahora este puerto es ya obsoleto) y encender una serie de diodos emisores de luz. El siguiente fue controlar el puerto serie para conectar un Microcontrolador con una computadora personal. Pero el que recuerdo con más gusto fue simular una máquina despachadora de refrescos. A través del puerto paralelo obtuve señales de monedas que viajaban en un dispositivo diseñado para adquirir el valor de la moneda de acuerdo a su tamaño, una vez detectado el total del valor del refresco

con el mismo puerto paralelo movía un motor a pasos para hacer que la lata saliera. Este último proyecto lo realicé con *Borland C++*.

Al ocaso de la carrera me llamo la atención *JAVA*, con el cual trabaje para hacer gráficos. Inicié a estudiar este lenguaje para crear pantallas con *JAVA SWING*. Tuve la curiosidad de aprender a manejar la Base de Datos y explotarla con *JDBC*. Esto lo realice haciendo mis primeras prácticas con *MySQL*. Llenando con esto conocimientos mi *Curriculum Vitae* me decidí a buscar empleo.

Como ya lo había comentado previamente, en mi segunda entrevista de trabajo fui contratado. La vacante solicitada era para explotar información del sistema *AS/400* y quiero colocar la siguiente frase que me dijeron en esa entrevista: *"puedes hacer carrera en la empresa, los limites tú los pones"*. Al tener la necesidad de un empleo y el sueldo bastante prometedor para un recién egresado, tomé el empleo sin siquiera haber tenido un contacto con este sistema; pero con la promesa de capacitación por parte de la empresa. Poco a poco me di cuenta que los conocimientos que tenía de los lenguajes de programación no aplicaban en lo absoluto para la vacante, esto me hace recordar una gran frase en el área de programación de Leobardo López Román:

"La programación es lógica y debemos desarrollarla en los estudiantes, independientemente de algún lenguaje de programación; porque los lenguajes van y viene y la lógica ahí debe estar, para ser implementada en el lenguaje en turno".

Con esto quiero decir que me faltaba desarrollar la lógica de programación necesaria para adaptarme a cualquier lenguaje de programación; aunque cada lenguaje tiene sus características y utilidades; además de que cada programador tiene gustos por un área en específico de la programación.

Por varios factores tarde aproximadamente un año y medio después de entrar a trabajar para iniciar a programar en RPG. El principal factor fue la curva de aprendizaje que todo novato tiene que pasar para conocer las necesidades del negocio. No quiero maximizar mis logros, pero el área de Sistemas Contables es una de las más exigidas, ya que para hacer un cambio en los programas tienes que conocer todas las fórmulas, conceptos contables, dependencias entre módulos, flujos de negocio, entre otros. Todo lo anterior para mantener la integridad del sistema y de la información.

El segundo factor fue mi mala actitud para programar en un sistema que no pertenece a mi generación. Es decir, mientras que *JAVA*[10] tiene un *IDE*[11] como *NETBEANS*[12], *ORACLE* tiene a *TOAD*[13], *AS/400* tiene su *Source Entry Utility*

10 JAVA. Lenguaje de programación orientado a objetos

11 IDE. Integrated Development Environment. Ambiente Integrado de desarrollo, por sus siglas en inglés.

12 NETBEANS. Ambiente de Desarrollo Integrado para JAVA.

13 TOAD. Aplicación para desarrollo de aplicaciones SQL.

(*SEU*); donde se editan las líneas de código, las cuales deben ser posicionales, cada elemento dentro del programa tiene una posición ya establecida. Actualmente existe *RPG FREE* donde pueden colocarse las instrucciones y variables en cualquier posición dentro del código fuente. La inserción de nuevas líneas de código, copiar una o varias líneas, borrar líneas, mover líneas se hace a través de teclas rápidas.

Por otro lado su lógica de programación es sumamente diferente a los lenguajes que conocía. Voy a colocar los siguientes ejemplos, mientras que un programa escrito en *C* las instrucciones se van ejecutando sucesivamente como fueron escritas dentro del programa fuente y el programa termina cuando la última instrucción es ejecutada. En *RPG* existe "El ciclo de lógica de programa RPG", el programa itera hasta que se prende el indicador de último registro LR (*Last Record*, por sus siglas en inglés), en ese momento el programa termina.

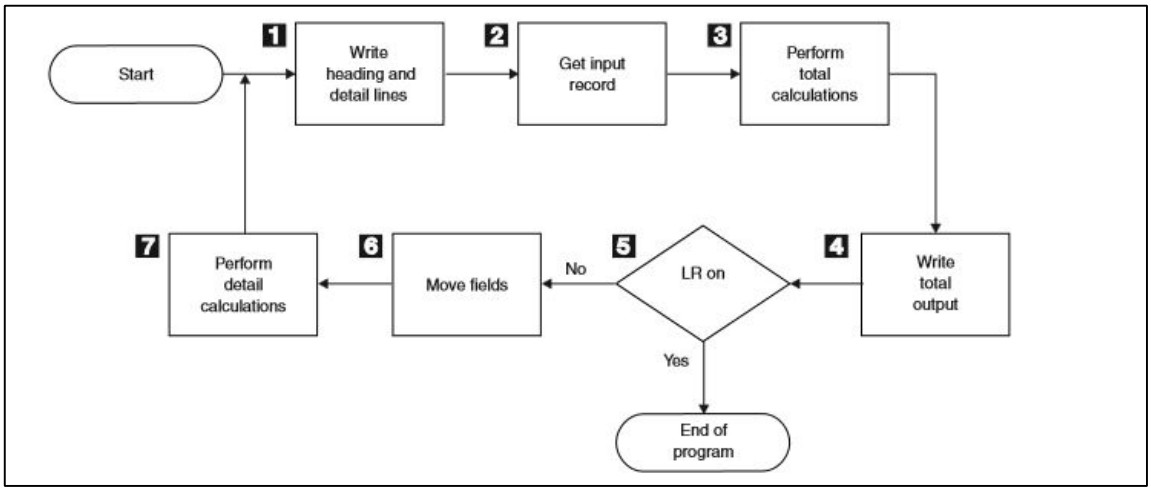

Figura 2.1 IBM (2001). Ciclo lógico de un programa RPG. WebSphere® Development Studio ILE RPG Reference pág.22.

"La primera y última vez del ciclo del programa difieren un poco de otros ciclos. Antes de leer el primer registro por primera vez mediante el ciclo, el programa efectúa tres cosas:

- *Maneja parámetros de entrada, abre archivos, inicializa datos del programa*
- *Escribe los registros condicionados mediante el indicador 1P (primera página)*
- *Procesa todas las operaciones de salida de detalle y de cabecera.*

*Por ejemplo, las líneas de cabecera impresas antes de leer el primer registro podrían estar formadas por información de cabecera de página o constante, o campos especiales como por ejemplo PAGE y *DATE. El programa*

también ignora los pasos de cálculos de totales y salida de totales en el primer ciclo.

La última vez que un programa pasa por el ciclo, cuando no hay más registros disponibles, el programa activa el indicador LR (último registro) y los indicadores L1 a L9 (nivel de control). El programa procesa los cálculos de totales y la salida de totales, a continuación se cierran todos los archivos y luego finaliza el programa" (IBM, WebSphere® Development Studio: Guía del programador en ILE RPG, 2001).

Hablo de los indicadores, sin embargo no los he definido aún. "*Un indicador es un campo de un carácter que está activado ('1') o desactivado ('0'). Normalmente se utiliza para indicar el resultado de una operación o para condicionar (controlar) el proceso de una operación. Los indicadores son como interruptores del flujo de la lógica del programa. Determinan la vía que tomará el programa durante el proceso, en función de cómo estén establecidos o de cómo se utilicen"* (IBM, WebSphere® Development Studio: Guía del programador en ILE RPG, 2001)

Por ejemplo, la instrucción *CHAIN* (*Random Retrieval from a File*, Recuperación Aleatoria de un Archivo).

Code	Factor 1	Factor 2	Result Field	Indicators		
CHAIN (E N)	search-arg	name (file or record format)	data-structure	NR	ER	_

Figura 2.2- IBM (2001). Representación del uso de indicadores en instrucciones RPG WebSphere® Development Studio ILE RPG Reference pág. 531.

El indicador *NR* (*No Record*) se coloca en 1 si ningún registro fue encontrado. Si existió algún error al leer el archivo el indicador *NR* se mantendrá en 0 mientras que el indicador de Error *ER* se colocar en 1.

Un tercer factor fue que contaba con conocimientos de Programación Orientada a Objetos, la cual está direccionada a la reutilización de código. *JAVA* tiene clases predefinidas que son utilizadas para la creación de nuevo código y de esta manera no tienes la preocupación de hacer algo que ya está hecho. Voy a poner el ejemplo de la creación de una ventana en *JAVA SWING*. Para mostrar una ventana en pantalla solo necesitas la siguiente línea de código:

```
import java.awt.*;
public class Ventana extends Frame
{
public Ventana()
{
super("Primera Ventana");
setLocation(100,100);
setSize(200,100);
```

show();
}

En cambio para desplegar una pantalla en *RPG* tienes la necesidad de definir el carácter con el que se requiere el contorno de la ventana. Como se puede deducir, estoy hablando de cosas distintas en todo tipo. Mientras *JAVA* tiene clases predefinidas, en *RPG* se tiene que configurar todo desde el principio, ya que su programación es totalmente estructurada.

2.3 Programación Estructurada

No quiero perder la oportunidad de dedicarle un pequeño espacio a esta metodología de programación, ya que aunque muchos creen que está en desuso o fuera de moda, muchas líneas de código son escritas anualmente con Programación Estructurada.

La programación estructurada se refiere al control de ejecución. El control de ejecución es una de las cuestiones más importantes que hay que tener en cuenta al construir un programa en un lenguaje de programación. La regla general es que las instrucciones se ejecuten sucesivamente una tras otra, pero diversas partes del programa se ejecuten o no dependiendo de que se cumpla alguna condición. Esta forma de programar se basa en el famoso teorema, desarrollado por *Edsger Dijkstra* en los años 60's, que demuestra que todo programa puede escribirse utilizando únicamente tres estructuras básicas de control siguientes:

- Secuencia: el bloque secuencial de instrucciones, instrucciones ejecutadas sucesivamente, una detrás de otra.
- Selección: la instrucción condicional con doble alternativa, de la forma "si condición entonces instrucción 1 sino instrucción 2"
- Iteración: el bucle condicional "mientras condición haz instrucción" o sus variantes.

Esta es la noción clásica de lo que se entiende por Programación Estructurada que hasta la aparición de la Programación Orientada a Objetos se convirtió en la forma de programar más extendida.

2.3.1. La segmentación en la Programación Estructurada

"La realización de un programa sin seguir una técnica de programación produce frecuentemente un conjunto enorme de sentencias cuya ejecución es compleja de seguir, y de entender, pudiendo hacer casi imposible la depuración

de errores y la introducción de mejoras. Cuando en la actualidad se habla de la programación estructurada, nos solemos referir a la división de un programa en partes más manejables (usualmente denominadas segmentos o módulos). A la visión moderna de un programa estructurado es un compuesto de segmentos, los cuales puedan estar constituidos por unas pocas instrucciones o por una página o más de código. Cada segmento tiene solamente una entrada y una salida, asumiendo que no poseen bucles infinitos y no tienen instrucciones que jamás se ejecuten, encontramos la relación entre ambas versiones el hecho de que los segmentos se combinan utilizando las tres estructuras básicas de control mencionadas anteriormente y, por tanto, el resultado es también un programa estructurado" (MCGrawHill, 2011).

La programación estructurada surgió como la solución de un problema: la creación de buenos programas; la aplicación de sus principios (disciplina en el estilo de programación) previene problemas desde las etapas de su diseño. Ya que una vez creado un software se debe de dar mantenimiento, modificación, actualización y depuración y los programas que lo conforman deben de ser más flexibles para minimizar los costos, donde costos no solo es a nivel monetario sino también a nivel de sistema, tener un tiempo de ejecución menor.

2.3.2. La programación Modular

Esta técnica de programación consiste en dividir un programa en partes bien diferenciadas lógicamente, llamadas módulos, que pueden ser analizadas y programadas por separado.

Un módulo se puede definir como un conjunto formado por una o varias instrucciones lógicamente enlazadas. A cada módulo se le asigna un nombre, que elige el programador, para poder identificarlo. Cuando en un punto de un programa se referencia un módulo, el programa le cede el control para que se ejecuten todas las instrucciones. Finalizando el mismo, el control se devuelve al punto del programa desde donde se llamó el módulo, y se continúa con la ejecución de la instrucción siguiente a la que realizo la llamada. El orden de ejecución está presentado en la siguiente figura:

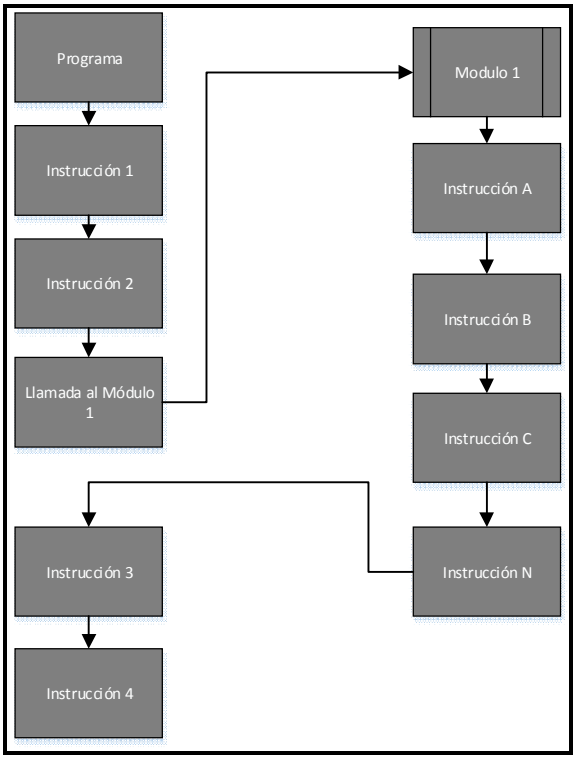

Figura 2.3. Elaboración propia (2014). Representación gráfica de la llamada a módulos

Si un módulo es lo suficientemente grande o complicado puede subdividirse en otros módulo, y estos, a su vez en otros, pudiendo representar un programa con diagrama similar al que se muestra en la figura 2.4

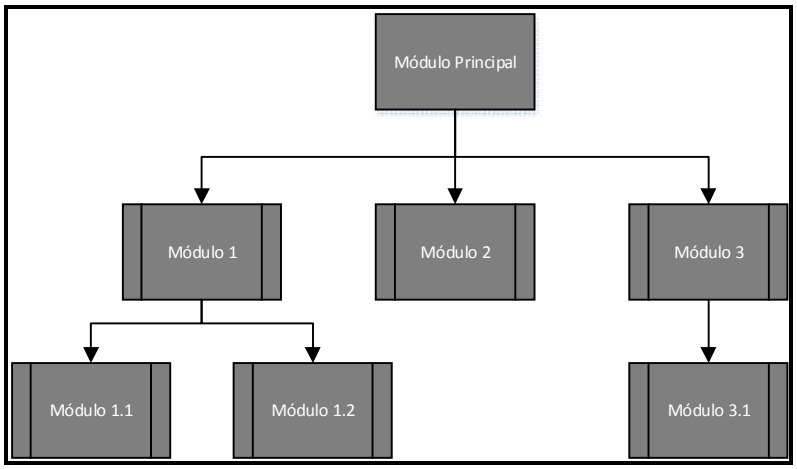

Figura 2.4. Elaboración propia (2014). Representación gráfica de módulos y sus raíces.

Como se puede observar en la figura 2.4 siempre debe existir un módulo raíz o principal, que se encarga de controlar y relacionar a todos los demás.

Si los módulos que componen un programa son parte integral del mismo, se les llama rutinas, subrutinas, procedimientos o funciones. Si por el contrario, son programas independientes, reclamados desde otros programas se les denomina subprogramas. No hay norma fija para dividir un programa en módulos. Se hace a criterio del programado. Sin embargo, se debe seguir las siguientes reglas:

- Cada módulo solo puede tener un punto de entrada y otro de salida.
- El Módulo principal debe ser conciso, mostrando claramente los módulos que lo componen y su relación. Es decir, debe indicar la solución completa del problema.
- Los módulos deben tener la máxima independencia entre ellos.
- Un módulo debe presentar por sí mismo una estructura lógica coherente y resolver una parte bien definida del problema.

La programación modular es una técnica que se basa en el desarrollo de programas de lo generar a lo particular, o dicho de otra forma, del conjunto al elemento. Diseño *TOP-DOWN*. Se comienza considerando que funciones debe realizar el programa desde un punto de vista muy general, dándolas como resueltas y dejando su diseño para un paso anterior. De esta manera se avanza hasta llegar al máximo nivel de detalle. La programación modular aporta una serie de ventajas frente a la programación convencional.

- Los programas son más sencillos de escribir y depurar, se pueden hacer pruebas parciales con cada uno de sus módulos.

- La corrección o modificación de un módulo se hace más cómoda y, en general, no tiene por qué afectar al resto de los módulos.
- Un programa se puede ampliar fácilmente con solo diseñar los nuevos módulos necesarios.
- Un mismo módulo escrito una sola vez puede ser referenciado desde varios puntos del programa, evitando la repetición de instrucciones ya escritas.

2.4 Mantenimientos

Se entendería por Mantenimientos a conservar los programas existentes, haciéndoles cambios para adaptarlos a las necesidades que se van presentando, sin embargo quiero definir los mantenimientos como programas que su función principal es la modificación a la información que se encuentra en la Base de Datos o en el Sistema.

2.4.1. Mantenimiento a Área de Datos

Dentro de mi labor de desarrollador inicié con la creación de pantallas y Mantenimientos. Uno de los primeros Mantenimientos que realice fue la creación de una pantalla para una *Data Area* (Área de Datos).

Un Área de Datos es una Estructura de Datos especial, la cual puede ser leída en cualquier programa. Esta Estructura de Datos la utilizábamos para colocar parámetros que afectaban el flujo de los programas; desde fechas, cadenas de caracteres o un simple indicador de una posición. Si teníamos un cambio extremadamente delicado lo condicionábamos desde el Área de Datos; de esta forma el cambio no entraba mientras nosotros no modificáramos el parámetro. Hacíamos esto porque nuestros procesos manejaban datos transaccionales y no teníamos el control de detener o liberar el flujo transaccional, además de no contar con un equipo de pruebas que fuera alimentado por las transacciones.

Al modificar el parámetro, el cambio entraba e iniciábamos con la revisión de los datos. Si el cambio no respondía con el resultado esperado apagábamos el parámetro e iniciábamos con la corrección de lo información afectada por el cambio.

Para visualizar un Área de Datos se tiene el comando *DSPDTAARA;* para hacer un cambio al Área de Datos se tiene el comando *CHGDTAARA*. Para hacer un cambio son necesarios los siguientes parámetros:

- Nombre del Área de Datos.

- Librería donde se encuentra el Área de Datos.
- Posición inicial de la estructura.
- Longitud del nuevo Dato.
- Nuevo Dato.

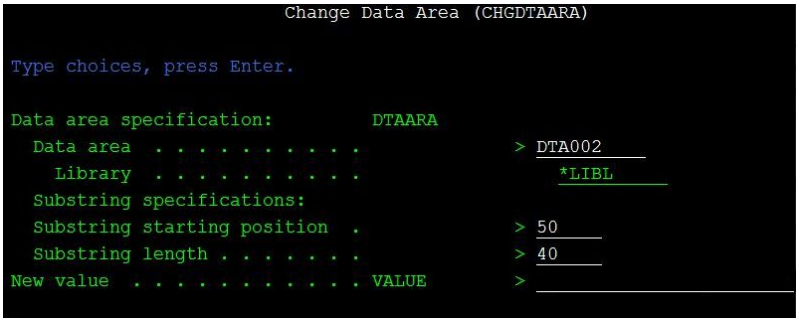

Figura 2.5. AS 400 Training (2012).Pantalla para modificar una Área de Datos. Recuperado del sitio: http://as400onlinecourse.blogspot.mx/2012/11/chgdtaara-in-as400.html

Sin embargo, el *OS/400* no despliega el Área de Datos con el comando *CHGDTAARA*, es necesario primero revisar con el comando *DSPDTAARA* para visualizar la posición a modificar. Por esta razón se me solicito una pantalla que diera la opción de visualizar el Área de Datos y dentro de la misma poder hacer los cambios necesarios a esta y visualizar los cambios hechos al instante

En la creación de una pantalla, el sistema tiene las opciones que un campo se pueda configurar obligatorio, condicionar los valores que puede tomar, manejar mensajes a presentar en la parte inferior de la pantalla. Sin embargo en este desarrollo aprendí que tiene más utilidad crear pequeñas ventanas con mensajes explicativos y más eficiente crear validaciones propias por programa.

El diagrama de flujo del programa es el siguiente:

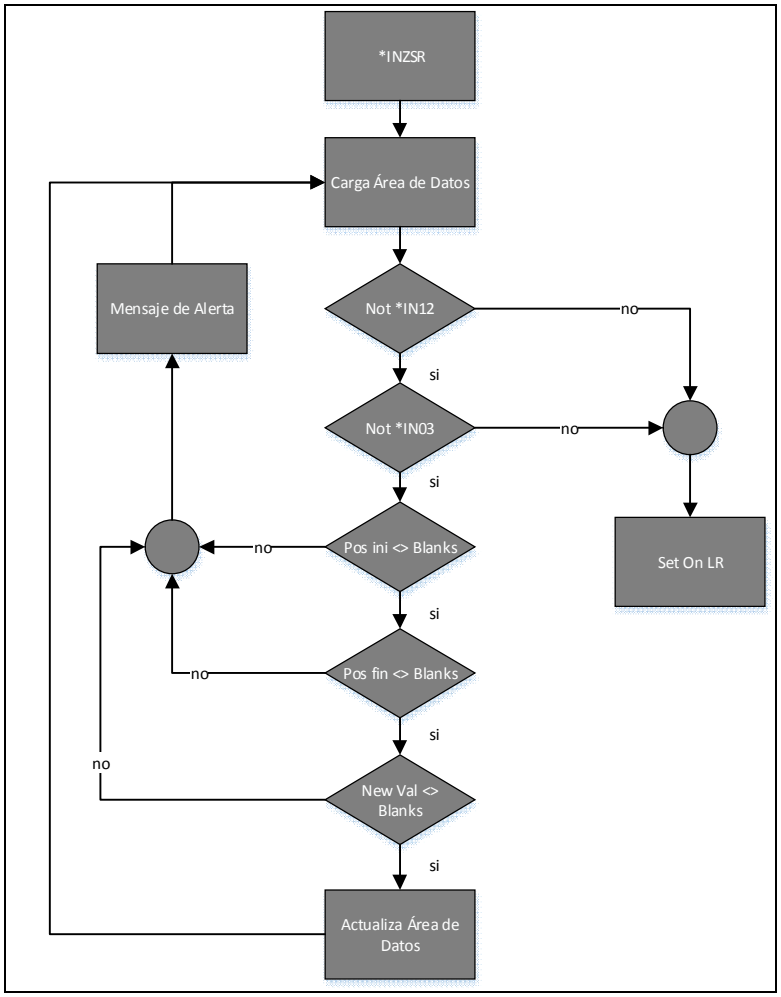

Figura 2.6. Elaboración propia (2014). Diagrama de representación del Ciclo Lógico de un programa RPG.

El diagrama inicia con la subrutina **INZSR*, la cual es una subrutina dedicada a inicializar valores. Los indicadores **IN03* e **IN12* son un estándar para salir de las aplicaciones. Si alguno se activa con la tecla *F3* o *F12* el usuario está abandonando el programa, esto se logra activado el indicador de último registro (*LR*). Como se puede observar existe un ciclo, mientras no se prendan los indicadores **IN03* e **IN12* el programa itera. Si se intenta hacer un cambio en el Área de Datos, es necesario que los parámetros se encuentren llenos, de lo contrario se visualizan los mensajes de alerta y se carga nuevamente el Área de Datos.

El ciclo mencionado no se programa dentro del código fuente, el ciclo está implícito y es natural en el *RPG*, es el "Ciclo lógico del *RPG*" mencionado con anterioridad, el cual si se sabe utilizar genera programas entendibles y pequeños. El diagrama de la Figura 2.6 es muy básico, solo se pretende mostrar la utilización del ciclo lógico de un programa *RPG*.

2.4.2. Mantenimiento a Tablas

Los mantenimientos a Tablas se logran a través de *SUBFILES*. "*Un SUBFILE es un grupo de registros que tienen el mismo formato y son leídos de y escritos a una pantalla en una operación. El siguiente ejemplo muestra un ejemplo de un SUBFILE"* (IBM, Application Display Programming, 1997)

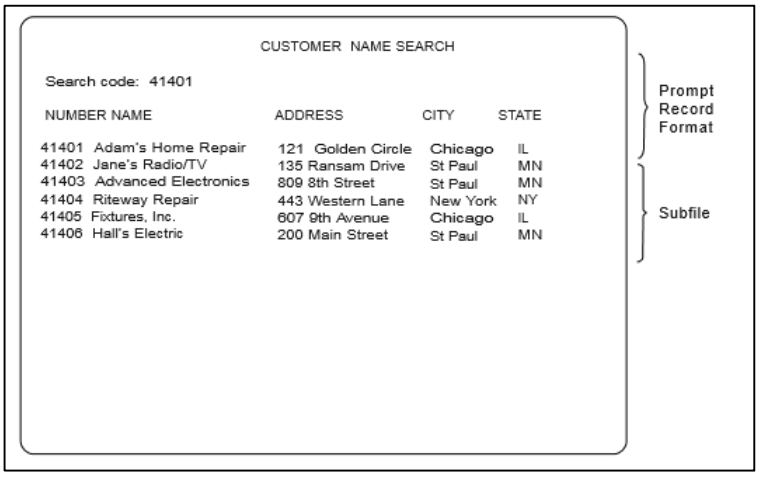

Figura 2.7. IBM (1997). Representación de una pantalla utilizando un SUBFILE. Application Display Programming pág. 134.

Para crear un *SUBFILE* es necesario crear dos tipos de formatos de registro: un formato de registro del *SUBFILE* y un registro de control del *SUBFILE*. El formato de registro define los campos en una fila del *SUBFILE*, el cual es utilizado para leer, escribir nuevos registros y actualizarlos.

El registro de control del *SUBFILE* contiene la información de cabecera y las funciones de control tales como tamaño, inicialización y clareado del *SUBFILE*.

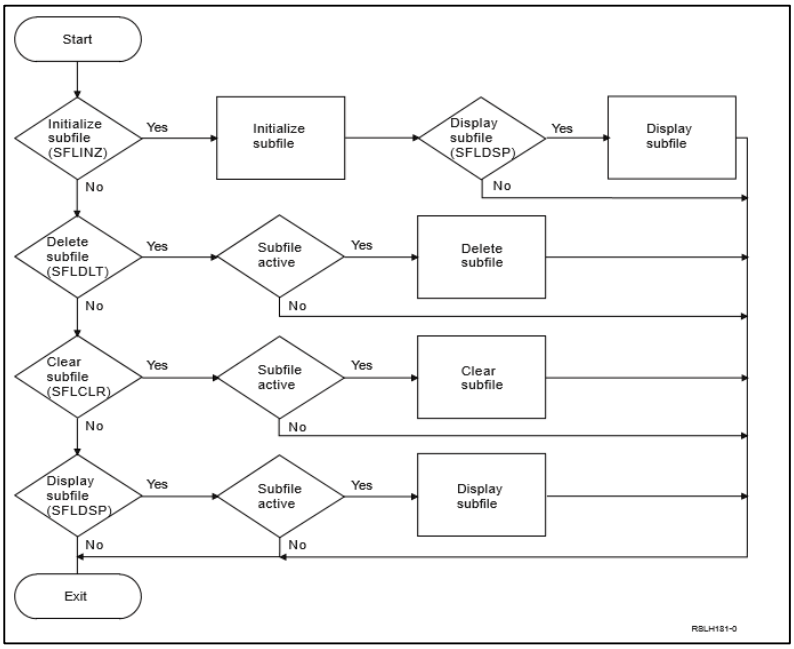

Figura 2.8. IBM (1997). Orden de procesamiento de palabras clave para el control de SUBFILE. Application Display Programming pág. 142.

Comento las palabras reservadas de los *SUBFILE* porque gran parte de los programadores no las conoce, en vez de limpiar el *SUBFILE* cuando no encuentran información a retornar me toco dar mantenimiento a programas donde mandaban una pantalla vacía para imitar el clareado del *SUBFILE.*

Es posible que un *SUBFILE* sea utilizado dentro de un programa *RPG*, el cual obtiene la informa la Base de Datos para ser presentados dentro del *SUBFILE*, además es posible leer los datos del *SUBFILE* y con esto valores escribir o actualizar dentro de la Base de Datos.

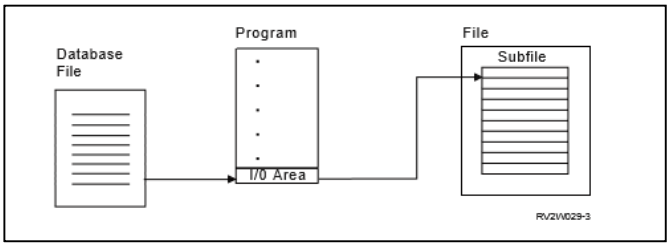

Figura 2.9. IBM (1997). Representación de uso de SUBFILE en un programa Application Display Programming pág. 143.

2.5 Programas, Procedimientos Almacenados y la *WEB*

Existe otra posibilidad de generar mantenimientos a tablas o presentar la información que arroja un programa y es a través de un servicio *WEB.* Para esto es necesaria la creación de un *External Stored Procedure*, ya que una aplicación externa no puede mandar a llamar a un programa *RPG* directamente y este último devolver los datos necesarios a quien lo llama.

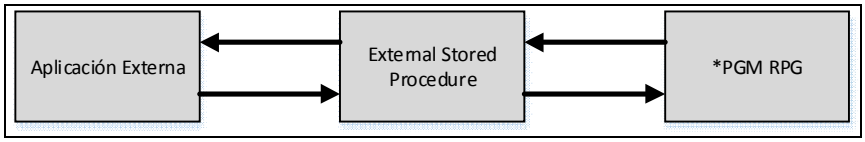

Figura 2.10. Elaboración propia (2014). Utilización de procedimientos almacenados externos.

En *DB2* es posible escribir dos tipos de procedimientos almacenados:

- *SQL Stored Procedures.* Son escritos en lenguaje *SQL*. Esto facilita implementar procedimientos almacenados de un sistema gestor de base de datos al sistema *iSeries* y del servidor *iSeries* a otro sistema gestor de base de datos.
- *External Stored Procedures.* Los procedimientos almacenados externos son escritos por el usuario en uno de los lenguajes de programación en el Servidor Host. Una vez compilado, el procedimiento almacenado ejecuta el programa objeto. El siguiente ejemplo el procedimiento almacenado retorna el nombre de un proveedor con las mayores ventas con un ejercicio y periodo dado.

```
c/EXEC SQL
c+ CREATE PROCEDURE HSALE
c+                (IN YEAR INTEGER    ,
c+                 IN MONTH INTEGER   ,
c+                 OUT SUPPLIER_NAME CHAR(20) ,
c+                 OUT HSALE DECIMAL(11,2))
c+        EXTERNAL NAME SPROCLIB.HSALES
c+        LANGUAGE RPGLE
c+        PARAMETER STYLE GENERAL
c/END_EXEC
```

Figura 2.11. IBM (2006).Código fuente de un procedimiento almacenado externo. Stored Procedures, Triggers, and User-Defined Functions on DB2 Universal Database for iSeries pág.67

La información obtenida del procedimiento externo es presentada por una aplicación WEB. No solamente pueden regresar parámetros, al llamar a un procedimiento almacenado externo puede ser posible el envío de un ResultSet, devolviendo un arreglo o un cursor. Esto es posible con la instrucción "*set result sets*":

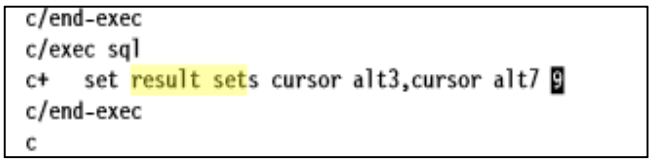

Figura 2.12. IBM (2006). Retorno de un Result Set a través de cursores. Stored Procedures, Triggers, and User-Defined Functions on DB2 Universal Database for iSeries pág. 147

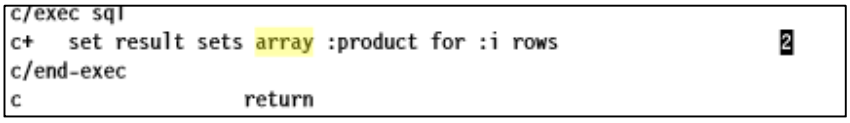

Figura 2.13. IBM (2006). Retorno de un Result Set a través de arreglos. Stored Procedures, Triggers, and User-Defined Functions on DB2 Universal Database for iSeries pág. 150

Uno de los primeros procesos que generé de esta forma fue para la presentación de información contable semanal para la revisión de los estados financieros. Sin embargo una de estos archivos se enviaba hasta el siguiente día, respetando la fecha contable del día domingo. Por esta razón, los usuarios los días lunes no podían ver en línea la información completa de las ventas semanales, faltaba la información que se enviaba hasta el día lunes. La información se tenía que complementar con lo que aún vivía en el servidor *AS/400* y no se había impactado en el libro mayor.

Los envíos de la información de los estados financieros se hacían manualmente, todos los días lunes se enviaba la información completa antes de la 13:00 horas. La solución dada fue crear un procedimiento externo que regresara la información de los estados financieros semanales en automático, esto para que los usuarios lo pudieran visualizar en un servicio WEB, y no necesitáramos estar presentes para el envío de la información. Aclaro se enviaba la información no importando que fuera un día festivo.

2.6 La Programación Estructurada y la Programación Orientada a Objetos

Las técnicas de programación constituyen una parte fundamental en el proceso de desarrollo e ingeniería de software, por ejemplo la programación estructurada y la programación orientada a objetos, cada técnica tiene sus propias características y distintos métodos de resolución de problemas. Es de gran importancia para los ingenieros en computación aprender a implementarlas al momento de adentrarse en cualquier proyecto de desarrollo del software.

Uno de los proyectos donde pude interactuar con programación estructurada y la programación orientada a objetos fue en el envío de información desde el sistema *AS/400* vía *FTP*[14] a un servidor *WINDOWS*. Tengo que aclarar que este proceso ya existía, lo que realice junto con mi Líder de Proyecto fue adaptarlo cuando se hizo un cambio en el servidor donde se depositaba la información y el protocolo de transferencia de red cambio de *FTP* a *FTPS*[15]. Además de analizar cada uno de los procesos ya que los códigos fuente no correspondían con los archivos objeto que se encontraban en el equipo productivo.

La información contable que se generaba a lo largo del día tenía que ser depositada en un archivo de texto plano dentro de un Servidor *WINDOWS*. ¿Cómo se logró hacer esta conexión y el depósito del archivo? *RPG* está diseñado para aplicaciones empresariales, por lo tanto no contiene, por sí mismo, herramientas necesarias para abrir conexiones a otros servidores con distinto sistema operativo. Para mitigar estas carencias los sistemas *AS/400* tiene la capacidad de desarrollar en diversos lenguajes de programación:

Integrated Language Environment (ILE)	Modelo del programa original (OPM)	Modelo del programa ampliado (EPM)
C++	BASIC (PRPQ)	FORTRAN
C	CL	PASCAL (PRPQ)
CL	COBOL	
COBOL	PL/I (PRPQ)	
RPG	RPG	

Figura 2.14. IBM (2001). Lenguajes soportados por un servidor AS/400. WebSphere® Development Studio: Guía del programador en ILE RPG pág. 19

Además de estos lenguajes, los *AS/400* tiene la capacidad de crear programas en *JAVA*. Esto se logra gracias al *AS/400 Developer Kit* para *JAVA*. "*El AS/400 Developer Kit está optimizado para su utilización en un entorno de servidor AS/400. Utiliza la compatibilidad de las interfaces de usuario y de programación JAVA para que el usuario pueda desarrollar aplicaciones propias para el sistema AS/400"* (IBM, AS/400 Developer Kit para JAVA, 1998, pág. 1).

El Sistema *AS/400* tiene la capacidad de contener archivos dentro de su Sistema de Archivos Integrados.

2.6.1. Sistema de Archivos Integrados

[14] FTP -File Transfer Protocol, por sus siglas en inglés. Protocolo de Transferencia de Archivos.

[15] FTPS- Es una extensión de FTP mediante SSL para el cifrado de los datos.

El sistema de archivos integrado es un componente de OS/400 que soporta la gestión de almacenamiento y entrada/salida continua de forma similar a los sistemas operativos *PC* y *UNIX*, proporcionando además una estructura de integración para toda la información almacenada en el *AS/400*. Las características clave del sistema de archivos integrado son las siguientes:

- Soporte para almacenar la información en archivos continuos que pueden contener largas series continuas de datos. Estas series de datos pueden ser, por ejemplo, el texto de un documento o los elementos de imagen en una imagen. El soporte de archivos continuos se ha diseñado para utilizarlo eficazmente en las aplicaciones cliente/servidor.
- Una estructura de directorios jerárquica que permite organizar los objetos como las frutas de las ramas de un árbol. Especificando la vía de acceso a un objeto a través de los directorios se accede al mismo.
- Una interfaz común que permite a los usuarios y a las aplicaciones acceder no sólo a los archivos continuos, sino también a archivos de base de datos, a documentos y a otros objetos almacenados en el *AS/400*.
- Una visión común de los archivos continuos almacenados localmente en el *AS/400*, en el *Integrated Netfinity Server* para *AS/400* o en un servidor *Windows NT* remoto. Los archivos continuos pueden estar almacenados también de manera remota en un servidor de red de área local (*LAN*), un servidor *Novell NetWare*, otro servidor *AS/400* remoto o un servidor de Sistema de Archivos de Red (*NFS*).

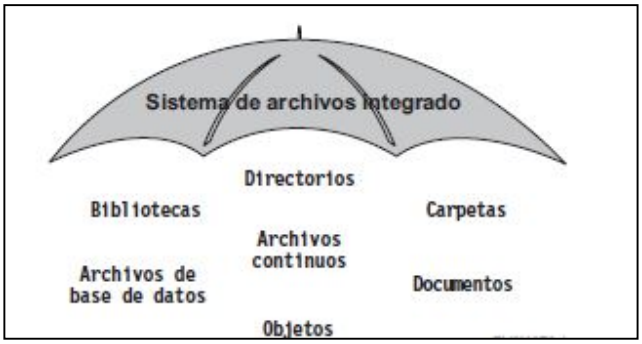

Figura 2.15. IBM (2000.)Estructura de la información almacenada en el AS/400. Introducción al sistema de archivos integrado pág.1

Los archivos *.JAVA* depositados dentro del Sistema de Archivos pueden ser compilados y crear archivos *.CLASS*. Una vez hecho esto pueden crearse los archivos *.JAR* para agrupar las Clases a utilizar y así definir la variable de entorno *CLASSPATH*. Todo esto desde el sistema *AS/400* gracias a la herramienta *QSHELL*.

2.6.2. *QSHELL Interpreter*

Una de las herramientas importantes que presenta el OS/400 es el *QSHELL (QSH)*. EL intérprete *QSHELL* (qsh) es un intérprete de comandos para el AS/400 que está basado en los estándares *POSIX*[16] y *X/open*[17]. Con *QSH* puede hacer lo siguiente:

- Correr comandos de cualquier sesión interactiva o de un archivo de comandos.
- Escribir scripts de *Shell* que se puedan correr sin modificaciones de otros sistemas.
- Trabajar con archivos en cualquier sistema de archivos soportado por el Sistema de archivos Integrado.
- Correr utilerías propias para extender las capacidades provistas por *QSH*.

Con la ayuda de este intérprete es posible ejecutar comandos de otros sistemas, para el caso de este ejemplo, las utilerías de *JAVA*:

- *ajar - Alternative Java archive.*
- *appletviewer - View Java applet.*
- *extcheck - A utility to detect JAR conflicts.*
- *jarsigner - Jar signing and verification.*
- *java - Run Java interpreter.*
- *javac - Compile a Java program.*
- *javadoc - Generate Java documentation.*
- *javah - Generate C header or stub file.*
- *javakey - Manage Java security keys and certificates.*
- *javap - Disassemble a compiled Java program.*
- *jar - Archive Java files.*
- *keytool - Key and certificate management tool.*
- *native2ascii - Convert native characters to ASCII.*
- *policytool - Policy file creation and management tool.*
- *rmic - Compile Java RMI stubs.*
- *rmid - The Java RMI activation system.*
- *rmiregistry - Start a remote object registry.*
- *serialver - Return serial version.*
- *tnameserv - Naming service.*

Una vez creados los archivos *.CLASS* y agrupados en archivos *.JAR* pueden ser utilizados dentro de programas *RPG*.

[16] POSIX es el acrónimo de *Portable Operating System Interface*; la X viene de UNIX como seña de identidad de la API. Estos son una familia de estándares de llamadas al Sistema Operativo definidos por el IEEE y especificados formalmente en el IEEE 1003.

[17] X/Open Company, Ltd. fue un consorcio fundado por varios manufacturadores europeos de UNIX en 1984 para identificar y promover el estándar abierto en el campo de la tecnología de la información

2.6.3. Ejecución de Clases *JAVA* desde *RPG*

Es posible mandar a ejecutar Clases *JAVA* desde programas *RPG*. Esto se logra utilizando *API's*. En la figura 2.16 se visualiza un ejemplo, la *API QCMDEXC* manda a llamar la Clase *Hello*:

Ejemplo 1: llamadas a Java desde RPG

```
D*            SE DEFINEN LOS PARÁMETROS DE LA API QCMDEXC
     D*
     DCMDSTRING        S             25    INZ('JAVA CLASS(''com.ibm.as400.system.Hello'')')
     DCMDLENGTH        S             15P 5 INZ(25)
     D*             AHORA SE LLAMA A QCMDEXC CON EL MANDATO CL 'JAVA'
     C                   CALL      'QCMDEXC'
     C                   PARM                    CMDSTRING
     C                   PARM                    CMDLENGTH
     C*       La siguiente línea visualizará 'DID IT' después de salir del
     C*       shell Java por medio de F3 o F12.
     C     'DID IT'      DSPLY
     C*       Se activa LR para salir del programa RPG
     C                   SETON                                        LR
     C
```

Figura 2.16. IBM (1998). Llamada de un programa JAVA desde RPG. AS/400 Developer Kit para JAVA pág. 122

2.6.4. Conexión a la Base de Datos con JDBC

El proceso también contemplaba una conexión a la base de datos a través de *JDBC* (*Java Data Base Connection*). "*Con el controlador JDBC de AS/400 Developer Kit para JAVA, los programas JAVA pueden acceder a archivos de base de datos AS/400, acceder a funciones de base de datos JDBC con Lenguaje de Consulta Estructurada (SQL) incorporado y ejecutar sentencias SQL y procesar resultados. JDBC es un componente estándar de JAVA y se incluye en AS/400. JDBC es la API JAVA para ejecutar sentencias SQL. Permite al usuario emitir sentencias SQL y procesar el resultado"* (IBM, AS/400 Developer Kit para JAVA, 1998, pág. 131).

Una vez contemplados los temas de Sistema de Archivos Integrado, *QSHELL Interpreter,* ejecución de Clases *JAVA* desde *RPG* y conexión a la Base de Datos con *JDBC* voy a explicar el proceso completo del envío del archivo a través de *SFTP*.

Todas las noches se ejecutaba un proceso planificado dentro del *AS/400*. Este proceso planificado ejecutaba un programa RPG. A su vez este programa mandaba a llamar varias Clases *JAVA*.

La primera Clase que se ejecutaba consistía en tomar la información de una tabla mediante *JDBC* y bajarla en un archivo de texto plano. El cual se guardaba dentro de una ruta del Sistema de Archivos. La segunda Clase iniciaba una conexión a través de *SFTP* al servidor *WINDOWS*. Una vez que se abría la conexión, la Clase tomaba el archivo de texto y lo depositaba dentro del Servidor *WINDOWS*. Mientras esto tanto dentro del programa *RPG* se tenía el control total del proceso, este grababa el *LOG* dentro de tablas de la Base de Datos.

Una vez que se recibía la confirmación de la entrega del archivo de texto, se ejecutaba la última Clase la cual mandaba correos para informar el estatus del envío, así como el número de registros enviados dentro del archivo de texto.

2.7 El *Back*

Dentro de la empresa donde trabajo, una de las divisiones que se hace en Sistemas se denomina "*El Front*" y "*El Back*". "*El Front*" corresponde a las aplicaciones que se presentan en las agencias, que es todo lo visual para que los vendedores de comercio y los socios de servicios financieros puedan trabajar y optimizar las ventas. Todos los datos que se arrojan de "*El Front*" son manejados en "*El Back*", lo cual refiere al uso de las transacciones enviadas de tienda para la el almacenamiento o generación de información.

La gerencia a la que pertenecí tomaba los datos de las agencias para la creación de la información contable. Dentro de la gerencia donde trabaje, dividíamos los procesos en dos ramas: "*La línea"* y *"El batch".* "*La Línea"* eran los procesos que se encargaban de generar información al momento de llegar las transacciones a central. *"El BATCH"* correspondía a los procesos que se ejecutaban día a día en las madrugadas para generar otro tipo de información contable una vez procesadas todas las transacciones de las agencias a lo largo del día.

2.7.1. Procesos Transaccionales

"La Línea" correspondían a los procesos que se ejecutaban una vez que llegaban las transacciones de las agencias, por lo tanto convivían con más procesos que no tenían nada que ver con los procesos contables, es decir, por cada transacción se ejecutaban un gran número de programas; para alimentar tablas con los abonos realizados por los clientes, el alta de un nuevo cliente, el cambio de domicilio de un cliente, la compra de un cliente con una tarjeta crédito, por mencionar algunas operaciones.

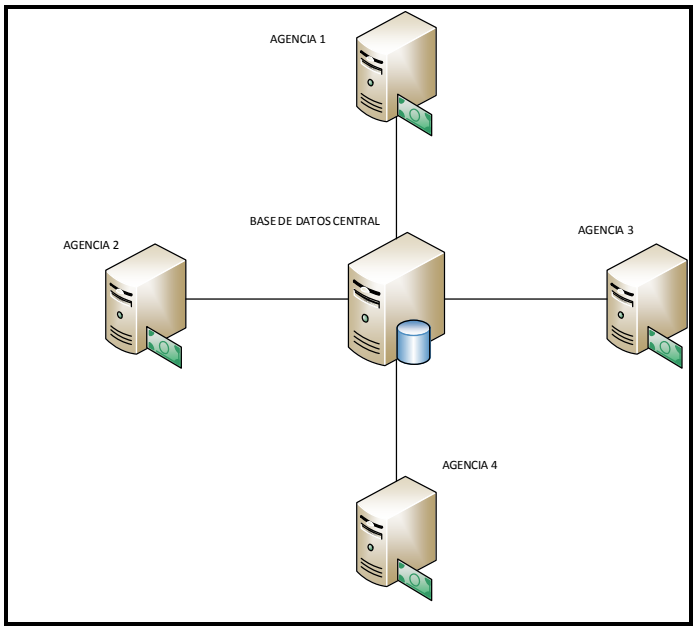

Figura 2.17. Elaboración propia (2014). Diagrama de la recepción de información en la Base de Datos Central.

La lógica del procesamiento de las transacciones era la siguiente, no se podía procesar una transacción mientras no fuera procesada su antecesora. Dependiendo del a tipo de transacción que llegara a Central, los procesos se ejecutaban de la siguiente manera:

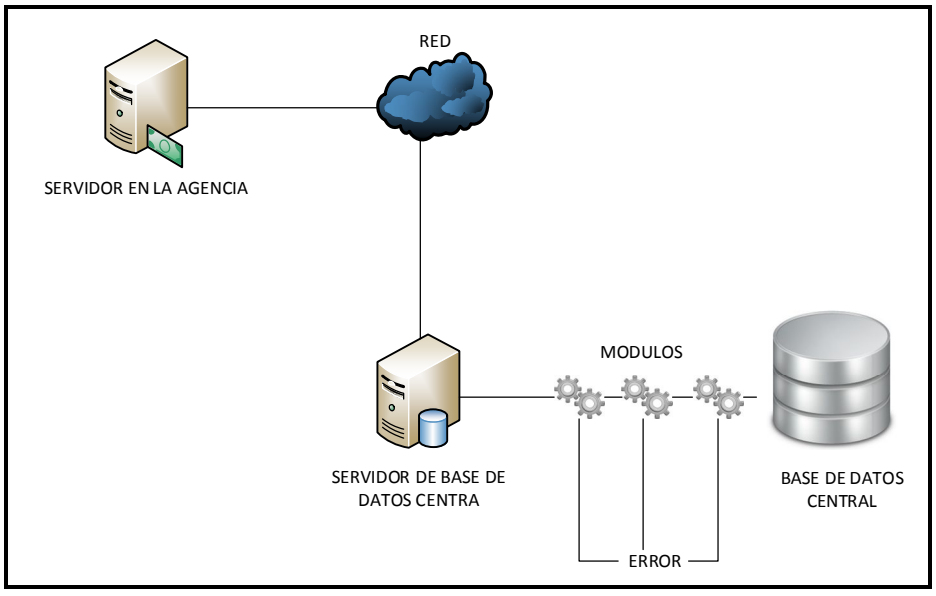

Figura 2.18. Elaboración propia (2015). Diagrama de procesamiento de la información en la Base de Datos Central.

Si dentro de un módulo existía un error, la transacción hacia un *Rollback* y todo lo que se había grabado se deshacía para que nuevamente fuera procesada la agencia desde la transacción en error. Los errores en la transaccionalidad provocaban que la agencia no pudiera impactar información, por lo tanto, los abonos realizados no actualizaban el saldo del cliente, el surtimiento de un producto no afectaba la capacidad de crédito de un cliente. Detener una sucursal significaba perdidas monetarias, además si el error se presentaba dentro de la mayoría de las tiendas esto se traducía en una contingencia.

Una de las contingencias que me toco revisar fue en un cambio relativamente sencillo. Los cambios a productivo se instalaban en la noche, por lo tanto los errores se presentaban en la mañana del siguiente día cuando las agencias iniciaban operaciones. A la mañana siguiente de pasar mi cambio, alrededor de las 10:30 a.m. recibí una llamada de los operadores responsables de la Base de Datos, en donde se me comento que había un error en el proceso al cual había cambiado el código fuente, el error ya se había expandido a 600 sucursales. Yo estaba consiente que el cambio realizado no significaba algo extraordinario. Me pidieron reversar mi cambio, a lo que me negué y solicite se me diera una terminal y un usuario con la capacidad de *debuggear* las transacciones.

Debuggeando el proceso encontré que las transacciones que se estaban mandando tenían los saldos corridos, por lo tanto enviaba cifras extraordinariamente altas, esto desbordaba las variables en los cálculos contables. Revisado esto, se nos informó que día a día se hacía una carga masiva de surtimientos de tarjetas de crédito. El archivo que se procesó ese día se había dañado y los importes no correspondían.

Como se puede observar los cambios realizados tenían que ser minuciosamente revisados para evitar errores o adjudicaciones de errores que no correspondían, por más pequeño que fuese el cambio podía desatar errores con grandes consecuencias.

2.7.2. Procesos *Batch*

Los procesos nocturnos tenían también sus particularidades. Cada noche se tenía una ventana de tiempo de aproximadamente 1.5 horas para la ejecución de los procesos contables. El tiempo es realmente amplio para la capacidad de equipo que se tenía. Esto quiere decir que todos los recursos del servidor eran dedicados a estos procesos. La importancia y el tiempo de ejecución de estos procesos se ven reflejados en la ventana de tiempo.

Los procesos se disparaban en automático, cada transacción tenía un cabecero y un detalle. El cabecero contenía la fecha de la transacción; una vez

que se detectaba el cambio de día en la fecha de las transacciones se detenía el proceso en línea y el proceso *Batch* se ejecutaba; copiando toda la información operativa a tablas temporales para liberar la línea y en paralelo ejecutar cuadres contables. Un proceso era ejecutado por cada agencia que mandaba transacciones.

Aunque la hora con más aglomeración en "*El batch*" eran entre las 00:30 a.m. y la 1:30 a.m. Los procesos podían ejecutarse desde las 11:00 p.m; esto dependía de cada agencia, si el servidor mandaba por adelantado alguna transacción o iniciaban las sucursales de Latinoamérica antes que los operadores *holdearan* las *Colas de Trabajo*. Por lo tanto si existía algún error este se tenía que resolver vía telefónica. A veces las llamadas se convertían en conferencias entre varias gerencias para resolver el problema.

Recuerdo con poco agrado un mantenimiento al equipo productivo, el cual se extendió hasta aproximadamente las 7:00 a.m. La función de los procesos BATCH era generar cuadres contables además de enviaba la información contable a través de archivos de texto plano. Se me informó vía telefónica aproximadamente a las 00:45 a.m. que el servidor productivo no levantaba su servicio después de su mantenimiento; a la 1:30 de la mañana se me informó que la información dirigida a Guatemala no se había enviado. A las 2:00 p.m. se me informó que México y Panamá se habían quedado sin su envío de información. Se revisó con las áreas responsables, tomando como hora límite las 4:00 a.m.; de pasar esta hora la información contable ya no podía procesarse, ya que existía información contable en otras plataformas. A la hora fijada marqué a la empresa esperando una respuesta positiva, sin embargo se me informó que se tenía que procesar la información sin nuestros archivos. Para terminar con la mala noche tenía que presentarme a las 9:00 a.m. para revisar todas las incidencias y como hacer un plan de contingencia para atacarlas.

2.8 El Arrendamiento Financiero

"*Compra todo aquello que aumente su valor y arrienda lo que se devalúa"*
Jean Paul Getty

Uno de los proyectos contables que en los que participé fue la adecuación de un esquema de venta a través del modo de renta, con Arrendamiento Financiero.

El Arrendamiento Financiero es el instrumento a través del cual una empresa (la Arrendadora), se obliga a comprar un bien para conceder el uso de éste a otra persona (Arrendatario o Cliente), durante un plazo forzoso; el arrendatario a su vez se obliga a pagar una renta, que pueden fijar desde un

principio las partes, siempre y cuando ésta sea suficiente para cubrir el valor de adquisición del bien, y en su caso los gastos accesorios aplicables.

"Una de las ventajas del arrendamiento es que permite al arrendatario utilizar el bien, sin que sea de su propiedad, lo cual evita que efectúe erogaciones en la compra de activos, al mismo tiempo que facilita la negociación con el proveedor pues le permite obtener precio de contado" (CONDUCEF, 2014).

Sin embargo, establecer los beneficios del arrendamiento financiero requiere de un análisis de los enfoques legal, financiero y fiscal, sin embargo, podemos precisar algunas de primera instancia, tales como:

- *"El financiamiento del bien puede llegar a ser hasta por el 100% de su valor.*
- *El financiamiento puede ser a largo plazo.*
- *Preserva el capital del arrendatario para otros proyectos de inversión.*
- *Los intereses e inventarios son deducibles para efectos del Impuesto Sobre la Renta (ISR), lo que beneficia a los arrendatarios y los arrendadores, respectivamente.*
- *Quienes obtengan ingresos derivados de estos contratos, pueden optar por considerar como ingreso el total del precio pactado, o bien, la parte del precio exigible.*
- *El arrendatario es quien selecciona el bien, y negocia el precio y las condiciones de la entrega directamente con el proveedor del bien.*
- *El arrendatario es quien figura como propietario del bien para efectos fiscales, lo que le permite hacer la deducción de la inversión*" (Lucina Trejo Ceseña, 2009).

Una de las diferencias contables entre una venta de una mercancía con el esquema de crédito normal y otro por Arrendamiento Financiero es que no existían intereses dentro del Arrendamiento Financiero. Las cuentas de Capital contenían el Capital, los Intereses e IVA. En un crédito tradicional los intereses se Incrementaban día a día, en el Arrendamiento Financiero los ingresos que se fincaban al inicio del crédito se disminuían diariamente.

2.9 Estados Financieros

"El Estado Financiero, conocido también como balance general, es un Estado básico donde se muestra los montos del activo, pasivo y del capital de un negocio a una fecha específica" (Guajardo, 2012, pág. 94).

"El activo es un recurso económico propiedad de una entidad, que se espera rinda beneficios económicos en el futuro. Su valor se determina con base en el costo de adquisición del artículo, más todas las erogaciones de su traslado,

instalación y arranque de operación" (Guajardo, 2012, pág. 94). El activo se puede clasificar de la siguiente manera:

- *"Activo circulante. Son aquellos activos de los cuales se espera obtener beneficios económicos en un periodo normal de operaciones, que se puedan convertir en efectivo en un plazo menor a un año"* (Guajardo, 2012, pág. 95).
- "*Activo no circulante. Se denomina así a los activos de los cuales se espera obtener un beneficio económico en un periodo mayor al de la operación normal*" (Guajardo, 2012, pág. 95)

El pasivo representa lo que debe el negocio, así como los compromisos y obligaciones económicas que tiene con otras personas o entidades conocidas como acreedores. "*Los acreedores tienen derecho prioritario sobre los activos del negocio, antes que los dueños. En caso de la disolución o cierre del negocio, con el producto de la venta de los activos se debe pagar primero a los trabajadores, en segundo lugar entrarían los acreedores y el remanente es para los dueños"* (Guajardo, 2012, pág. 95).

La clasificación del pasivo queda de la siguiente manera:

- *"Pasivo circulante. Lo constituye aquellas cuentas que representan obligaciones que normalmente requerirán el uso de algún activo circulante antes del término del periodo normal de operaciones"* (Guajardo, 2012, pág. 95).
- *"Pasivo a largo plazo. Lo forman cuentas que representan las obligaciones cuyo vencimiento es mayor al periodo normal de operaciones"* (Guajardo, 2012, pág. 95).

Otro elemento del estado financiero es el Capital, el cual representa la parte de los activos que pertenecen a los dueños del negocio; también conocido como activos netos de una entidad, es decir, activos menos pasivos. "*El capital no significa el derecho sobre un monto determinado de efectivo sobre algún activo específico; más bien, se refiere al derecho que se tiene sobre los activos totales de la empresa. En resumen, el activo es la diferencia entre el monto de los activos que posee el negocio y los pasivos que debe"* (Guajardo, 2012, pág. 96).

Existen otros términos comunes de uso frecuente para designar la participación de los dueños, el cual es el Capital Contable y el Capital Neto. El Capital Contable puede aumentar en dos formas:

1. *"Por la aportación en efectivo o de otros activos al negocio"* (Guajardo, 2012, pág. 96).
2. *"Por las utilidades retenidas provenientes de la operación del negocio"* (Guajardo, 2012, pág. 96).

El Capital Contable puede disminuir de las siguientes formas:

1. *"Por el retiro de efectivo u otros activos del negocio por parte de los accionistas. Dicho retiro puede ser un reembolso del Capital aportado o un reparto de las utilidades obtenidas, a las que se llaman dividendos"* (Guajardo, 2012, pág. 96).
2. *"Por perdidas provenientes de la operación del negocio"* (Guajardo, 2012, pág. 96).

El capital de una empresa se clasifica en Capital contribuido y capital ganado:

- *"Capital contribuido. Está formado por las aportaciones de los dueños o propietarios. En este apartado se encuentra el capital social, que incluye el monto de los diferentes tipos de acciones adquiridas por los socios"* (Guajardo, 2012, pág. 96).
- *"Capital ganado. Está conformado por los resultados de las operaciones normales de la empresa (ganancias o pérdidas). Las cuentas que se encuentran en este apartado son las utilidades retenidas y/o las pérdidas acumuladas, que representan los resultados de los ejercicios anteriores"* (Guajardo, 2012, pág. 96).

2.9.1. Transacciones financieras

Una transacción es un evento que afecta económicamente a una entidad y que puede medirse en términos monetarios. "*Las transacciones de negocios se clasifican en grupos de partidas similares llamadas Cuentas y es ahí donde se registran los aumentos o las disminuciones de cada partida provocados por la transacción de negocio. Todo sistema contable tiene una cuenta por separado para cada clase de activo, pasivo, capital, ingreso y gasto"* (Guajardo, 2012, pág. 127).

Cada cuenta, también llamada cuenta de mayor tiene una sección para anotar los aumentos y otra para anotar las disminuciones. "*Mediante el registro contable se pretende clasificar los efectos de las transacciones realizadas por un negocio en los lugares correspondientes, es decir, todas las actividades relacionadas con el efectivo en la cuenta de efectivo, todas las de materiales de oficina en la cuenta de materiales de oficina, todas las cuentas por pagar en cuentas por pagar y así sucesivamente"* (Guajardo, 2012, pág. 128).

"Una cuenta en su forma más básica consta de tres partes: el nombre, un espacio para registrar los aumentos y otra para registrar las disminuciones, a lo cual se le llama movimientos. Mientras tanto las columnas donde se identifican los aumentos y las disminuciones se denomina debe y haber o cargo y abono. Según

la naturaleza de la cuenta, él lado que se utilice para registrar los aumentos y las disminuciones será diferente" (Guajardo, 2012, pág. 128).

Para poder aumentar o disminuir las cuentas se ha establecido un par de reglas sencillas, las cuales son básicas para el registro de las operaciones. Éstas se reducen a saber qué movimiento contable se debe realizar para aumentar o disminuir cada una de las cuentas básicas. Estas reglas son conocidas como reblas del cargo y abono:

- *Cargo. Movimiento del lado izquierdo de la cuenta. Representa un aumento en las cuentas de activo y gasto, o una disminución en las cuentas del pasivo, capital e ingreso* (Guajardo, 2012, pág. 129).
- *"Abono. Movimiento del lado derecho de la cuenta. Representa una disminución en las cuentas de activos y gasto, o un aumento en las cuentas de pasivo, capital e ingreso"* (Guajardo, 2012, pág. 129)

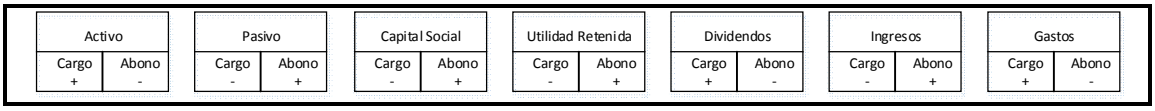

Figura 2.19. (Guajardo, 2012, pág. 129) *Ecuación contable básica con las reglas de cargo y abono.*

El saldo de una cuenta es la diferencia entre las columnas del debe y el haber, para obtenerla se suman algebraicamente ambas columnas, es decir, se suman por separado todos los cargos y todos los abonos. "*Una vez que se tiene la suma, se resta la suma total de los cargos a la suma total de los abonos. La columna con el importe más alto determina si el saldo es deudor (debe) o un saldo acreedor (haber*)" (Guajardo, 2012, pág. 129).

2.10 Los Circuitos Contables

Los Circuitos Contables corresponde a un documento donde se explica las cuentas contables que pretende un contador afectar cuando se genera una operación. Cada Cuenta Contable debe de tener un importe y puede ser generada dependiendo de ciertas circunstancias, es decir, en un evento de Abono a Saldo se afectan el Capital; pero dependiendo de la morosidad del cliente se afecta una u otra cuenta de Capital Vigente o Capital Vencido. Colocare un ejemplo muy sencillo de un Circuito Contable:

La compra de una mercancía contablemente se traduciría de la siguiente manera:

En la entrada de mercancía:

Proveedor	
	Importe del Producto

Figura 2.20. Elaboración propia (2014). Cuenta de proveedor al abono

Una cuenta de proveedor al debe corresponde a una obligación que se tiene con este proveedor.

Gasto	
Importe del producto/ (1 + (tasa de IVA/100))	

Figura 2.21. Elaboración propia (2014). Cuenta de gasto al cargo

Al cargar una cuenta de Gasto se entiende que hice una afectación a mis cuentas de resultados.

IVA	
(Importe del producto/ (1 + (tasa de IVA/100)))* tasa de IVA/100	

Figura 2.22. Elaboración propia (2014). Cuenta de IVA al cargo

La cuenta de IVA cargado corresponde a un IVA pagado, el cual posteriormente se puede hacer acreditable. Si se presenta al abono corresponde a un IVA Cobrado

Aceptación de la deuda:

Existen ocasiones en donde el Contador quiere plasmar un movimiento que representa una acción operativa. La cual no tiene realmente una afectación contable, simplemente se tiene que ver reflejado el movimiento.

Proveedor	
Importe del Producto	

Figura 2.23. Elaboración propia (2014). Cuenta de proveedor al cargo

Proveedor	
	Importe del Producto

Figura 2.24. Elaboración propia (2014). Cuenta de proveedor al abono

Al cargar y abonar la misma cuenta contable, la operación aritmética es 0 (cero), sin embargo en la contabilidad la operación tiene otro significado. El cargo al Proveedor se entiende por la pérdida de una obligación con este, mientras el abono se entiende por una nueva deuda con el Proveedor.

Pago de la deuda:

Banco	
	Importe del Producto

Figura 2.25. Elaboración propia (2014). Cuenta de banco al abono

Cuando se realiza un abono a una cuenta de Banco o de Caja, se hace una salida de dinero de las arcas.

Proveedor	
Importe del Producto	

Figura 2.26. Elaboración propia (2014). Cuenta de proveedor al cargo

Como dije anteriormente, el cargo a una cuenta de Proveedor se realiza una pérdida de obligación con él.

2.11 Unificación de Procesos

El Sistema Contable en el que participaba, generaba la información contable de los movimientos generados por compras a crédito; la información contable de movimientos al contado lo generaba otra área. Aproximadamente en 2011 surgieron cambios en la Dirección a la que pertenecía, lo cual llevo como consecuencia que a nuestra área se anexara el área de pagos a Socios Comerciales. El área estaba constituida por una sola persona, un desarrollador muy ordenado y con varios años recorridos dentro del RPG de quien aprendí muchas cosas.

Me gustaría comentar a que se dedicaba este desarrollador antes explicar lo realizado en este proyecto. Existen módulos bancarios de la empresa dentro de otros negocios otorgando crédito a sus clientes en la mercancía comprada dentro del establecimiento. Estos otros negocios se le denominan Socios Comerciales. Además existían Socios Comerciales muy cercanos a la empresa, como lo eran las ventas de Seguros.

Si existía un crédito otorgado a un cliente de un Socio Comercial esté se liquidaba al momento al Socio Comercial, mientras la parte bancaria de la empresa se quedaba con la deuda del cliente. Los intereses del crédito era la ganancia de la parte bancaria de la empresa. Si un producto se vendía con un seguro, este se tenía que liquidar a la empresa aseguradora.

Como el Área Contable de Crédito estaba separada del Área Contable del pago a los Socios Comerciales existían diferencias en lo contabilizado, primeramente en tiempo de generación de la información; mientras el área contable generaba la información a través del flujo transaccional; el área de pagos generaba su información en semi-linea, esto es, esperaba la acumulación de un cierto número de registros para procesarlos. Otro problema era la diferencia en cálculos, la salida de dinero por la liquidación de los créditos y la entrada o salida por las comisiones cobradas o pagadas debía ser por la misma cantidad reflejada en la contabilidad entre ambos procesos.

Por ejemplo, en el cobro de una comisión por la venta de un seguro se tenía generaban los siguientes apuntes contables:

Información Contable del Crédito:

Comisión		Ingreso		IVA	
x			x/(1+IVA)		(x/(1+IVA)) * IVA

Figura 2.27. Elaboración propia (2014). Apunte contable de una Comisión

Información de Contable del Socio Comercial:

Comisión		Banco	
x			x

Figura 2.28. Elaboración propia (2014). Apunte contable del cobro de una comisión.

Como se puede observar la cuenta de Comisión aparece en ambos circuitos contables. En la parte del Crédito aparece la cuenta cargada y se lleva contra una cuenta de Ingreso y un IVA cobrado. Esto por ser un Ingreso por esta comisión. Mientras en la contabilidad del pago a los Socios Comerciales se ve la contrapartida de la cuenta de Comisión, la cual está abonada y se lleva contra una cuenta de Banco. La cuenta de Banco está cargada, lo que significa que hay un flujo de dinero hacia dentro de la cuenta.

La solución que se dio a este problema fue la homologación de la generación de la información del pago a los Socios Comerciales y la información contable del Crédito. Como la información contable del Crédito se generaba a través del flujo transaccional se decidió que la información se generara desde ese

flujo, además que la generación de la información contable del Crédito era bastante más grande que la de los Socios Comerciales. Entonces se me encomendó la tarea de adaptar los procesos para que generar la información de los Socios Comerciales, la cual tenía ciertas particularidades.

La primera particularidad era la siguiente. La corrección de un apunte contable del crédito se podía dejar de la misma manera, simplemente se colocaba un indicador de anulación para cambiar el apunte al cargo o al abono. Es decir, un abono con indicador de anulación se convertía en cargo y viceversa. Sin embargo dentro de la información contable del pago a Socios Comerciales se tenían que generar los reversos sin indicador de anulación, por lo tanto un cargo se anulaba con un abono y viceversa. Además de esto se tenía que tener un control cada movimiento en el banco, ya que este aparecería en el estado de cuenta del Socio Comercial, para saber cuántas veces se le hacía un deposito o retiro bancario y por qué concepto. Este concepto se guardaba en una referencia única que se colocaba en la cadena de cobro o pago.

Otra particularidad era que la información contable se podía cuadrar hasta el fin de mes, para cerrar correctamente el periodo, sin embargo si existía un error en el flujo de efectivo a un Socio Comercial este se tenía que arreglar a la brevedad, ya que eran movimientos que afectaba las cuentas bancarias del Socio Comercial.

El programa para generar la información se adaptó para estas particularidades. En especial se tomó en cuenta los reversos para no afectar de manera excesiva las cuentas bancarias de los Socios Comerciales y que no fuera un tema que los preocupara.

Fuimos liberando poco a poco los productos para no perder el control y de esta manera revisar lo afectado por el cambio. Cuando fui promovido al Área Fiscal como Líder de Proyecto ya teníamos alrededor del 70% de los productos liberados en la homologación de interfaces. Una de las razones que hacía lenta la liberación de productos era la investigación de como parar la generación de la información, ya que todo se controlaba dentro de programas, mientras que en el Sistema Contable de Crédito todo se manejaba por parámetros en catálogos.

2.12 INFORMATICA POWER CENTER

Una de las herramientas que tuve la oportunidad de aprenden en mi etapa de desarrollador y aplicarla brevemente fue INFORMATICA POWER CENTER. INFORMATICA es una herramienta capaz de extraer información de diferentes fuentes, trabajar con ella y cargarla a otra base de datos. Para esto INFORMATICA tiene tres herramientas:

2.12.1. *Power Center Designer*

En esta herramienta se diseñan los Mapas; dentro de los Mapas se encuentran las tablas o archivos origen, es decir, de donde se va a extraer la información. Además se definen las tablas o archivos destino; a donde se requiere cargar la información. Las fuentes y los destinos pueden variar de tecnología, es decir, se pueden enviar y recibir entre diferentes manejadores de Bases de Datos, además de leer o escribir un archivo de texto plano dentro de un *FTP*. Las conexiones entre las bases de datos se hacen a través de *ODBC*[18].

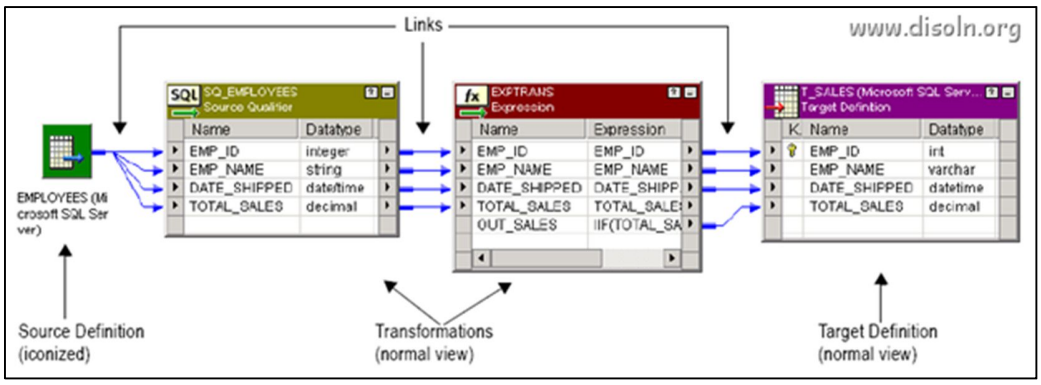

Figura 2.29- Data Intelligence Solution(2012). Visualización grafica de los Mapas de INFORMATICA POWER CENTER. Recuperado del sitio: http://www.disoln.org/2012/08/Understand-Informatica-PowerCenter-Mapping-Designer.html

En la figura 2.29 se muestra la definición la tabla origen *EMPLOYEES*, la cuenta contiene 12 campos, mientras que la tabla destino *T_EMPLOYYES* contiene solo 4 campos. La relación de los campos se hace a través de enlaces llamados *Link*. Como se puede observar la creación de los mapas de *INFORMATICA* es muy gráfica, lo que facilita al programador el desarrollo y mejora los tiempos.

Dentro de los mapas también se definen las transformaciones que requiere la información, como por ejemplo filtros, concatenaciones, cambios de tipo de dato, entre otros. INFORMATICA es capaz de ejecutar procedimientos almacenados antes o después de la carga de información.

2.12.2. *Power Center WorkFlow Manager*

[18] ODBC. Open Data Base Conectivity. Conectividad abierta de bases de datos.

En esta herramienta se crean las tareas ligadas a los mapas, es decir, un flujo completo puede constar de varios mapas. Cada tarea tiene un orden de ejecución. Dentro de los *WorkFlow* se define el orden para ejecutar estas tareas:

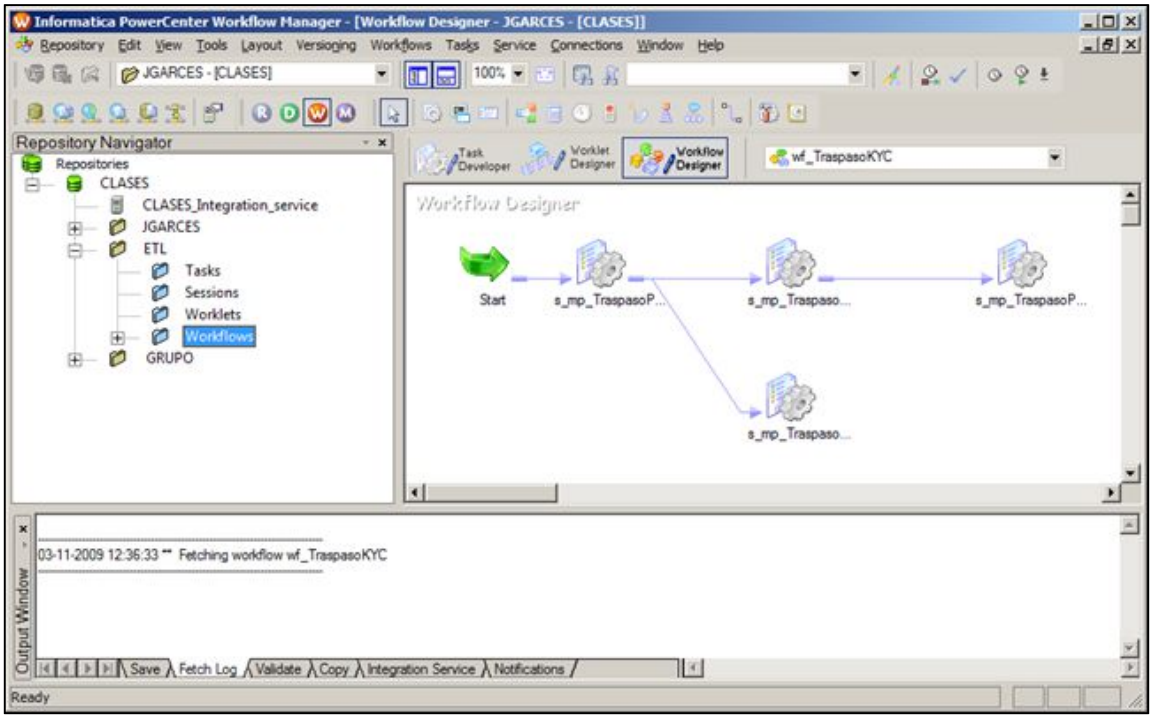

Figura 2.30. Juan Garces (2013). Flujo del trabajo en INFORMATICA POWER CENTER. Recuperado del sitio: http://www.jgarces.info/introduccion-a-informatica-powercenter/6/

2.12.3. *Power Center Monitor*

Una vez creados los mapas, estos se pueden ejecutar por demanda o a través de tareas planificadas. La revisión de los procesos en ambos casos se hace a través del *Power Center Monitor*:

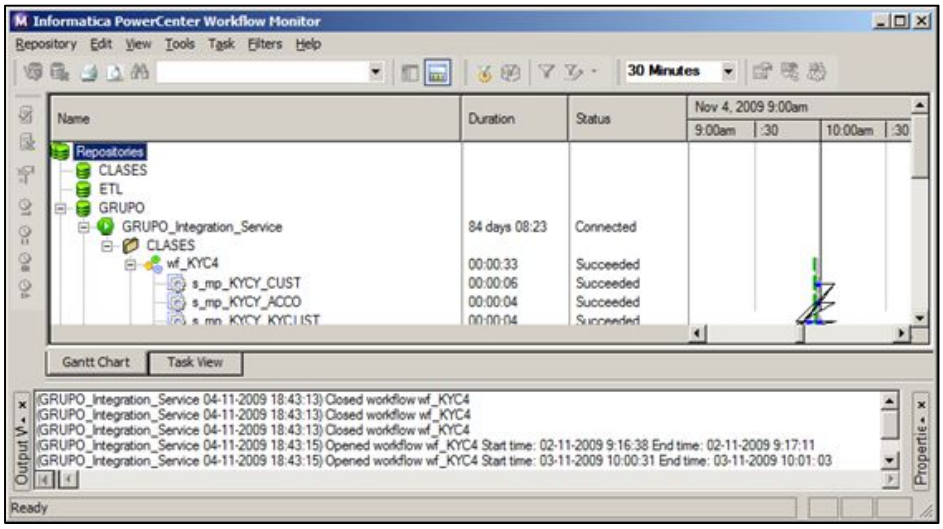

Figura 30. Juan Garces (2013). Monitoreo de procesos en INFORMATICA POWER CENTER. Recuperado del sitio: http://www.jgarces.info/introduccion-a-informatica-powercenter/9/.

Los desarrollos realizados fueron relativamente pocos, sin embargo el estudio de la herramienta me abrió las puertas para conocer otras tecnologías como ORACLE y DB2. Hoy en día dentro de la empresa es la herramienta oficial para la extracción de información entre servidores de distinta tecnología.

2.13 Base de Datos ORACLE

2.13.1. Breve historia de ORACLE

"Una Base de Datos ORACLE es una colección de datos con uno o más archivos. La Base de Datos contiene estructuras físicas y lógicas" (Loney, 2009, pág. 5)*V*. La actual versión de la Base de Datos *ORACLE* es el resultado de 30 años de innovador desarrollo. En 1977 Larry Ellison, Bob Miner y Ed Oates iniciaron la consultoría *Software Development Laboratories*, la cual llego a ser *Relational Software, Inc. (RSI)*. En 1983, RSI se convirtió en *Oracle Systems Corporation* y posteriormente en *Oracle Corporation*.

"En 1979 RSI introduce ORACLE V2 como el primer Sistema de Gestión de Base de Datos Relacional comercialmente disponible, un evento marcado en la historia de las Bases de Datos relacionales. En 1983 ORACLE introduce la V3. Fue la primera Base de Datos ejecutable en mainframes, minicomputadores y computadoras personales. La Base de Datos fue escrita en C, permitiendo a la Base de Datos fuera ejecutada en múltiples plataformas.

La versión 4 introdujo la versión múltiple de vista consistente de datos vistos por un usuario. La versión 5, apareció en 1985, soportando diseño de

Cliente Servidor. Con la versión 6 se publicó la primera versión del lenguaje PL-SQL.

La versión 7 publicada en 1992 introduce los Procedimientos Almacenados y Disparadores. En 1997 aparece la versión 8, la cual soportaba nuevos tipos de datos, además de soportar el particionamiento de grandes tablas. La versión 8i, apareció en 1999, equipado con soporte nativo para protocolos de internet y JAVA.

La versión 9i en 2001 contiene la habilidad de almacenar y consultar archivos XML. La versión 10g, capaz de coordinar gran número de servidores y almacenamiento para actual como una sola gran computadora" (ORACLE, Concepts 11g Release 2 (11.2), 2014, pág. 4).

2.13.2. Curso de Oracle a la medida

A finales de 2011, me dieron la oportunidad de tomar un curso de *PL-SQL*. La empresa a la que pertenezco pretendía la transición del equipo *AS/400* a aplicaciones ejecutándose en Bases de Datos *ORACLE*. Los sistemas *AS/400* están dedicados al manejo de grandes volúmenes de información, el manejo de una gran transaccionalidad. No tengo los datos correctos, sin embargo se llegó a comentar en el trabajo que el servidor *AS/400* utilizado dentro de la empresa es el de mayor capacidad en Lationamerica y supera varias zonas de la Unión Americana. Intentar cambiar un *AS/400* por servidores con *ORACLE* en mi opinión no es viable. Esto por el tamaño de servidor y la relación de años entre la empresa y esta tecnología.

Con forme fue creciendo la empresa, el servidor *AS/400* fue actualizándose, dentro de mi estancia en la empresa se migraron 3 servidores. Para traer una nueva tecnología e intentar tomar toda la operación que se ejecuta día a día, en mi opinión, tendrían que trabajar varios años para migrar los aplicativos y estabilizarlos. El costo de esta migración y la apuesta para que fuera correcta la nueva tecnología sería muy costosa y al final el ahorro pretendido se convertiría en gasto.

Accedí a tomar el curso de Oracle para conocer otra tecnología y no perder la oportunidad de colocar nuevos conocimientos dentro de mi Curriculum Vitae. El curso fue impartido por Intersoftware, en sus instalaciones en el Centro de Comercio Internacional de la Ciudad de México (WTC). Con una duración de 40 horas el contenido del curso fue el siguiente:

1.- *The Basic Parts of Speech in SQL.*

Este tema se refería a los comandos básicos en SQL. Las palabras claves de estos comandos son simples verbos en inglés, como son *SELECT, INSERT,*

UPDATE y DELETE. Existen otras instrucciones, como son *MERGE* que mejora a las instrucciones *INSERT y UPDATE* con solo un comando.

2.- *Introduction to TOAD for ORACLE*.

TOAD (*Tool for ORACLE Application Developers*) es un herramienta de desarrollo de la empresa QUEST *Software*, con la capacidad de tener varias conexiones a la Base de Datos; además de contar con un explorador de objetos y ventanas de desarrollo y varias características para los administradores de la Base de Datos.

3. - *Getting Text Information and Changing It.*

Dentro de este tema se revisó la manipulación de las cadenas de caracteres. Se vieron funciones tales como *CONCAT, INICAP, LENGTH, SUBSTR* entro otras.

4.- *Searching for Regular Expressions.*

Una expresión regular es una cadena de caracteres que definen un patrón de búsqueda. En una expresión regular encontramos literales y metacaracteres. Los literales se leen al pie de la letra. Los metacaracteres son caracteres que tienen un significado especial.

En una base de datos ORACLE existen diversos escenarios en que la implementación de expresiones regulares constituye una herramienta de gran utilidad:

- *"Búsqueda de texto. Las expresiones regulares nos permiten hacer búsquedas complejas de cadenas de caracteres dentro de las columnas de nuestras tablas.*
- *Formateo de datos. Podemos modificar los datos proyectados en las sentencias SELECT, invirtiendo palabras, agregando o quitando caracteres, etc.*
- *Definición de constraints. A fin de forzar el contenido de una columna para que se ajuste a un formato determinado: casilla de correo, número telefónico, etc.*
- *Manipulación de datos. Por ejemplo, regularizando datos en procesos de migración desde aplicaciones legacy y aplicando reglas de búsqueda y reemplazo (Search & Replace)"* (ORACLE, 2014).

5.- *Conversion and Transformation Functions.*

Las funciones de conversión y transformación como lo son *TO_CHAR, TO_NUMBER* y *TO_DATE* sirven para convertir un tipo de dato en otro.

6.- *Grouping Things Together*.

Existe la palabra reservada *GROUP BY*, la cual es utilizada dentro de la sentencia *SELECT*. La sentencia *GROUP BY* sirve para agrupar elementos del

mismo tipo. De esta sentencia se desprende la palabra reservada *HAVING*, que es utilizada como la sentencia *WHERE* para condicionar el resultado de la sentencia.

7.- *When One Query Depends upon Another Some Complex possibilities*.

Se pueden generar consultas que dependen de otras consultas, esto es, cabe la posibilidad que la cláusula *WHERE* depende del resultado de una consulta, ya sea con uno o múltiples resultados. Además es posible que en sentencias *CASE* dentro de la cláusula *SELECT* dependan también del resultado de consultas. Esto tiene de ventaja hacer las consultas más dinámicas.

Por otro lado existen las uniones de una o varias tablas que tienen relación entre sí. Esto se logra con las distintas versiones de la instrucción *join*. Se puede encontrar junto con la palabra reservada *outer*, *inner*, *natural,* entre otros. La instrucción *join* en *ORACLE* tiene la naturaleza unir tablas que contengan la misma información, sin embargo carece de la instrucción *exception* que contiene *DB2*, la cual regresa los datos que se encuentra en una tabla y no en la tabla que se busca una unión. *ORACLE* lo sustituye con la instrucción *not exist,* la cual hace la misma tarea.

9.- *Changing Data: insert, update, merge, and delete.*

La operaciones de para la manipulación de datos permitidas en PL-SQL son *SELECT* que devuelve las filas de la Base de Datos que cumplen con la condición de la cláusula *WHERE. INSERT* añade filas a una tabla de la Base de Datos. *UPDATE* modifica las filas de una tabla que cumplan con las condiciones de la cláusula *WHERE*. *DELETE* borra las filas identificadas en la cláusula *WHERE*.

10.- *DECODE and CASE: if, then, and else in SQL.*

PL-SQL tiene diversas estructuras de control que permiten modificar el flujo del programa. La sintaxis para esta expresión es la siguiente:

IF expresión booleana **THEN**
Secuencia de instrucciones;
ELSEIF expresión booleana 2 **THEN**
Secuencia de instrucciones;
ELSE
Secuencia de instrucciones;
ENDIF;

La sentencia DECODE es similar a una serie de instrucciones IF-THEN-ELSE anidadas. DECODE tiene la siguiente expresión:

DECODE (Expresión_Base, Compara1, Valor1
Compara 2, Valor2 … Valor predeterminado)

Si Expresión_Base coincide con el elemento de comparación numero n, se devuelve el valor número n. Si ninguna expresión coincide se devuelve el valor predeterminado.

Otra expresión similar al bloque IF-THEN-ELSE es la sentencia CASE, donde su sintaxis es la siguiente:

CASE Expresión_Base
WHEN Valor1 **THEN** sentencia de instrucciones 1;
WHEN Valor2 **THEN** sentencia de instrucciones 2;
…
WHEN Valor n **THEN** sentencia de instrucciones n;
ELSE otra sentencia;
END CASE;

11.- *Creating and Managing Tables, Views, Indexes and Clusters.*

La creación y manipulación de Tablas es de los temas más importantes dentro de las Bases de Datos, ya que el diseño de la Base de Datos depende de la creación de tablas. Un buen diseño de la Base de Dato minimiza problemas posteriores en la relación entre Tablas. Por otro lado, el diseño de los Índices es extremadamente delicado. Las tablas se deben diseñar para que contengan su Índice principal, que se la Llave Primaria; sin embargo la creación de Índices se refiere a la vía de acceso alternativo que se tomará hacia los datos.

La creación de tablas se da con las palabra clave *CREATE TABLE* y su manipulación mediante las palabras reservadas *ALTER TABLE*, la cual nos ayuda a agregar o quitar columnas, *constrains,* cambiar el tipo de una dato; *DROP TABLE* nos ayuda a eliminar las tablas.

Una vista es una consulta, que refleja el contenido de una o más tablas, desde la que se puede acceder a los datos como si fuera una tabla.

Dos de las principales razones por las que podemos crear vistas.

- Seguridad, nos pueden interesar que los usuarios tengan acceso a una parte de la información que hay en una tabla, pero no a toda la tabla.
- Comodidad, para visualizar datos de la Base de Datos puede llevar a tener que escribir complejas sentencias SQL, tener una vista nos simplifica esta tarea, ya que puede ejecutarse con una solo sentencia.

Un Índice tiene varias repercusiones en una Tabla y en el *storage*, es decir, un Índice no solamente crea una acceso a la esta, también genera más espacio en disco: mientras más Índices tenga una tabla mayor es su tiempo de respuesta al insertar registros, ya que por cada registro se tiene que indexar por

cada Índice. Un Índice mal diseñado hace el acceso más lento a la Tabla, el mejor criterio que se maneja para la creación de un Índice es que los campos pertenecientes a este sea lo más selectivo posible.

"Una tabla Cluster es un grupo de tablas que comparten columnas comunes y almacenan datos relacionados en bloques similares. Cuando una tabla es agrupada un solo block de datos puede contener filas de múltiples tablas. La clave agrupada es la columna o columnas que las tablas agrupadas tienen en común" (ORACLE, Concepts 11g Release 2 (11.2), 2014, pág. 23).

12.- *Using SQL*Loader to Load Data.*

La herramienta *SQL Loader* es utilizada para la carga de datos, generalmente de un gran número de registros. Aunque las herramientas de desarrollo como *TOAD* contiene utilidades para la importación de datos, con gran volúmenes de registros es muy lento. *SQL Loader* permite la creación de archivos *.LOG* donde se visualizan los detalles de la carga; además genera archivos .BAD donde se guardan todos aquellos registros que no se pudieron cargar.

Para la utilización de *SQL Loader* se debe configurar el archivo *.CTL*, el cual contiene todas las condiciones para la importación. Mientras el archivo contenedor de los datos tiene la extensión *.DAT*.

13.- *An Introduction to PL/SQL.*

ORACLE es una Base de Datos relacional. El lenguaje utilizado para acceder a las Bases de Datos es el Lenguaje estructurado de Consulta (*SQL)*. *SQL* es un lenguaje flexible y eficiente para la manipulación y la recuperación de los datos. Por otro lado *SQL* es un lenguaje de cuarta generación, lo que quiere decir que el lenguaje describe lo que debe hacerse, pero no la manera de llevarlo a cabo. Los lenguajes de tercera generación como lo son *C* o *COBOL*, son de naturaleza más procedimental. Un programa escrito en un lenguaje de tercera generación interpreta un algoritmo paso a paso para resolver el problema.

Los lenguajes orientados a objetos como lo son *C++* o *JAVA*, también son de tercera generación. Aunque incorporan los principios del diseño orientado a objetos, donde los algoritmos siguen especificándose paso a paso. Cada lenguaje tiene sus ventajas y desventajas. Los lenguajes de cuarta generación como *SQL* son, por regla general bastantes simples (comparados con los lenguajes de tercera generación) y tiene menos instrucciones.

PL-SQL, que significa *Procedural Languaje/SQL* (Lenguaje Procedimental/SQL). Como su propio nombre lo indica, PL-SQL amplia la funcionalidad de SQL añadiendo estructuras de las que pueden encontrarse en otros lenguajes tales como:

- Variables y tipos.
- Estructuras de control, como bucles e Instrucciones IF-THEN-ELSE.

- Procedimientos y funciones.
- Tipos de objetos y métodos.

"*Las construcciones procedimentales están perfectamente integradas con ORACLE SQL, lo que da como resultado un lenguaje potente y estructurado*" (Scott, 2002, pág. 4).

14.- *Procedures, Functions, and Packages.*

Los Procedimientos y Funciones son un tipo especial de bloque *PL/SQL* que se almacena en la Base de Datos en formato compilado y posteriormente son llamados desde un bloque. Procedimientos y Funciones tienen la capacidad de recibir parámetros de entrada y salida, sin embargo una Función siembre debe regresar un valor. Además que una función puede ser utilizada dentro de sentencias *SELECT*.

Los Paquetes son agrupadores, en ellos se puede contener Procedimiento y Funciones, variables y Tipos de datos. Un paquete consta de dos partes, la especificación y el cuerpo; los Paquetes permiten que objetos relacionados entre sí se almacenen juntos en la Base de Datos.

15. – *Triggers.*

Los *Triggers* o Disparadores son parecidos a los procedimientos y funciones, en el sentido de que son bloques nominados de *PL/SQL* con secciones declarativa, ejecutable y tratamiento de errores. Al contrario de un procedimiento, el cual es llamado explícitamente desde un bloque en otro procedimiento; un Disparador se ejecuta de forma implícita cuando ocurre un evento que lo activa y no acepta argumentos. El evento que lo activa puede ser una operación de *INSERT, UPDATE o DELETE* sobre una Tabla de la Base de Datos o algunas Vistas. Otro evento que activa un Disparador puede ser el inicio de la conexión o la desconexión a la Base de Datos.

Algunas finalidades de los Triggers son:

- *"Mantener restricciones de integridad complejas que no pueden hacerse mediante a través de restricciones declarativas especificadas durante la creación de una tabla.*
- *Auditar la información contenida en una tabla, registrando las modificaciones y el autor de las mismas.*
- *Indicar automáticamente a otros programas que es necesario realizar alguna acción, cuando se modifica la tabla".* (Scott, 2002, pág. 407).

16.- *Using Native Dynamic SQL and DBMS_SQL.*

PL/SQL está diseñado para ejecutar instrucciones *SQL*, sin embargo una de sus limitaciones es que solo puede ejecutar instrucciones de manipulación de datos. Para mitigar esta limitación se desarrolló *SQL* Dinámico. Existen dos técnicas para ejecutar *SQL* Dinámico. La primera es a través del paquete *DBMS_SQL*. La segunda técnica es más sencilla es con *SQL* Dinámico Nativo, como su nombre lo indica es parte integrante del propio lenguaje. La instrucción básica utilizada es *EXECUTE IMMEDIATE*.

2.13.3. Desarrollos en PL-SQL

Han sido muy pocos los desarrollos realizados con PL-SQL, no he tenido la oportunidad de mostrar mis conocimientos en un proyecto grande. La mayoría de los desarrollos corresponden a pequeñas adecuaciones en procesos y aplicaciones propias para optimizar procesos ya existentes.

Capitulo 3. Líder de Proyecto Sistemas Fiscal

3.1 Proyecto del Sistema Fiscal

La apertura de un nuevo proyecto del área de Sistemas dedicado a los usuarios del departamento Fiscal llegó en Abril de 2013, era necesaria la creación de un Sistema para automatizar la generación de los reportes fiscales. Mi entonces Gerente, quien era el encargado de la información contable generada por las ventas a créditos fue seleccionado por la Vicepresidencia de Sistemas para llevar el proyecto. Al principio para mi Gerente fue muy confusa la decisión, ya que él llevaba aproximadamente 9 años con el mismo jefe y con un equipo de trabajo consolidado de más de 5 años junto a él.

Por nuestra parte, los subordinados, no teníamos las mismas confusiones; sentíamos que teníamos la capacidad de sacar la gerencia adelante. Además teníamos la esperanza de escalar puestos; ya que las formas de conseguir un aumento salarial dentro de la empresa tiene pocas variantes; se puede dar por un aumento generar por parte de la empresa, cuando algún jefe renuncia o como en este caso, el Gerente era promovido a otra área.

Mi idea original era hablar con mi Director para ser tomado en cuenta en los aumentos salariales. Próximamente cumpliría 5 años dentro de la empresa. Tenía el puesto de Desarrollador de Sistemas y sabía que escalar al puesto de Líder de proyecto era muy difícil; ya que antes de mi había otros dos desarrolladores con más años en la empresa y con gran capacidad en su trabajo. Uno de ellos era mi compañero de proyecto, manejaba el *AS400* mucho antes que yo y es una persona muy ordenada. El otro desarrollador trabajaba en la parte de *ORACLE* y llevaba el proyecto de los Centros de Costo. Años atrás había participado en el desarrollo de un nuevo sistema para el manejo de los Centros de Costo, el cual había tenido una gran aceptación dentro de la empresa.

La segunda idea era hablar con mi Gerente y pedirle me diera una oportunidad en su nuevo equipo de trabajo. La idea no era del todo viable; primero porque se decidió hacer el proyecto en una Base de Datos SQL SERVER y el Front en .NET. Ambas tecnologías son desconocidas para mí. El segundo impedimento era el permiso de mi Director, dentro de la empresa no se puede cambiar de área si no es con el consentimiento de ambos Directores, el Director del área que pierde y el Director que recibe al recurso. El tercer impedimento era el miedo a fracasar dentro del proyecto. Ya que se estimaban 10 meses para terminar el proyecto, eso indicaba jornadas laborales extensas y poco tiempo para la curva de aprendizaje.

No tuve el tiempo de hablar con mi Director y solicitarle el aumento salarial; él ya tenía planes para las plazas desocupadas. Mi compañero de AS/400 fue

promovido a Líder de Proyecto. Lo cual lejos de generarme un sentimiento negativo hacia él, me pareció una buena decisión.

Mi buena suerte llego en Agosto del mismo año, cuando después de varias juntas entre los Directivos del área Fiscal y Directivos del área de Sistemas se decidió cambiar la tecnología para el proyecto; la Base de Datos se alojaría en *ORACLE* y el *Front* se construiría en *JAVA*, además de tener la necesidad de contratar una persona de *AS/400* para los reportes fiscales del área bancaría. Sin saberlo, un día fui llamado a la oficina de mi Director para que me dieran la noticia. En la decisión intervinieron tres personas: mi Director, mi Gerente y mi Líder de proyecto. Se me preguntó si quería hacer el cambio y las consecuencias que este podía tener. Aunque en mis primeros años de programador tomaba decisiones con menos análisis, ahora tenía que tomar una decisión más acertada. En mi puesto de desarrollador conocía a los usuarios y el calendario de las cargas de trabajo, las incidencias que se podían presentar, las tablas de la Base de Datos y un conocimiento aceptable del *AS/400* así como de *SQL*. Por otro lado no podía desperdiciar una oportunidad que quizá nunca se volvería a presentar. El cambio se hizo el 1 de Septiembre de 2013, 6 semanas después de cumplir 5 años en la empresa y 2 días antes que yo cumpliera 30 años de edad.

3.2 Las nuevas Funciones como Líder de Proyecto

Una vez dentro de mi nuevo puesto fue difícil encontrar el área donde debía desarrollarme porque no se integró el área Fiscal bancaría al proyecto inmediatamente, a quienes debía atender con sus reportes; sumado a esto, la plantilla tardo en integrarse y fui requerido para otros procesos. Mi primera tarea fue hacer una presentación del preliminar del proyecto, además de hacer un demo en *JAVA* para que los usuarios pudieran observar a grandes rasgos la idea de las pantallas dentro del nuevo Sistema. Tengo que aclarar que el área Fiscal no contaba con un área de Sistemas dedicada para ella, todos sus reportes se generaban en Excel y pocas veces definían sus procesos a un área de Sistemas para la generación de reportes.

3.2.1. Demo del proyecto Fiscal

Desde que cursaba la Universidad no hacía un desarrollo en *JAVA*, investigando encontré *PRIMEFACES* como *Framework* y el desarrollo lo hice en *NetBeans*. Una de las pantallas que hice fue el arrastre de cuentas contables para

configurar conjuntos de cuentas llamados *Rubros*. Para esto utilicé un elemento *Drag and Drop*, esto en una *DataTable*.

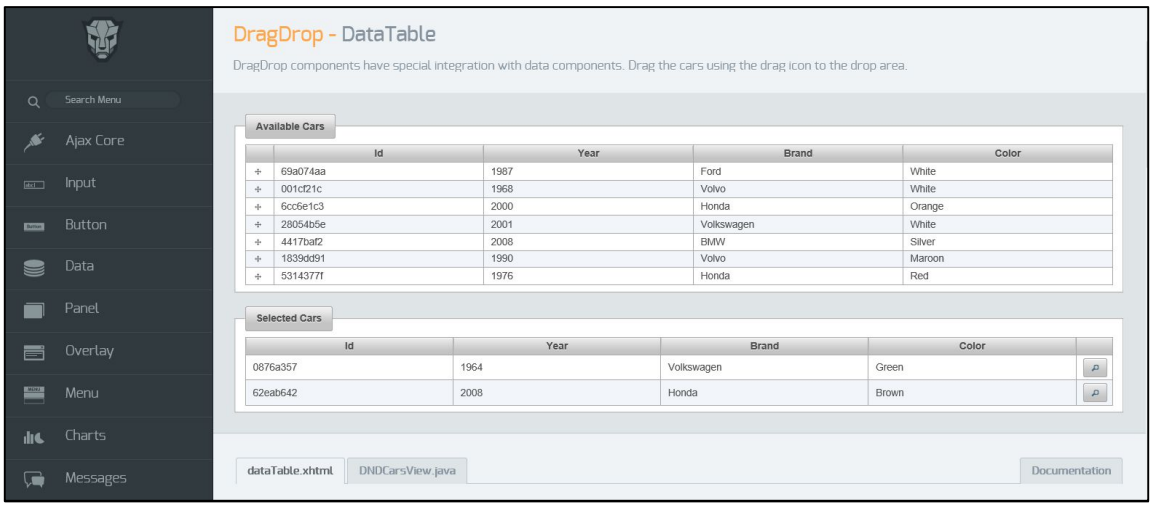

Figura 3.1.PrimeFaces (2014) Drag and Drop en una tabla con PRIMEFACES. Recuperado del sitio: http://www.primefaces.org/showcase/ui/dnd/dataTable.xhtml

Los elementos que se pueden crear con *PRIMEFACES* son muy vistosos, mucho más para mí que venía de crear pantallas en fondo negro y los elementos a base de caracteres.

Otro de los elementos que utilicé fue el *Keyboard.* Este elemento lo utilicé para generar operaciones que se podían presentar dentro de algún reporte. Es decir, con el elemento *DataTable* se creaban los Rubros. Una vez creados estos Rubros podían tener interacción con otro grupo de cuentas; ya sea sumar o restar con otro grupo de cuentas o con una variable creada por el usuario y así poder crear un nuevo rubro y poder mostrarse dentro del reporte. Para esto se tenía que hacer una pantalla que pudiera visualizar estos grupos de cuentas, dar de alta un rubro nuevo y seleccionar la operación que debía realizar entre rubros. No se puede dejar en manos de los usuarios la integridad del Sistema por lo tanto se decidió crear una *Keyboard* limitado con los caracteres estrictamente necesarios, es decir: ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘0’, ‘(‘, ‘)’, ‘*’, ‘/’, ‘+’, ‘-‘.

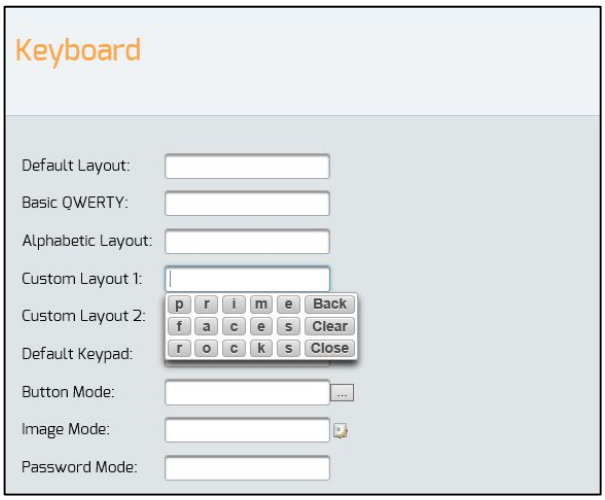

Figura 3.2. PrimeFaces (2014) Keyboard con PRIMEFACES. Recuperado del sitio: http://www.primefaces.org/showcase/ui/input/keyboard.xhtml

La Figura 3.2 corresponde al ejemplo que se muestra en la página de *PRIMEFACES* del elemento *Keyboard*, como se puede observar el elemento se puede personalizar con las teclas que se deciden presentar. Más de estos ejemplos se pueden ver en la página de *PRIMEFACES* en http://www.primefaces.org/showcase/ui/dnd/dataTable.xhtml.

3.3 Explotación de la Información de SAP

Mi segunda actividad fue revisar los reportes que se tenían que presentar del área comercial. La información contable se genera en *SAP*, el cual es un Sistema ERP, Sistemas de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés, E*nterprise Resource Planning*). La información que se genera en SAP es almacenada en una Base de Datos, la Base de Datos puede ser *SAP MAXDB*, *ORACLE*, *SQL SERVER*, *DB2 UDB*. Para comentar esto me baso en la página de *SAP Service MarketPlace* donde se muestran las de las actualizaciones de las Bases de Datos disponibles.

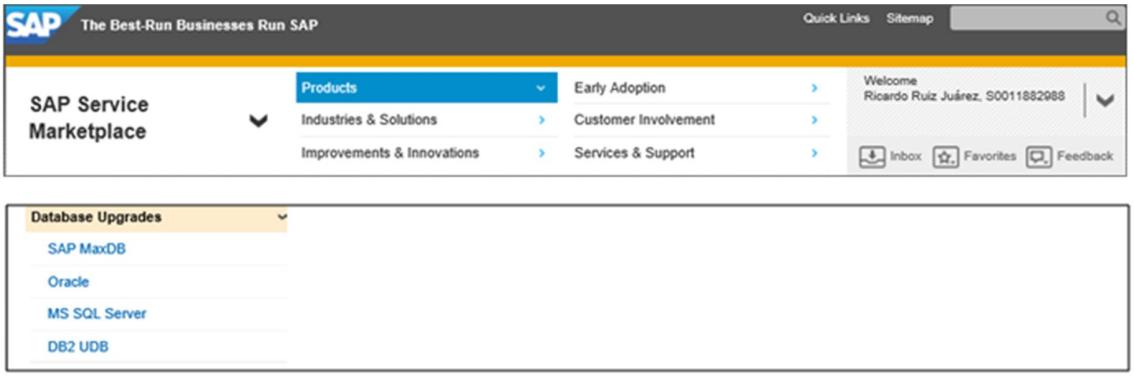

Figura 3.3 SAP Service Marketplace (2015) Actualizaciones de Bases de Datos disponibles para SAP. Recuperado del sitio: https://websmp102.sap-ag.de/

Dentro de la empresa se maneja una Base de Datos *ORACLE* para el Sistema *SAP*. La idea original era bajar la información contable generada en la Base de Datos *ORACLE* y colocarla en una Base de Datos también *ORACLE* que se encuentra en un servidor dedicado al proyecto Fiscal. Se planteó esta estrategia para no consumir los recursos del servidor de *SAP*. El servidor de *SAP* está dedicado al área Contable y no al área Fiscal, mantener nuestras aplicaciones dentro del servidor de *SAP* podía traernos futuras consecuencias en prioridades de ejecución de procesos y un mayor control de nuestros procesos al cargarlos a servidor productivo. Sin embargo esta actividad se complicó ya que al acceder directamente a la Base de Datos se podía perder la garantía de *SAP*.

Se buscaron varias alternativas; el proceso vigente nos hacer depender de otra área para dejarnos en una tabla temporal la información necesaria.

Al no poder bajar la información a la Base de Datos *ORACLE* del sistema Fiscal los reportes se tenían que generar en *SAP* a través de *ABAP* mientras se lograba la conexión directa. Por esta razón se decidió contratar un Líder de Proyecto de *ABAP* y un desarrollador que lo apoyara a programar. Los desarrollos de este esquipo consistirían en procesos de extracción de información. Se requería bajar la información para posteriormente el área de Base de Datos trabajarla y hacer los reportes necesarios.

3.4 Programación en ABAP

Una vez que se contrató el Líder de Proyecto de *ABAP* se iniciaron los desarrollos en el servidor de *SAP* para las descargas de la información. Pero

había un gran detalle dentro del servidor de *SAP*, el detalle radicaba en una homologación entre todas las empresas del grupo. Anteriormente cada empresa trabajaba en un servidor propio de *SAP* o un *Mandante*[19] diferente. Se decidió comprar un sistema con mayor capacidad y homologar las contabilización de las Sociedades[20] Comerciales, Televisivas, Mineras y Arrendadoras. Esta homologación tardó más de un año en terminarse y se planificó para que entrara con el ejercicio 2014. Una de las consecuencias de esta migración fue que muchos procesos quedaran obsoletos. Entre ellos el proceso del cálculo del IVA acreditable.

3.4.1. El IVA acreditable

El artículo 4º de la Ley del IVA está dedicado al IVA Acreditable:

"*Artículo 4o.- El acreditamiento consiste en restar el impuesto acreditable, de la cantidad que resulte de aplicar a los valores señalados en esta Ley la tasa que corresponda según sea el caso.*

Para los efectos del párrafo anterior, se entiende por impuesto acreditable el impuesto al valor agregado que haya sido trasladado al contribuyente y el propio impuesto que él hubiese pagado con motivo de la importación de bienes o servicios, en el mes de que se trate.

El derecho al acreditamiento es personal para los contribuyentes del impuesto al valor agregado y no podrá ser trasmitido por acto entre vivos, excepto tratándose de fusión. En el caso de escisión, el acreditamiento del impuesto pendiente de acreditar a la fecha de la escisión sólo lo podrá efectuar la sociedad escindente. Cuando desaparezca la sociedad escindente, se estará a lo dispuesto en el antepenúltimo párrafo del artículo 14-B del Código Fiscal de la Federación." (SAT, 2015)

Como lo explica el artículo anterior, el Acreditamiento del IVA se da por el impuesto pagado por bienes o servicios, en nuestro caso a los proveedores de las mercancías del área comercial. *SAP* tiene reportes para obtener el IVA que se grabó al momento de generar un documento de entrada de mercancías o un documento de factura. Sin embargo estos reportes no son útiles cuando no se lleva la contabilización como *SAP* la espera. Se debe de llevar un estándar en la contabilización; sin embargo las contabilizaciones dependen de la operación de cada empresa y en mi caso la empresa tiene una operación muy basta y el Sistema se debe adaptar a la Empresa y no la Empresa al Sistema.

19 Es el nivel más alto en la jerarquía del Sistema SAP

20 Una Sociedad representa una entidad independiente de contabilidad/compensación.

Los usuarios del nuevo Sistema Fiscal decidieron abandonar por un momento la creación del Sistema y que el trabajo se enfocara a mantener la operación que se truncó con la homologación del *SAP*, se decidió la creación de un proceso para la obtención del IVA Acreditable. Ya existía un proceso *Z* que obtenía este impuesto, pero se tenía que desarrollar el proceso para obtener el IVA acreditable de las compras que se dieron antes de la migración, ya que al momento de migrar los saldos de las cuentas contables solo se migró el saldo total de las cuentas y el detalle de las contabilizaciones estaba almacenado en el equipo productivo anterior.

Con solo un desarrollador *Senior* el trabajo se iba acumulando. Fue entonces cuando decidí involucrarme en la programación en *ABAP*. Aun no tenía desarrollos de *RPG,* por lo que comente con mi gerente la posibilidad de tomar algún desarrollo y apoyar para liberar el trabajo en menos tiempo.

Fue una gran sorpresa al estudiar *ABAP*. *ABAP* soporta la programación estructurada y la programación orientada a objetos; me di cuenta que las instrucciones de *ABAP* son muy similares a las instrucciones que utiliza *RPG*. Por lo tanto la adaptación tardo mucho menos de lo esperado, aun por mí. Para ese momento ya había desarrollado la lógica de programación para adaptarme a cualquier lenguaje. Como todo lenguaje de programación, es necesario contar con un ambiente de desarrollo. Para poder desarrollar pedí a mis compañeros de ABAP me capacitaran para usar el IDE; es decir, como compilar, *debuggear*, controlar las versiones de los programas. Fue entonces cuando conocí los términos de Activación y Orden de Transporte. La primera se refiere a activar un objeto, se puede grabar un objeto tal como un programa, tabla o estructura pero solo puede ser utilizado o tomarse en cuenta en un cambio si es activado. Por otro lado la versión del cambios se lleva por Ordenes de Transporte, cada cambio a los Objetos dentro del Sistema SAP se relaciona con una Orden de Transporte para poder ser transportado a servidor de Calidad y posteriormente al servidor Productivo, Otra sorpresa fue poder trabajar en un ambiente de Calidad, un gran privilegio del que no contaba en el *AS/400*.

Se me dio la tarea de analizar el proceso de IVA Acreditable, al terminar esta tarea mi reporte fue que el proceso estaba mal, ya que no contemplaba bien el importe del Gasto que generaba el IVA y algunos otros detalles de duplicidad. Se decidió generar un nuevo proceso con las adecuaciones necesarias, además de lograr la conexión con el productivo anterior e implementar las nuevas reglas en el nuevo productivo. Por cada proceso de IVA acreditable se tenían que generar 2, uno para el productivo anterior y otro para el productico actual. El proceso del IVA acreditable se divida en Compras, Factoraje y Fletes.

Una vez que se conoció la noticia de que nos íbamos a dedicar a generar estos procesos se nos solicitó la creación de los mismos procesos para las sociedades dedicadas a las tiendas de autoservicio. Esto implicaba otros desarrollos, ya que para estas sociedades se presentaban otras tasas de IVA y se

agregaban nuevos impuestos como el IEPS (Impuesto Especial sobre Producción y Servicios).

3.4.2. Las tablas de SAP, tablas Cluster y tablas Transparentes

Dentro del desarrollo de procesos en la Base de Datos uno de los temas principales es la optimización de los tiempos. Agradezco a quien fuera mi Líder de Proyecto en mi etapa de desarrollador del *AS/400*, ya que me obligaba a buscar los accesos más rápidos a las tablas y no tener condiciones innecesarias dentro de los programas. El proceso que se tenía del IVA Acreditable manejaba como tabla principal la tabla *Cluster BSEG*, la cual está formada de varias tablas transparentes.

"*Los datos de varias tablas están almacenados juntos en tablas Pool o Cluster. Las tablas asignadas a una tabla Pool o tabla Cluster se denominan tablas agrupadas o Cluster*" (SAP, 2015)

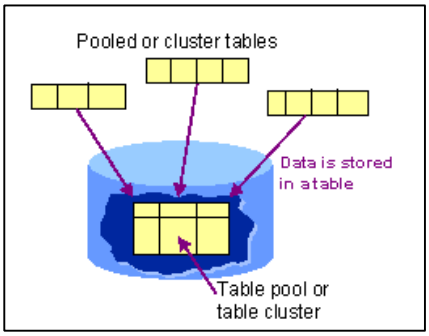

Figura 3.4 Help SAP (2015) Descripción de tablas Pool y Cluster. Recuperado del sito: http://help.sap.com/saphelp_snc70/helpdata/en/81/415d363640933fe10000009b38f839/frameset.htm

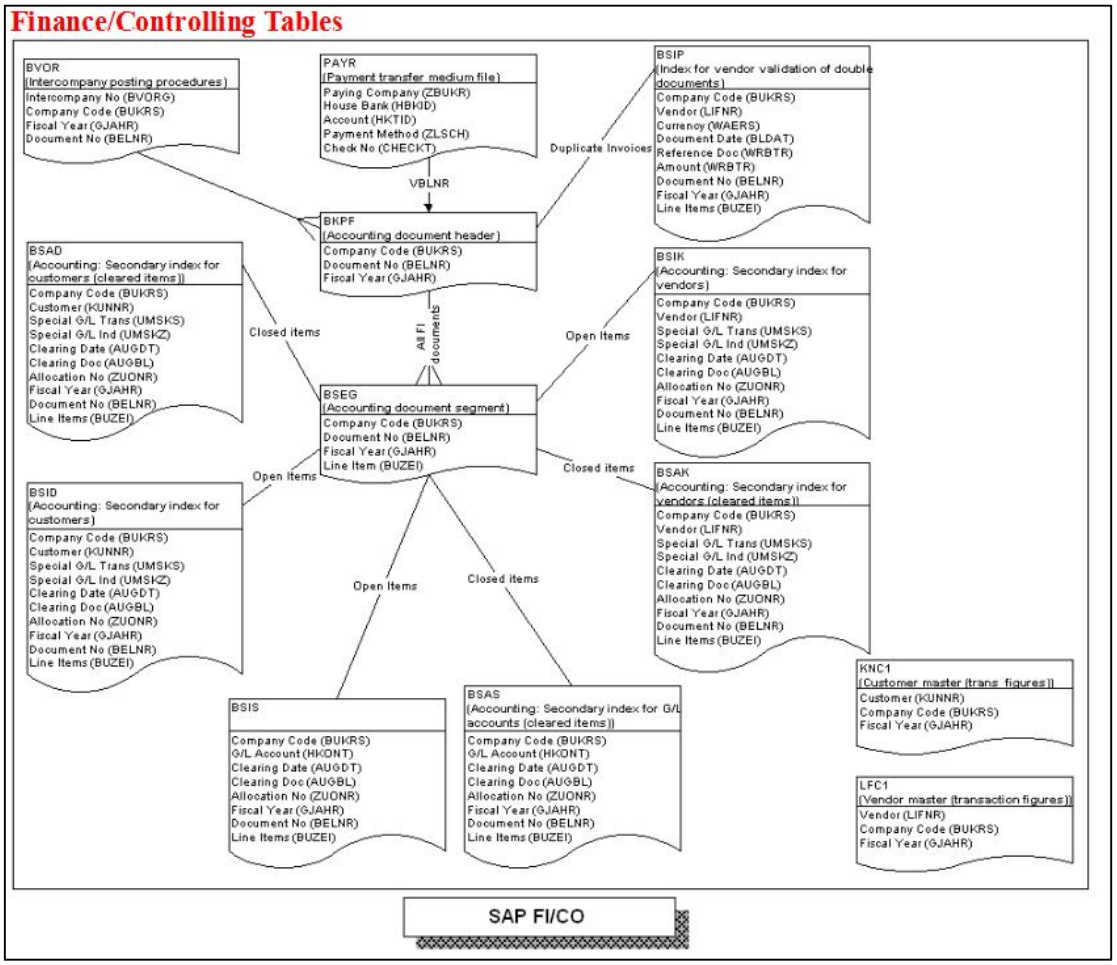

Figura 3.5 ERPGREAT (2015) Definición de la tabla BSEG. Recuperado del sitio: http://www.erpgreat.com/general/sap-r3-tables.htm

3.4.3. El IVA acreditable de Cuentas por Pagar

El proceso de IVA Acreditable inicia con una entrada de mercancías, en ese momento se genera un documento contable. Dentro grupo empresarial existen diferentes formas de contabilizar la entrada de mercancías. Una empresa tiene el siguiente flujo:

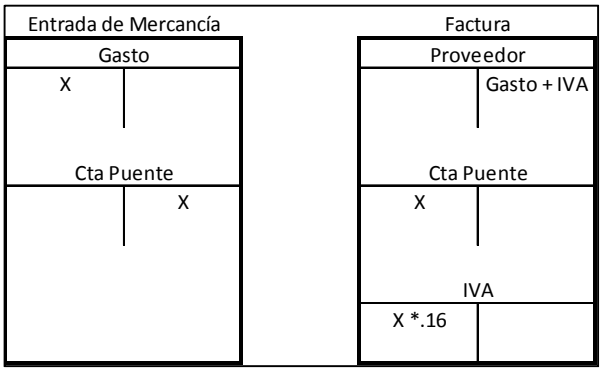

Figura 3.6 Elaboración propia (2014). Factura con cuenta puente

Una vez que se genera el documento contable de Factura, en ese momento adquiere la deuda con el Proveedor, antes de eso solo existe una cuenta de Gasto, pero no se tiene definido que gasto se va a realizar ni la Tasa de IVA al que se va a grabar dicho gasto.

Otra Filial de la Empresa tiene otra forma de manejar las facturas:

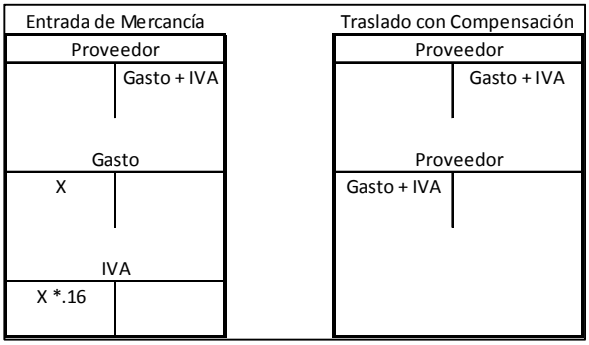

Figura 3.7 Elaboración propia (2014). Factura con Traslado con Compensación

En la figura 3.7 se observa que no hay una cuenta puente y que desde la entrada de mercancías existe una cuenta por pagar. Sin embargo, en la operación de la empresa se tomaba como una deuda con el proveedor en el momento que se hacía el documento de Traslado con Compensación. La forma de contabilizar depende de cada Contador y de la operación de cada empresa, sin embargo a mí se me hizo más fácil encontrar el Gasto y el IVA con las Facturas con Traslado con Compensación. Ya que usando Cuentas Puente es válido dividir el Gasto en Gasto con un IVA y Gasto no afecto al IVA. Por esta razón se tienen que hacer más operaciones para la búsqueda del IVA.

La figura 3.6 y 3.7 representan la entrada de mercancías y la aceptación de la cuenta por pagar. Sin embargo en ese momento no se puede hacer acreditable el IVA, el artículo 5 de la Ley del Impuesto al Valor Agregado en su párrafo 3:

"Artículo 5o.- Para que sea acreditable el impuesto al valor agregado deberán reunirse los siguientes requisitos:

III. Que el impuesto al valor agregado trasladado al contribuyente haya sido efectivamente pagado en el mes de que se trate;" (SAT, 2015).

Por tanto, la búsqueda del IVA acreditable inicia en los pagos y posteriormente hacer el recorrido en todos los documentos que fueron compensados por el documento de pago.

3.4.4. El proceso de Compensación

Las partidas abiertas deben compensarse para completar una transacción. "*Para que una transacción se considere finalizada se debe compensar. Una transacción se compensará cuando se efectúe una contabilización para una partida o grupo de partidas de forma que el saldo que resulte de las partidas sea cero"* (SAP, TF1N50_1, 2014, pág. 326). Esto es, existen cuentas que por configuración pueden ser compensadas, como por ejemplo las cuentas de Balance. Por otro lado existen cuentas que no son compensables como lo son las cuentas de Resultados.

Una posición puede compensar una o varias posiciones de un documento. Existen diferentes tipos de documentos en los que se pueden contabilizar diferentes tipos de cuentas:

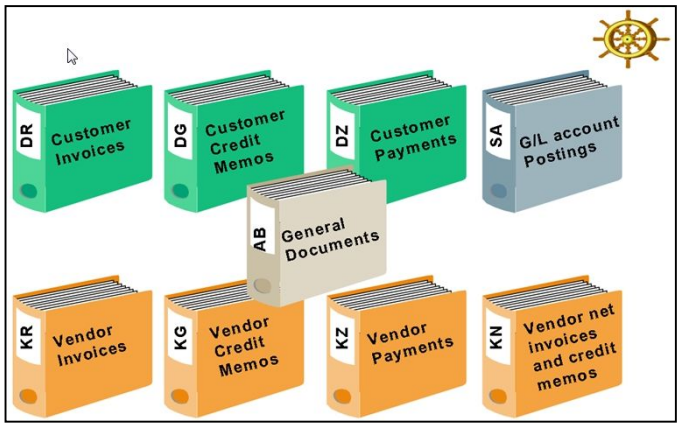

Figura 3.8 (SAP, TF1N50_1, 2014, pág. 145) Clases de documentos en SAP.

Como se observa en la figura 3.8 las clases de documentos que inician con K están dedicadas con el Proveedor mientras los documentos relacionados con el Cliente inician con D. Existen las clases de documento AB, el cual se utiliza para cualquier contabilización.

Regresando al tema de la compensación, un documento KZ corresponde a pagos a proveedores; un documento KR se refiere a una factura. Por lo tanto un documento KZ puede contener cuentas que compensen posiciones de un documento KR de la siguiente manera:

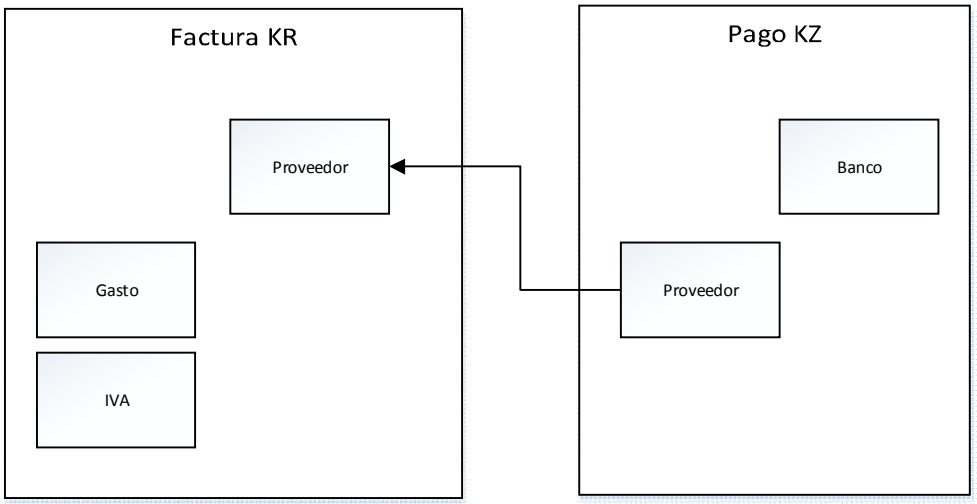

Figura 3.9 Elaboración propia (2014). Compensación de una posición de proveedor

La Factura KR contiene una cuenta por pagar abonada; el documento de pago KZ contiene una posición del Proveedor cargada. Ambas cuentas contienen el mismo importe. Por lo tanto la posición 2 del documento KZ puede compensar la posición 1 del documento KR. Una vez que se realiza la compensación el número de documento de compensación del documento KR es el número del documento KZ.

Retomando el proceso del IVA acreditable, el tema principal consistía en buscar todos los documentos que se ligaban entre el documento de pago hasta terminar la cadena de compensaciones. Si existía una cuenta de IVA acreditable dentro de alguno de los documentos recorridos se debía grabar en el reporte.

No todos los documentos iniciaban con una salida de dinero. Existe otro tipo de pago como el Factoraje. En este proceso existe un tercero, el cual es una entidad financiera. La entidad financiera absorbe la deuda que se tiene con el proveedor. En este momento el IVA ya se puede dar como acreditable, aunque la deuda con la entidad financiera aún no se liquide.

3.5 Curso de Finanzas SAP FI

El proceso del IVA acreditable se nos definió por otra colaboradora, una Funcional de *SAP*. Las tablas que debíamos consultar los desarrolladores y el proceso que teníamos que seguir lo debíamos hacer de acuerdo a sus validaciones. Para la operación, configuración y programación el Sistema *SAP* está dividido entre Funcionales y Desarrolladores. Los primeros configuran la

parametría del Sistema y definen procesos a desarrollar. Los programadores solo siguen documentos realizados por los funcionales. No estoy de acuerdo con esta política de dividir funciones. Como lo comenté desde el primer capítulo, un desarrollador debe tener el conocimiento de otras asignaturas para que las aplicaciones desarrolladas tengan mejor calidad.

Las 6 semanas más pesadas dentro de mi trabajo iniciaron en Abril de 2014 cuando tomé el curso de *SAP FI-CO* (SAP Finanzas y *Controling*). El curso está programado para 5 semanas, de lunes a viernes de 6:00 de la tarde a las 11:00 de la noche. Una semana después se tiene que presentar el examen de certificación. La sede del curso fue en las oficinas de la empresa *Compueducacion,* ubicadas en Polanco en la Ciudad de México. Mientras que mi trabajo se encuentra en Insurgentes SUR.

Junto con mi Gerente y otro compañero de trabajo tomé el curso. Teníamos que salirnos a las 5:00 de la tarde del trabajo y emprender el largo viaje. Por otro lado teníamos que presentarnos normalmente a trabajar al día siguiente a partir de las 9:00 am. Para tomar cualquier el curso la empresa nos obliga a obtener la calificación de 80/100, de lo contrario el pago del curso es por nuestra cuenta.

Colocar información del curso no sería conveniente, primero porque el curso es demasiado largo, se ven 5 manuales y algunos constan de más de 300 páginas. Tendría que hacer un trabajo completo para poder colocar lo entendido en el curso. Colocaré el temario de este:

- Parametrizaciones básicas
- Datos maestros
- Control de Documentos
- Control de Contabilización
- Compensación
- Libros de Caja
- Pagos automáticos
- Reclamación automática
- Correspondencia
- Deudores y Acreedores
- Periodificaciones
- Cierres Mensuales
- Movimientos de Activos Fijos
- Valoración y operaciones periódicas
- *List viewer.*
- Cuentas de mayor especiales

Lo aprendido dentro de este curso me ha servido mucho para mis demás desarrollos dentro de *SAP*. Por ejemplo en validaciones, ya que como programador colocaba validaciones sin saber que el Sistema *SAP* ya las tienen implícitas.

Conclusiones

Me gustaría hacer énfasis en que el presente escrito pretende integrar tres partes diferentes del ámbito laboral; en primer lugar, la parte técnica que un Ingeniero en Computación debe tener. En mi caso, la dedique a la interacción con las Bases de Datos. La parte técnica es la que nos define como Ingenieros, es decir, buscamos inventar, diseñar, construir y aplicar conocimientos en la programación y el diseño de hardware.

Para alcanzar un aceptable conocimiento técnico se necesita asistir a cursos, leer manuales, ser autodidacta y una parte fundamental es el conocimiento del idioma inglés. En mis primeros años en la empresa me ofrecieron un curso de *AS/400* que por cuestiones del idioma no pude tomar, ya que era impartido por un profesor nativo norteamericano. En su mayoría, los cursos son impartidos con material en inglés y los libros más actuales se encuentran en este idioma. Un acierto fue tomar clases por mi cuenta con una profesora Nigeriana; después de algunos meses ya podía consultar lecturas e interpretarlas de manera aceptable.

En segundo lugar se necesita tener conocimientos de otras asignaturas y así dominar la lógica de negocio de la empresa donde nos encontremos; al combinar la lógica de negocio con la parte técnica se logran mejores resultados en la labor diaria. Como lo mostré, dentro del presente trabajo la Contabilidad está muy presente.

Para dominar la lógica de negocio lo más importante es eliminar la indiferencia y la decidía que muchos desarrolladores tenemos cuando se trata de entenderla; la mayoría prefieren que un intermediario siempre les defina el proceso a desarrollar. Me ha tocado escuchar a desarrolladores que no pretenden integrarse con los usuarios, su único objetivo es tener un mejor conocimiento técnico de un lenguaje de programación.

Y en tercer lugar es necesario generar una figura de líder en nuestra persona. Ser líder en un proyecto no significa tener un puesto más alto para poder mandar a los subordinados, significa resolver los problemas para que el equipo de trabajo pueda hacer sus labores en tiempo y forma; administrar los proyectos de tal manera de integrar a todos y explotar sus capacidades. Y siempre es necesario educar con el ejemplo.

Hago un exhorto a todos los egresados de la Carrera de Ingeniería en Computación y que se dedican al desarrollo de aplicaciones: Nuestra educación no fue una carrera técnica, fue una educación integral; la cual comprende también asignaturas de las áreas administrativas, sociales y humanistas. Por lo que tenemos la capacidad de hacer un excelente trabajo en cualquier Empresa donde seamos contratados.

Bibliografía

Bernal, O. (2015). *oscarbernal.net*. Recuperado el 18 de 04 de 2015, de http://www.oscarbernal.net/index.php?/content/view/17/22/

Blanch, A. (11 de 06 de 2014). *help400*. Recuperado el 06 de 10 de 2014, de help400: http://www.help400.com/asp/scripts/nwart.asp?Num=135&Pag=32&Tip=M

CONDUCEF. (2014). *SHCP*. Recuperado el 11 de 11 de 2014, de SHCP: http://www.condusef.gob.mx/index.php/instituciones-financieras/otros-sectores/arrendadoras-financieras

Conte, P. (11 de 06 de 2014). *help400*. Recuperado el 16 de 11 de 2014, de help400: http://www.help400.es/asp/scripts/nwart.asp?Num=174&Pag=24&Tip=M

Guajardo, G. (2012). *Contabilidad para no contadores.* Mc Graw Hill.

IBM. (1997). Application Display Programming. En IBM.

IBM. (1998). *AS/400 Developer Kit para JAVA.*

IBM. (2001). *WebSphere® Development Studio: Guía del programador en ILE RPG.*

IBM. (2001). *WebSphere® Development Studio: Guía del programador en ILE RPG.* Barcelona, España.

IBM. (2006). *IBM Knowledge Center*. Recuperado el 05 de 10 de 2014, de IBM Knowledge Center: https://publib.boulder.ibm.com/infocenter/iadthelp/v6r0/index.jsp?topic=/com.ibm.etools.iseries.pgmgd.doc/c0925076330.htm

Loney, K. (2009). *ORACLE Data Base The complete reference.* McGraw Hill.

Lucina Trejo Ceseña. (2009). *PricewaterhouseCoopers*. Recuperado el 17 de 11 de 2014, de PricewaterhouseCoopers: http://www.pwc.com/es_MX/mx/publicaciones/archivo/dic09-PF-LTC-contrato.pdf

MCGrawHill. (10 de 06 de 2011). *MCGrawHill.* Recuperado el 10 de 11 de 2014, de http://www.mcgrawhill.es/bcv/guide/capitulo/8448148703.pdf

ORACLE. (2014). *Concepts 11g Release 2 (11.2).*

ORACLE. (2014). *ORACLE*. Recuperado el 23 de 11 de 2014, de http://www.oracle.com/technetwork/es/articles/sql/expresiones-regulares-base-de-datos-1569340-esa.html

SAP. (2014). *TF1N50_1.*

SAP. (2015). *Help Portal*. Recuperado el 19 de 04 de 2015, de Help Portal: http://help.sap.com/saphelp_snc70/helpdata/en/81/415d363640933fe10000009b38f839/frameset.htm

SAT. (2015). Recuperado el 15 de 02 de 2015, de http://www.sat.gob.mx/informacion_fiscal/normatividad/Paginas/leyes_2015.aspx

SAT. (11 de 04 de 2015). *SAT*. Recuperado el 19 de 04 de 2015, de SAT: http://www.sat.gob.mx/informacion_fiscal/normatividad/Paginas/leyes_2015.aspx

Scott, I. (2002). *Oracle 9i Programación PL/SQL.* Mc Graw Hill.

Soltis, F. (11 de 06 de 2014). *http://www.help400.com/asp/scripts/nwart.asp?Num=102&Pag=34&Tip=M*. Recuperado el 04 de 10 de 2014, de help400: http://www.help400.com/asp/scripts/nwart.asp?Num=102&Pag=34&Tip=M

Soule, C. (11 de 06 de 2014). *help400*. Recuperado el 05 de 10 de 2014, de help400: http://www.help400.com/asp/scripts/nwart.asp?Num=180&Pag=10&Tip=T