UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"SISTEMA DE MEDIACIÓN PARA PROCESAMIENTO DE CDRs"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTAN:

ERNESTO AGUILAR SANTANDER

CARLOS ALBERTO GARCÍA BLANCAS

KAREN ISABEL GONZÁLEZ CORTÉS

GABRIEL HERNÁNDEZ PALOMO

PATRICIA MARTÍNEZ FUENTES

ASESOR: M.I. JUAN CARLOS ROA BEIZA

MÉXICO, 2012

Dedico este trabajo a:

A Dios por poner en mi camino las circunstancias de la vida que me dieron la oportunidad y determinación necesarias para llevarlo a cabo.

A mi Madre por no perder la Fe y por dedicarme cada día un pedacito de sus oraciones y sus buenos deseos.

A mi Padre por no dejar que me faltara nada y por comprometerme con su admiración expresada desde niño a ser mejor cada día

A mis hermanas por su constante apoyo, por creer en mí y por protegerme aún cuando ya no lo necesitaba.

A mis maestros porque con el tiempo que nos dedicaron en las aulas nos dejaron parte de su vida y de su corazón.

A la Facultad de Ingeniería y a la UNAM por ser mi casa y por darme el privilegio de ser llamado Ingeniero.

A mi hija Regina para que sepa que haré todo lo que esté a mi alcance para que si así lo decide, un día ella pueda dedicar uno o más trabajos como este.

Ernesto Aguilar Santander

Dedicatorias:

A mis padres, porque creyeron en mí y porque me impulsaron a seguir adelante, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta.

A mi hija, hermana, familia y amigos.

*A la **Universidad Nacional Autónoma de México** y en especial a la **Facultad de Ingeniería** que me dieron la oportunidad de formar parte de ellas.*

Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida.

Mis palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

...y a todos aquellos que hicieron posible la elaboración y desarrollo de este trabajo.

¡Gracias!

Karen González

"Por Mi Raza Hablará el Espíritu"

A MIS PADRES

Con mucho cariño principalmente a mis padres que me dieron la vida y han estado conmigo en todo momento. Gracias papá y mamá, por creer en mí, por todos los sacrificios que hicieron a lo largo de mi carrera y la de mis hermanos, sé que fueron momentos difíciles pero gracias a sus deseos de heredarnos una buena educación salieron adelante. Gracias por toda su comprensión y paciencia siempre han estado ahí en todo momento apoyando y brindándome todo su amor de manera incondicional, por todo esto les agradezco de todo corazón, esto es un pequeño regalo de todo lo que han hecho por mí, este título es para ustedes, me siento orgulloso que por fin les entrego este título por cual luché y me esforcé teniendo como motivación y motor el gran amor que me han inculcado, se llega la meta que me había planteado y espero seguir cosechando logros los cuales siempre estarán dedicados a ustedes. Los admiro mucho, siempre los llevo en mi corazón y en mi mente que dios me los bendiga y me los cuide muchos años. Los amo.

A MIS HERMANOS

Jorge, Eduardo y Ana Lucia por todo el apoyo brindado, por su comprensión y cariño en todos estos años de estar juntos, que siempre sigamos unidos, que siempre haya sonrisas y buenos deseos entre nosotros, que siempre exista ese deseo de estar en comunicación .Los quiero mucho.

A MIS FAMILIARES

Que siempre me apoyaron en los años de carrera, son muchas las personas que influyeron en mi vida y de las cuales aprendí muchas cosas de cada uno, la lista es muy larga quiero mencionar algunos a mi Tía María que quiero mucho, especial agradecimiento a la Familia Hernández Espinosa por todos los buenos momentos que pasamos juntos Ángel, Rosangel, Carito y Citlalli gracias por todo, a la familia Nemegyei Hernández por su hermosa amistad Prisciliano, Rosario, Libertad y Priscila bellos recuerdos guardo de ustedes, a mis Abuelitos Lorenzo y Georgina (QED) que quiero mucho, a todos los tíos, primos y sobrinos de San Luis Potosí un saludo enorme para ellos y a toda mi bella Familia de Rio Grande Oaxaca los llevo siempre en mi corazón, las raíces nunca se olvidan, Jeshua para ti con mucho cariño.

A MIS AMIGOS.

Que a lo largo de la vida hemos compartido buenos momentos especialmente a Jalil y Pinky, para Antonio, Mane, Orlando, Carla, Carolina, Brenda, Rogelio, Rocío, Elizabeth, Jairo, Richard, Toño, Mario, Javier, Pepe, René, Migue, Omar, Alejandro, Raúl, Fernando, Alin, Martha, Cristóbal, Noemí, Christian, Cesar, Emmanuel, Gustavo, Suly, Viridiana para ustedes y los que me faltan gracias por su amistad.

A LA UNAM

Por ser una fuente inagotable en la formación de profesionistas humanos.

La vida no es esperar a que pase la tormenta, es aprender a bailar bajo la lluvia..."La vida es una obra de teatro que no permite ensayos..... Por eso, Canta, Ríe, Baila, Ama y Vive Intensamente cada momento de tu Vida Antes de que el telón baje y la obra termine sin aplausos...."

Gabriel Hernández Palomo

Agradezco a:

Dios, por haberme permitido terminar esta carrera, por darme sabiduría, paciencia y fortaleza para soportar las noches en vela que se requirieron para lograr esta meta.

Mis padres, José y Estela, por darme su apoyo total y no permitir que yo renunciara a esta carrera, incitándome a seguir sin importar las adversidades que se me presentarán.

Mis hermanos Verónica, José Oscar, Ricardo y María Esther, por darme su apoyo siempre que lo necesité.

Mi esposo Jericó, por ser mi apoyo incondicional, por no dejarme renunciar tanto en el transcurso de la carrera como en la realización de este trabajo, por ser mi amigo, mi cómplice y el padre de mis hijos.

Mis hijos, Edahi y Yaretzi, por ser el motivo por el cuál estoy realizando este trabajo y este objetivo en mi vida.

Mis suegros, Raúl y Rosa, por su insistencia y su empuje para lograr esta meta y recordándome siempre la importancia de la misma.

Mis amigos Erick, Juan, Leslie, Rosalba, Adrián, Adriana, Sergio, etc., por apoyarme en todos los momentos y saber que puedo contar con ellos.

La UNAM, por permitirme ser parte de la Facultad de Ingeniería y a esta por darme la formación que tengo.

Muchas Gracias a todos, espero no decepcionarlos.

Papás, la deuda que tenía con ustedes, se da por liquidada.

Patricia Martínez Fuentes.

Dedico esta tesis a:

Dedico esta tesis con todo mi cariño y amor:

> “A Dios, como creador de todo y quien me dio la fuerza y salud para conseguir esta meta.”

Carlos Alberto García Blancas

Dedico esta tesis con todo mi cariño y amor:

"A mi Esposa Araceli por darme todo su apoyo y amor así como su paciencia para concluir con una meta más así como todas las que nos faltan juntos, Gracias!"

"A mi hija Karla. Mi niña adorada, por ser siempre mi inspiración, motivación y orgullo en la vida"

"A mi hijo Gustavo. Mi príncipe, Mi muñeco de carne, por ser tan genial, sabes que eres mi inspiración, motivación y orgullo en la vida"

Carlos Alberto García Blancas

Dedico esta tesis con todo mi cariño y amor:

"A mis Padres Juan y Paty por darme todo así como la libertad de elegir y decidir sobre mi vida, esta meta y todos mis triunfos son suyos. Los quiero mucho! Padre, sigues viviendo en nuestro corazón!."

"A mi querida Hermana Ángela, que en el momento más difícil de la familia llegaste y sin saberlo fuiste nuestra inspiración y motivación para continuar."

"A todos aquellos que han confiado en mí: a mi Hermana Nadia y toda su familia, a mi Abue Ángela y Juan, a Doña U. A todos mis Tíos, Primos, Primas, amigos y compañeros."

Carlos Alberto García Blancas

Agradezco a:

A la Universidad Nacional Autónoma de México por darme tanto, es un orgullo pertenecer a ustedes y deseo que pueda corresponderles de igual manera.

A la Facultad de Ingeniería donde sufrí, reí, disfrute, aprendí y soñé, Gracias por todo!

Al PAT, gracias a su empeño y apoyo este trabajo de tesis es una realidad.

A mis compañeros de Tesis, Sin su empuje, disposición y ánimo se habría perdido el rumbo. Gracias por todo compañeros!

Carlos Alberto García Blancas

ÍNDICE

CAPÍTULO 1

SITUACIÓN ACTUAL

1.1 Introducción

Este trabajo de Tesis se desarrolla para una empresa operadora de telefonía con centrales telefónicas y cobertura en todo el país, sus sistemas de gestión residen en el Distrito Federal y su negocio principal es la telefonía celular

Por obvias razones, los sistemas de facturación son de alta prioridad para la empresa, debiendo ser capaces de procesar un gran volumen de información y sus resultados deben ser muy confiables, pese a la complejidad de unificar la información desde distintas fuentes en distintos formatos y medios de entrega, así como la complejidad de sus reglas de negocio, estas características regularmente se observan en los sistemas de empresas de telecomunicaciones.

El Sistema de Mediación (SIMED), nace con el propósito de descargar ciertas responsabilidades a los sistemas de facturación permitiendo que éstos se enfoquen en implementar sus reglas de negocio asegurando un procesamiento confiable y oportuno y evitará que continúen interactuando directamente con las plataformas de conmutación celular y por tanto sus nuevas versiones o cambios serán provocados solamente por la dinámica del negocio y no por algún cambio realizado en dichas plataformas.

El SIMED será ahora la única aplicación que interactuará con las plataformas de conmutación y será el encargado de recolectar la información correspondiente a las llamadas realizadas por los clientes tanto de telefonía móvil como de telefonía fija, garantizará el correcto transporte y almacenamiento de los archivos

generados por estas plataformas y consolidará esta información en un formato único y estándar que permita a los sistemas de facturación, de inteligencia de negocio y de aseguramiento de ingresos su consulta mediante un método sencillo y de bajo costo en su implementación.

El SIMED transformará la información recolectada de su estructura nativa a un formato estándar de base de datos que será definido por cada uno de estos sistemas cliente, facilitando el acceso y procesamiento de la información dado que el SIMED, adicionalmente, enriquecerá la información con atributos de valor de negocio.

Dentro del desarrollo del SIMED se definirá un CDR (Call Detail Record) que contenga toda la información relevante y necesaria para los sistemas cliente, así como los atributos que al SIMED le será posible agregar a partir de la información del CDR original.

Algunas centrales en su evolución comienzan a implementar protocolos de envío de CDRs muy eficientes además de la entrega por archivos, el SIMED implementará uno de estos protocolos, lo que permitirá procesar la información en un tiempo cercano al real sin necesidad de manejar archivos programados .

1.2 Operación actual.

Actualmente la compañía cuenta con presencia en la República Mexicana y cuenta además con diversos acuerdos de roaming nacional e internacional con otros operadores, lo que le permite tener cobertura nacional e internacional.

En la figura 1.2.1 se ubican las MSC (Mobile Service Center) o MTX (Mobile Telephone Exchange) con las que cuenta la compañía para hacer un total de 18 centrales.

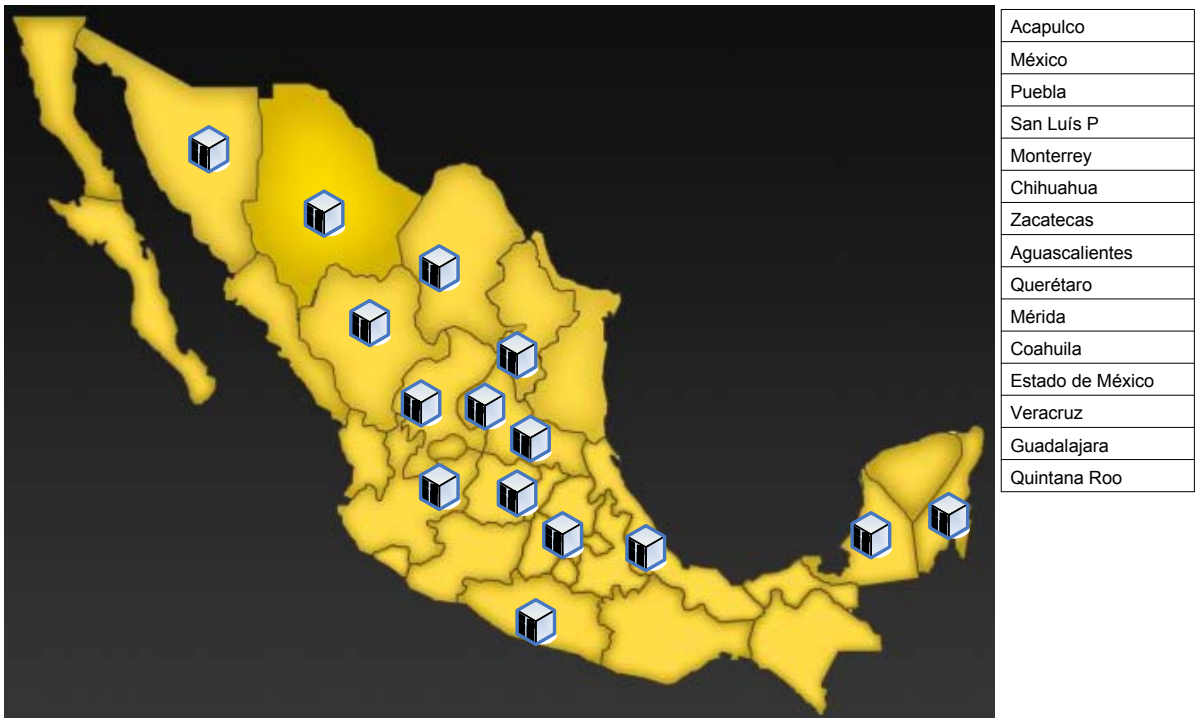

Figura 1.2.1 Mapa de cobertura a nivel nacional

Cada una de estas MSC es responsable de gestionar todas las llamadas de voz y de datos correspondientes a los clientes de la compañía registrando su actividad mediante CDRs (Call Detail Record) en ellos se almacena información detallada sobre dónde se generan las llamadas, dónde terminan y por dónde pasan, así como los números que intervienen y que regularmente son escritos por bloques de 8 en archivos binarios, por tanto todos los días se generan millones de CDRs.

Estas MSC están configuradas para cerrar los archivos de escritura de CDRs cada 2 horas o bien alcanzando un umbral de tamaño, para algunas MSC de regiones con más tráfico, como es el caso del área metropolitana, estos archivos alcanzan el umbral de tamaño cada treinta minutos en horas pico de tráfico, en otros horarios y en la mayoría de las centrales los archivos prácticamente nunca alcanzan su umbral de tamaño y son cerrados cada 2 horas. Estos tiempos y umbrales son establecidos y configurados por los fabricantes de las MSC para garantizar un óptimo desempeño de su central.

Las MSC no cuentan con un tamaño de espacio considerable para almacenamiento, solo tienen capacidad para una semana e incluso para centrales con más tráfico solo pocos días previo a alcanzar el tamaño máximo de almacenamiento, la MSC envía alarmas notificando de la situación, dado que si no puede escribir CDRs está configurada para detenerse lo que resulta grave para la empresa.

Por lo anterior, los sistemas de facturación, aseguramiento de ingresos y de inteligencia de negocio que dependen CDRs requieren llevar un control de éstos archivos y realizar sus respectivas transferencias dado que en la MSC no es posible manipularlos, está configurada para que los archivos sean consultados y nunca actualizados, las transferencias de los archivos deben ser realizadas por los protocolos que especifique el proveedor de la MSC.

En la figura 1.2.2, se muestra el flujo actual de trabajo para la recolección de CDRs de cada una de las MSC o MTX.

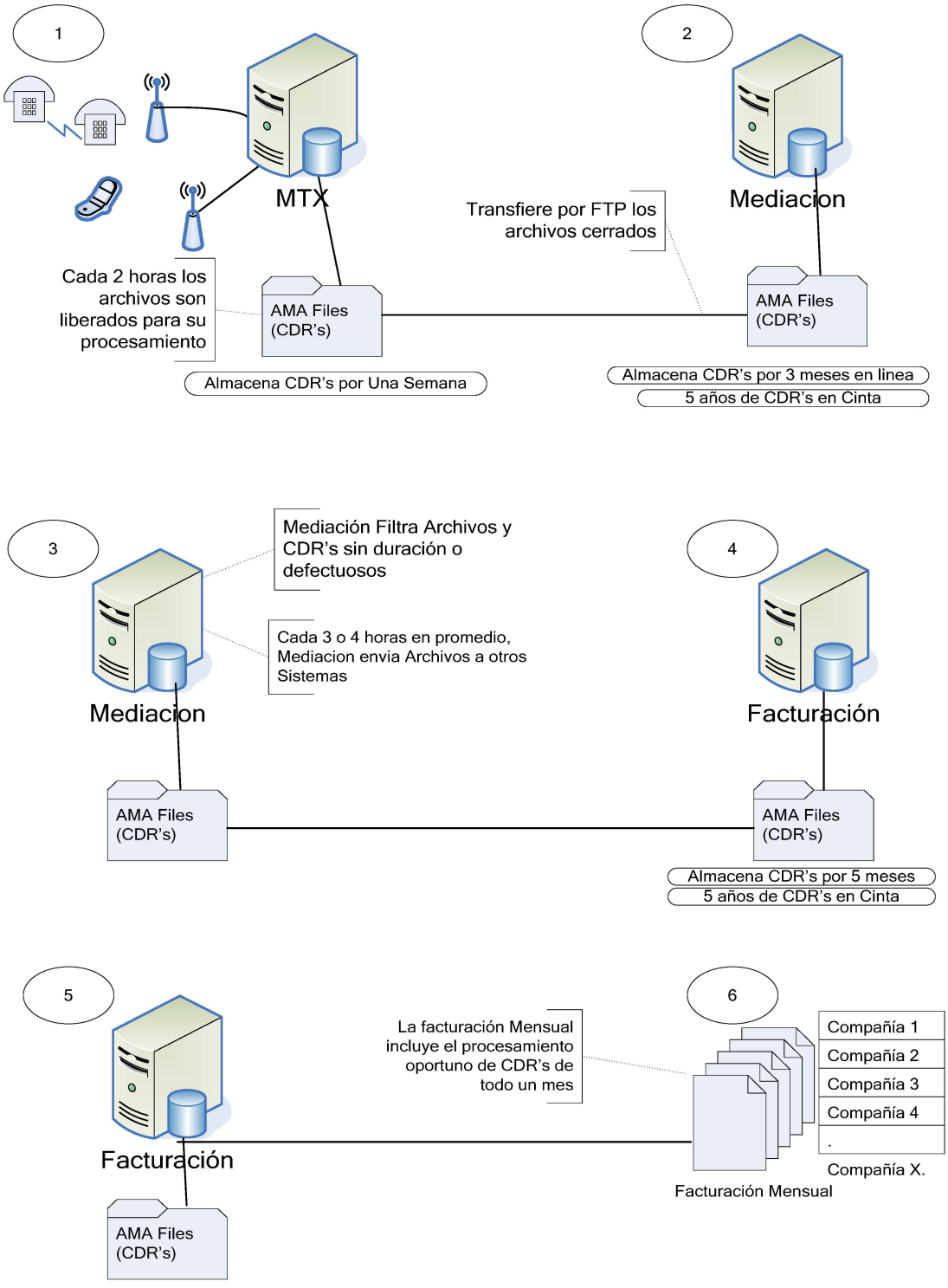

Figura 1.2.2 Flujo actual de trabajo con CDRs

Dada la criticidad e importancia de la recolección de CDRs desde la MSC para la compañía, se definió un sistema de mediación como la aplicación responsable de realizar la transferencia y recolección de estos archivos, teniendo entre sus configuraciones el tiempo en el que cada central tomará los archivos y debe de garantizar la correcta transferencia evitando pérdida de información procesándolas en la medida de lo posible de forma inmediata estos archivos para entregarlos a las aplicaciones que requieren de esta información.

El Sistema de Mediación actual tiene la capacidad de recolectar estos archivos y asegurar la correcta transferencia solamente, limitándose a filtrar aquellos CDRs que contengan duración mayor a 0 segundos y que sean contenidos en archivos bien formados, con esto entrega a los sistemas de facturación un 98% de los CDRs de entrada, sin agregar mayor valor al negocio y por tanto a los sistemas que dependen de esta información.

En la figura 1.2.3, se muestra el proceso de tiempos actual para la transferencia de CDRs desde la MSC y su procesamiento hasta los sistemas de facturación.

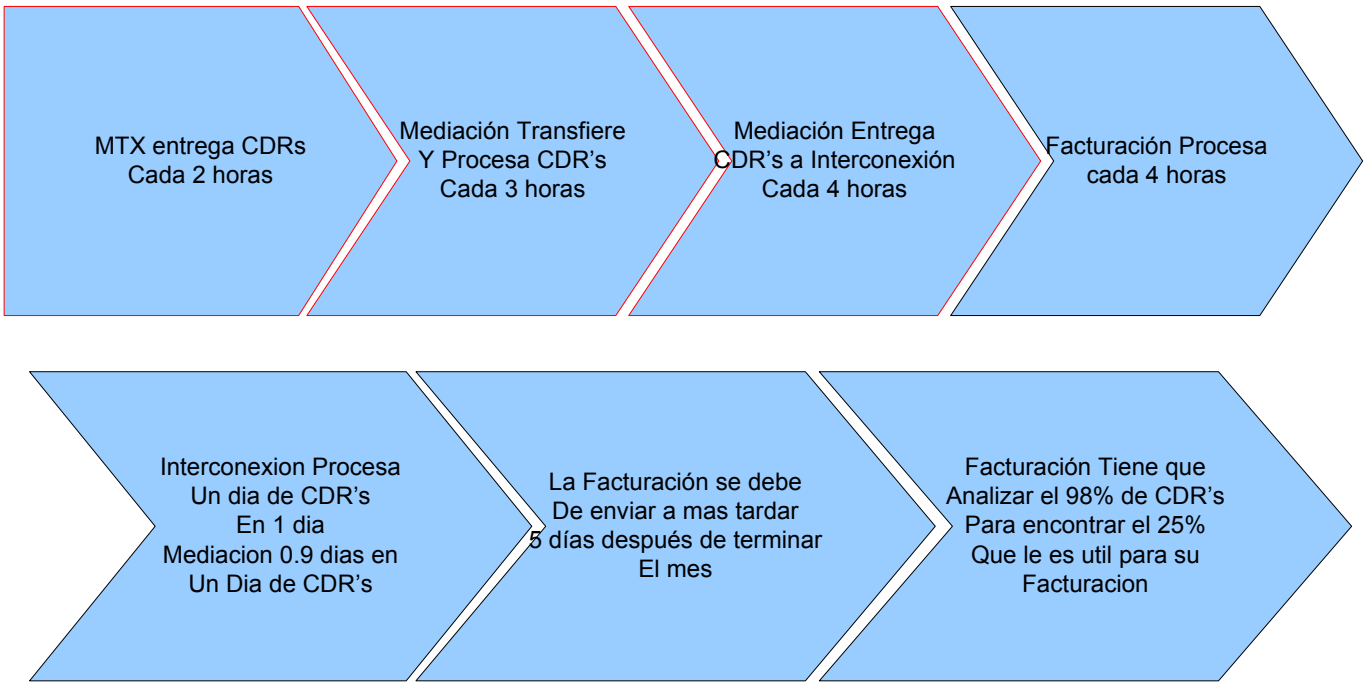

Figura 1.2.3 Procesamiento de CDRs desde la MSC.

Como se observa, los sistemas de facturación (interconexión) cuentan con un tiempo límite de entrega y con un tiempo de procesamiento poco ideal derivado también de que la mediación no aporta mucho valor a estos sistemas, éstos tienen que analizar en su totalidad los CDRs y filtrar sólo los necesarios para posteriormente aplicar sus reglas de negocio, además de que la facturación debe ser altamente confiable dado que los acuerdos de interconexión entre operadores de telefonía obligan a que la factura no tenga un error en la información mayor al 3%, si este umbral es rebasado la compañía podría verse obligada a tener que realizar algún descuento sobre la facturación y si el error es más grave podría incluso perderla.

Actualmente el Sistema de Mediación tarda 0.9 días en procesar un día de CDRs, sin embargo se estima que ese volumen comenzará a crecer derivado de un futuro incremento de clientes en la compañía, lo que obligará a mejorar u optimizar la capacidad de procesamiento del Sistema de Mediación y de los sistemas de facturación.

El sistema actual se encuentra implementado con la siguiente tecnología:

- Perl
- Lenguaje C
- MySql

Las ventajas del sistema actual son las siguientes:

- Desarrollo en casa de bajo costo
- Utiliza tecnología de software libre, lo que implica bajo costo.
- Ha soportado hasta el momento el volumen de procesamiento.
- Ha permitido a la compañía recolectar los CDRs de todas sus centrales.

Las desventajas que tiene el sistema actual son las siguientes:

- No cuenta con ningún tipo de soporte
- No existe documentación
- El conocimiento se ha diluido en la compañía debido a la rotación de personal.
- Demanda demasiada operación.
- Es necesario reiniciarlo varias veces al día.
- No refleja el comportamiento real del procesamiento de la información.

El Sistema actual de Mediación se ha intentado escalar para mejorar su desempeño sin que hasta el momento haya sido posible con resultados exitosos, los problemas de duplicidad y los reprocesos se hicieron más comunes generando incluso un incremento en tiempo y operación.

Por todo lo anterior la compañía ha decidido construir un nuevo sistema que se ajuste a sus políticas tecnológicas, utilizando Programación Orientada a Objetos (POO) con Java y Oracle, y que cumpla con la nueva dinámica del negocio y de la industria apoyando a los sistemas de facturación, aseguramiento de ingresos e

inteligencia de negocio consiguiendo con esto reducir los tiempos en la facturación.

1.3 Importancia del correcto procesamiento de CDRs.

En las redes de telefonía convencionales los sistemas que generan los CDRs y los que los procesan son entidades separadas. Esto implica por tanto, la existencia de un proceso de recolección de CDRs (históricamente mediante el envío de cintas de papel o más tarde magnéticas, posteriormente mediante FTP y hoy en día protocolos propietarios que permiten transmitirlos por bloques). Una vez recolectados deben ser validados y normalizados a nivel formato para su posterior procesamiento.

El formato en el cual se proporciona CDRs varía y es a menudo configurable, un ejemplo es el de Contabilidad de Mensaje Automático o AMA introducido al mercado por Bellcore (AMA o BAF) cuyos registros tienen una estructura compleja la cual incluye una parte de longitud fija y otra parte de longitud variable.

El diseño interno de cada MSC es propietario y los formatos de sus respectivos CDRs también lo son por tanto todo esta sujeto a la “marca” de la MSC en lugar de un estándar de la industria. Las MSC están originalmente diseñadas para autoridades de telecomunicaciones de varios países y no previeron que en determinado momento deben intercambiar información y que forman parte de una red heterogénea inclusive muchas de estas tienen protocolos de comunicación y sistemas operativos propietarios.

El proceso de recolección y procesamiento de estos CDRs es especialmente delicado, considerando que los ingresos de una compañía telefónica dependen en gran medida de la facturación hacia sus clientes y a otros operadores, es muy importante que en este proceso sea lo mas efectivo posible dado que un CDR sin facturarse representa pérdidas a la compañía.

El registro de encabezado del bloque es el primer registro en cada bloque de 2 Kbyte recolectado por el sistema automático de mensajes de contabilidad o AMA.

Indica el comienzo de un nuevo bloque de registros de llamadas y proporciona un número de bloque secuencial para cada bloque escrito en el dispositivo de grabación. La figura 1.3.2 y tabla 1.3.1 se describen los campos en el registro de encabezado del bloque.

Código de Registro	Día	Hora	Número de bloque	Código de la operadora

Figura 1.3.1 Registro de encabezado del bloque

No. de campo	Descripción	No. de dígitos	Valor
1	Código de registro: Identifica el registro	4	# C1C1
2	Tiempo: Día: Identifica el día del año Hora: Identifica la hora de entrada	5 en total 3 2	 001-365 00-23
3	Número de bloque: cuenta la secuencia n+1	5	00000-65535
4	Código de operadora: número asignado por la operadora	6	Numérico

Tabla 1.3.1 Campos del registro de encabezado del bloque

En la figura 1.3.2 muestra el mapa de bits para el encabezado del bloque registro.

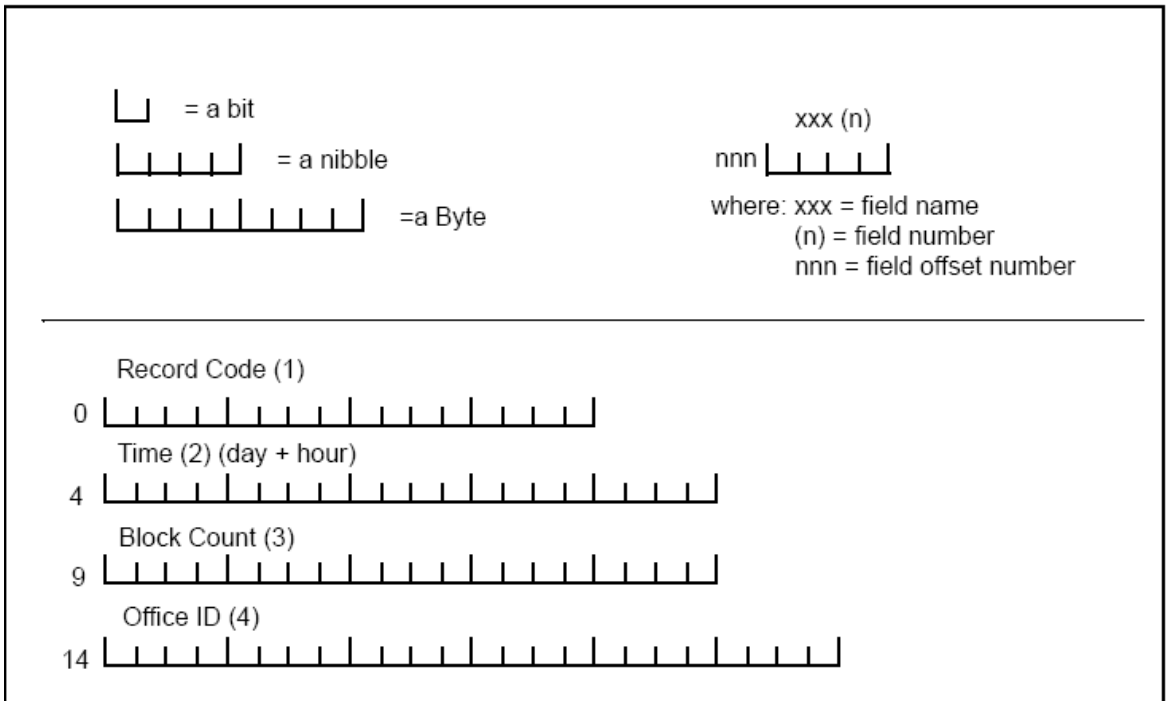

Figura 1.3.2 Mapa de bits para el encabezado del bloque registro

Un registro de detalle de la llamada se genera para cada intento. Un CDR tiene 580 nibbles de largo y tiene el formato de BCD / HEX.

La figura 1.3.3 muestra el formato de registro de detalle DMS-MTX.El CDR está compuesto por los campos que describe la MSC.

Ejemplos de campos son:

- El número que hace la llamada (el que llama)
- El número que recibe la llamada (interlocutor)
- Fecha y hora de cuando se inició la llamada
- La duración de la llamada expresada en segundos(duración)
- El identificador de la central telefónica
- Número de serie de los equipos involucrados
- Dígitos adicionales en el número de llamada para la llamada ruta o cargo
- El resultado de la llamada (si fue respondida, ocupado, etc.).

- La ruta por la que la llamada entró en la MSC.
- Tipo de llamada (voz, SMS, etc.).
- Cualquier condición de falla encontrada.

La figura 1.3.3 muestra parte del formato de un archivo CDR

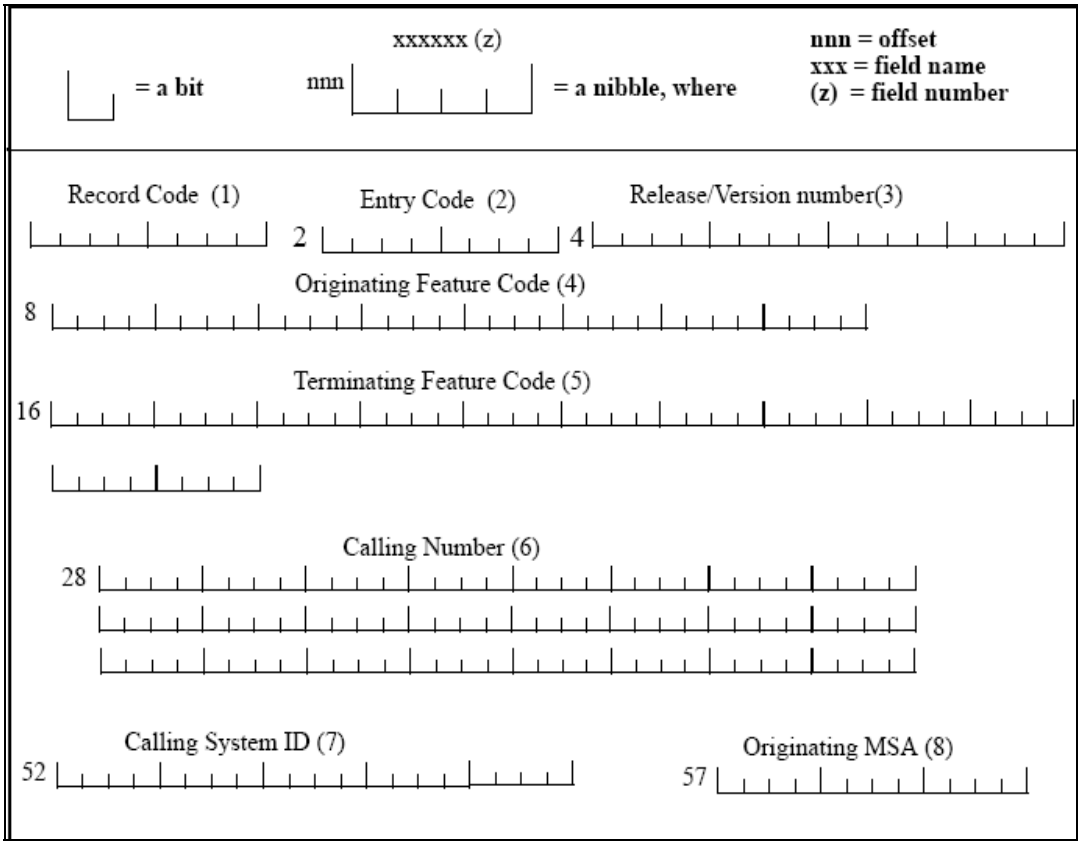

Figura 1.3.3 Formato de un archivo CDR

La figura 1.3.4 muestra detalle de los campos de un archivo CDR.

Call Detail Record format

Field No.	Data	Symbolic Field Name	No. of digits (nibbles)	Field Values And Field Type	Offset (nibbles)
1	RECORD CODE	RECCD	2	F7 HEX	0
2	ENTRY CODE	ENTRYCD	2	00 to 99 BCD	2
3	RELEASE VERSION NUMBER	VERINFO	4	00 to 99 BCD	4
4	ORIGINATING FEATURE CODE	OFEATCD	8	bitmap HEX	8
5	TERMINATING FEATURE CODE	TFEATCD	12	bitmap HEX	16
6	CALLING NUMBER	CLLNGNUM	24	set of {0 - 9, A - F} HEX	28
7	CALLING SYSTEM ID	CLLNGSYS	5	00000 to 32767 BCD	52
8	ORIGINATING MSA	ORIGMSA	3	000 to 255 BCD	57
9	ORIGINATING ROAMER INDICATOR	ORIGROAM	1	0 to 3 BCD	60
10	ORIGINATING STATION CLASS MARK	ORIGSCM	2	bitmap BCD	61
11	CALLING CATEGORY	CLNGCAT	1	1 to 11 HEX	63

Figura 1.3.4 Campos de un archivo CDR

En la figura 1.3.5 se muestra el ejemplo de un archivo CDR.

```
MTXD   MCDR100 JAN10 12:21:57 9506 INFO CDR_CALL_ENTRY
    RECCD          F7
    ENTRYCD        63
    VERINFO        1100
    OFEATCD
    TFEATCD
    CLLNGNUM       904944331
    CLLNGSYS       01236
    ORIGMSA        001
    ORIGROAM       3
    ORIGSCM        1NN
    CLNGCAT
    OCHANCAP       NNNNY
    OCHANUSE       NNNNY
    CLLNGSER       14200227466
    CLDSVC
    CCMPACCA
    DIALDNUM       9444330
    CALLDNUM       9049444330
    CALLDSYS
    TERMMSA
    TERMROAM
    TERMSCM
    CLLDCAT
    TCHANCAP
    TCHANUSE
    CALLDSER
    CALLTYPE       3
    BLLNGNUM
    ACCNTCOD
    BLNGCAT
    AUTHCODE
    TRMTCD         000
    BLLNGSER
    EVENTDIG       1
    ONWKBID
    OMTXCT
    TNWKBID        01236004000000000128
    TMTXCT         AMPS_TDMA
```

Figura 1.3.5 Ejemplo de un CDR

1.4 Principales propósitos del procesamiento de CDRs.

Alguno de los propósitos de procesar CDRs son los siguientes:

- Para el cobro a clientes: Existen servicios que se ofrecen para clientes grandes (empresas), clientes en la modalidad de pospago y clientes VIP, los cuales no están controlados por una caja o una plataforma de tarificación, por lo que estos clientes se habilitan en la red para que puedan

utilizarla (llamadas de voz, datos…etc.) y posteriormente mediante CDRs se realiza el cobro.

- Para el cobro entre operadores, se realizan cobros y pagos por uso de sus redes, esta facturación representa uno de los ingresos más importantes para la compañía pues es de varios millones de pesos mensuales.
- Para cada compañía se firma un acuerdo de entrega y recepción de facturación, se tiene que enviar llamada por llamada con duración exacta y en algunos casos no debe de tener un margen de error alto, puesto que se corre el riesgo de tener que realizar un descuento e incluso perder la factura.
- Para aseguramiento de ingresos, mediante los CDRs se puede saber con precisión cuál fue el escenario de una llamada si fue larga distancia nacional o internacional, si fue dentro de la red o fuera de la red, si lleva un cobro de interconexión o no, donde se encontraba físicamente el equipo, que número marco, si realizó una marcación especial, etc.
- Si este cliente estuviera controlado por una plataforma de tarificación se vería cuál fue el cobro que realizo esta plataforma y mediante CDRs, siguiendo las reglas de cobro, asegurar que el cobro que se realizó fue el adecuado. Lo anterior se realiza para un cliente solamente, pero se puede realizar este mismo ejercicio para varios clientes como por ejemplo para todos los clientes prepago.
- Para auditoría de la propia central, para verificar su funcionamiento, rendimiento y eficiencia, pudiendo con esto prever alguna falla en su operación o poder detectarla a tiempo. Ya que si llega a tener una falla, podría costar varios millones de pesos y esto conlleva a diferentes consecuencias, como sería el retraso del monitoreo de las llamadas, pérdida de información, etc.
- Para algún tipo de requerimiento judicial, ubicar de dónde provino una llamada, el día que se hizo, así como la hora, y con esta información poder contar con una base de datos especial, que ayude a la captura de

delincuentes que han hecho uso de la telefonía para cometer diferentes tipos de crímenes. Un ejemplo muy claro en la actualidad serían las llamadas desde los reclusorios para poder extorsionar.

- Para la generación de reportes de usuarios, esto es, que se tenga una base de datos donde se puedan verificar los movimientos que hace cada usuario en el uso de la telefonía, y que la compañía donde trabajan puedan corroborar la información con las restricciones establecidas en el uso de dicho servicio.

1.5 Diagrama de bloques del nuevo sistema propuesto.

El diagrama de bloques para el nuevo sistema de mediación (SIMED) propuesto se muestra en la figura 1.5.1.

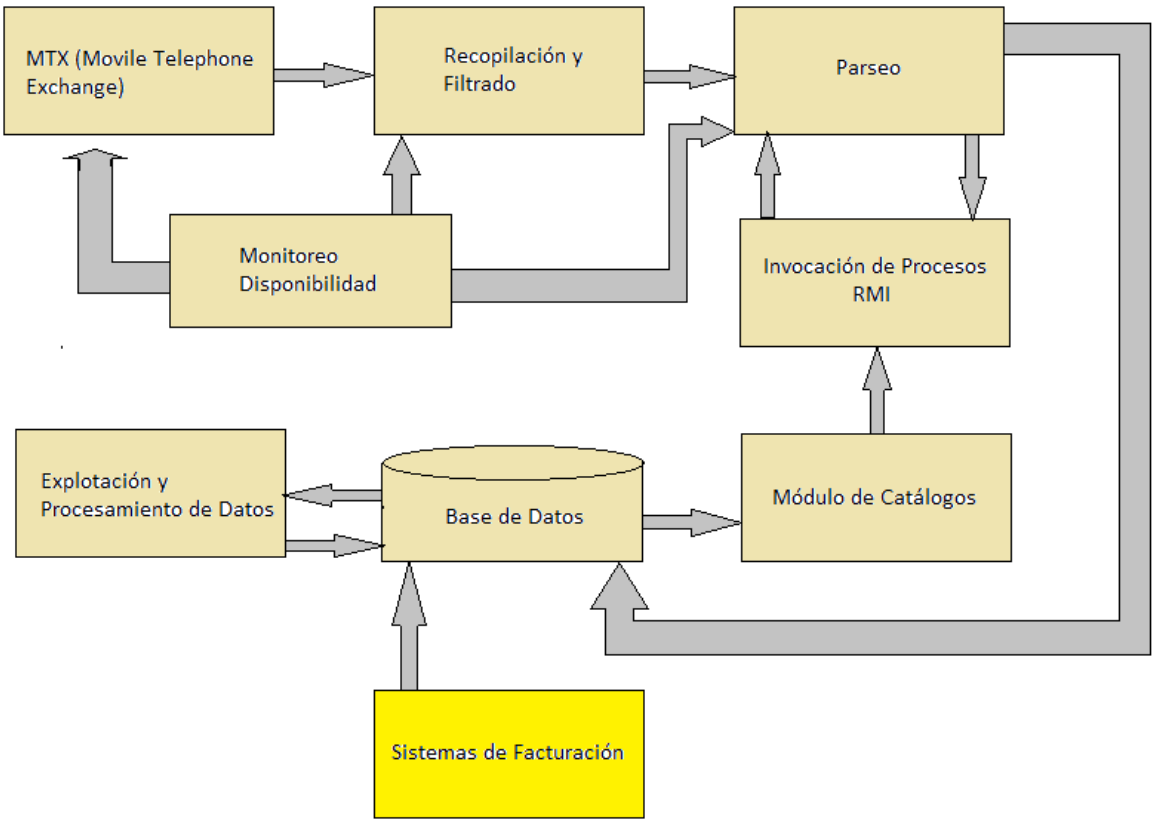

Figura 1.5.1 Diagrama de bloques del Sistema de Mediación.

El módulo de recolección y filtrado, establecerá conexiones autenticadas con cada MSC mediante el protocolo de transferencia por bloque.

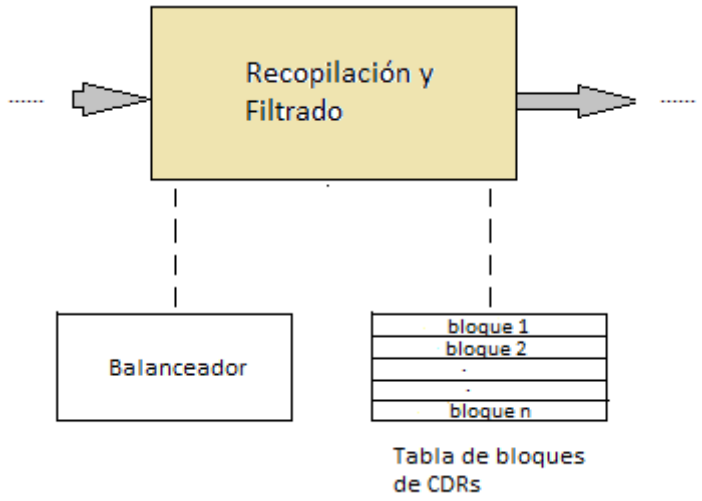

Figura 1.5.2 Bloque de recopilación y filtrado.

Como se muestra en la figura 1.5.3 se conectará mediante FTP a cada una de las MSC para transferir los archivos para su posterior conciliación. En un horario configurable por cada MSC estará verificando cuál fue el último archivo transmitido y si existe un nuevo por transferir.

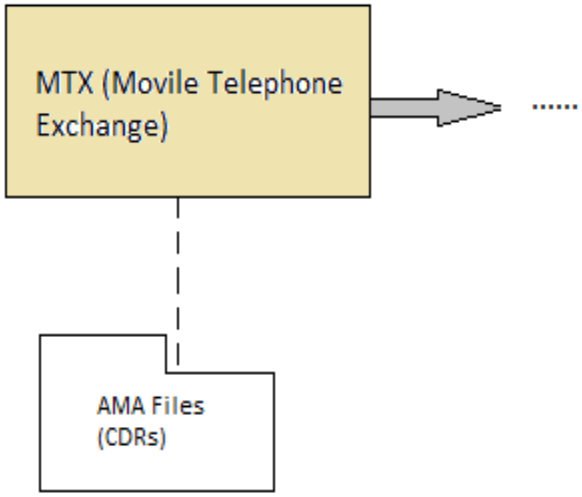

Figura 1.5.3 Bloque MTX

Realizará la conversión de estos CDRs de su estructura nativa BCD (formato original de un archivo AMA) a ASCII y posteriormente establecerá una conexión dedicada con el módulo de parseo para el envío de esta información.

El Módulo de parseo (figura 1.5.4) almacenará la información recibida dentro de una Queue de entrada para posteriormente levantar tantos threads como su configuración le permita, cada uno de éstos hilos se encargará de procesar un CDR, consultará los catálogos necesarios para complementar y homologar el CDR y posteriormente lo colocará dentro de una Queue de salida y que a su vez estará atendida por tantos threads como su configuración le permita que se encargarán de tomar estos CDRs complementados y homologados para su inserción a base de datos.

En esta etapa el módulo de parseo consumirá las interfaces publicadas mediante RMI por el módulo de catálogos dado que requiere información de los catálogos de base de datos para complementar y homologar CDR.

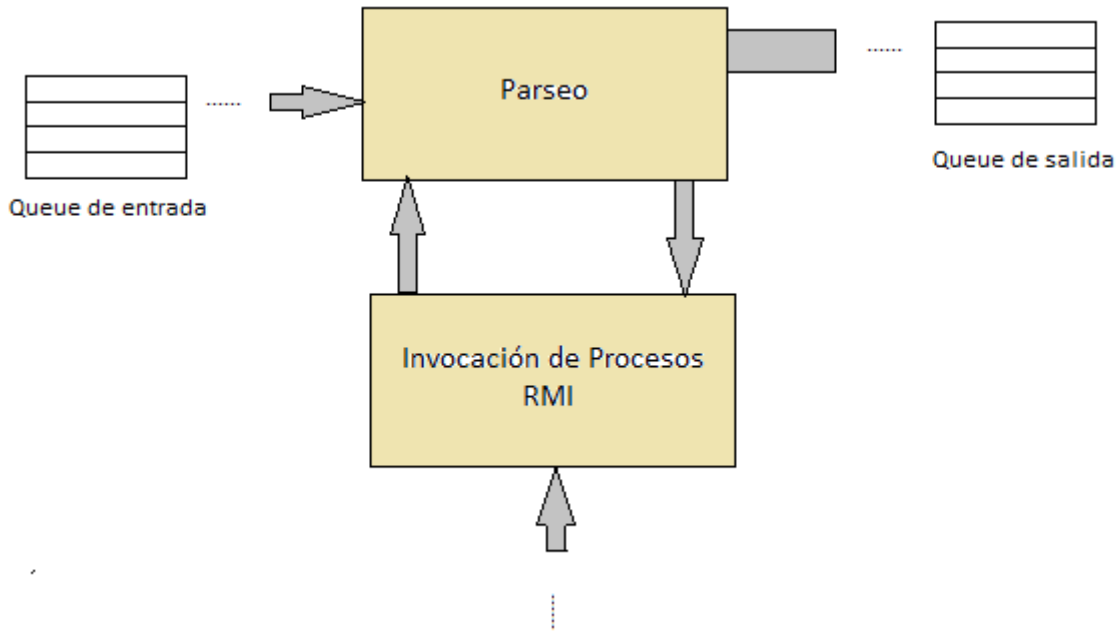

Figura 1.5.4 Sistema de Mediación.

En la base de datos del SIMED (figura 1.5.5) se almacenarán los catálogos necesarios para la aplicación como por ejemplo las series nacionales, las clientes

VIP, las celdas, las troncales, las MSCs y más, contendrá las tablas particionadas por mes y por calltype (escenario) de todos los CDRs homologados y clasificados con una historia en línea de 90 días.

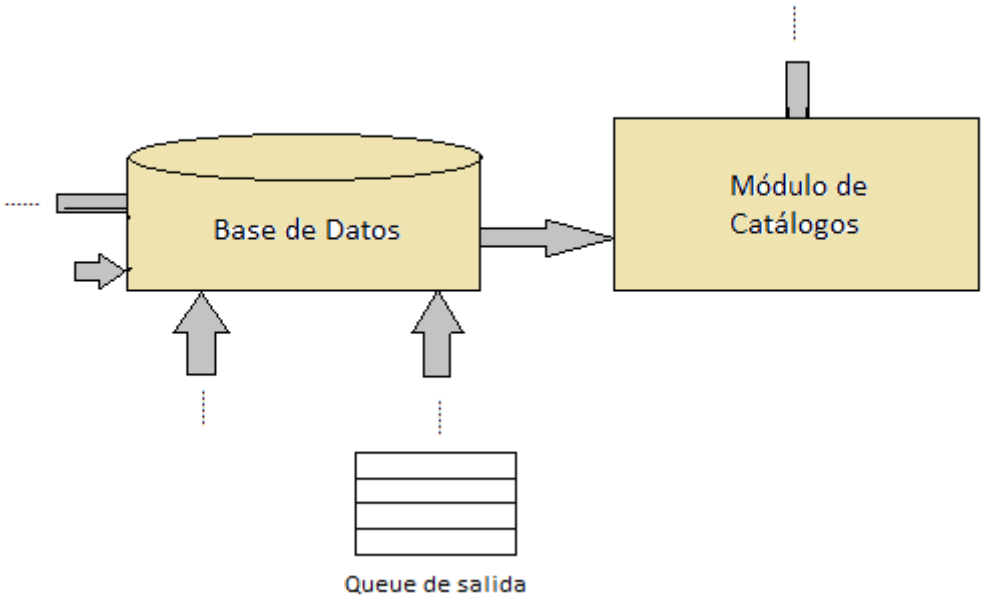

Figura 1.5.5 Resguardo y transferencia de la información por medio de la base de datos.

De igual manera la base de datos contendrá CDRs recolectados desde las plataformas de datos y de contenido multimedia para su posterior envío a los sistemas de tarificación, de aseguramiento de ingresos o de Inteligencia de negocio.

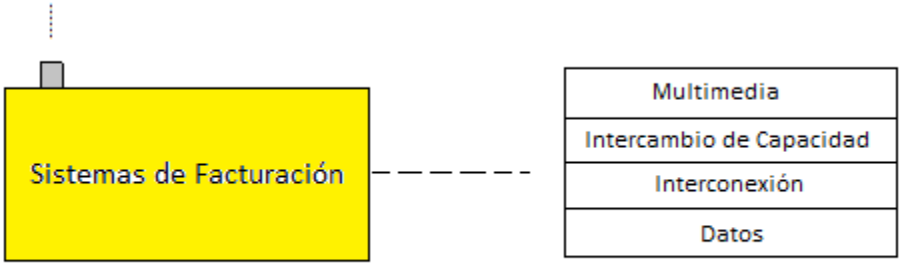

Figura 1.5.6 Plataformas del Sistema de Mediación.

Dentro de la base de datos se encontrarán todos los procesos para explotación y procesamiento de estos CDRs y todos estarán programados en el lenguaje nativo del manejador es decir PL/SQL (Procedural Languaje/Structured Query Language).

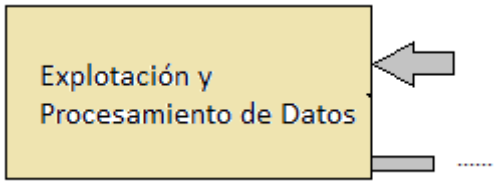

Figura 1.5.7 Bloque de explotación y procesamiento.

Con el fin de mantener la consistencia y coordinación de todos los módulos, así como la de asegurar correcto funcionamiento se realizarán verificaciones y monitoreos constantes, figura 1.5.8.

- Monitoreo del parser, revisión de queues y de threads de parseo y de BD.
- Monitoreo de los módulos de recolección.
- Proceso de conciliación entre CDRs procesados en línea vs. archivos.

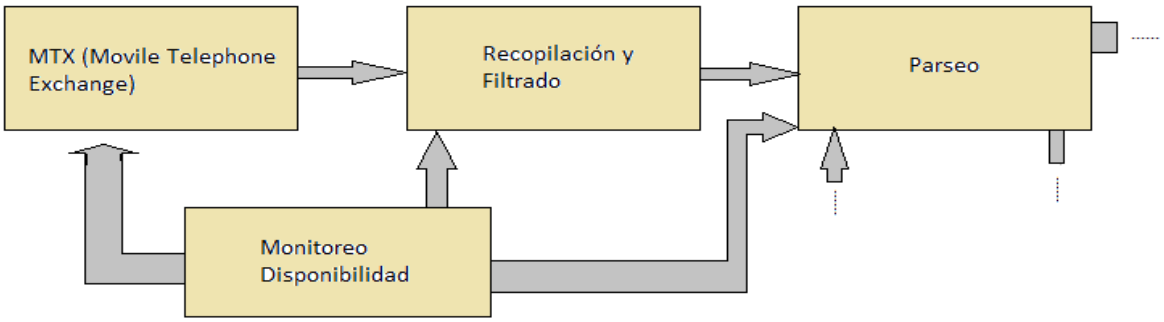

Figura 1.5.8 Monitoreo de los bloques MTX, recopilación y filtrado y parseo.

CAPÍTULO

2

MARCO TEÓRICO

2.1 Características, ventajas y desventajas de las bases de datos relacionales.

El uso constante de las tecnologías, requieren de gestión automatizada en el uso de sus sistemas de ficheros.
La tecnología informática existente, debe disponer de técnicas de bases de datos que se organicen y se gestionen por grandes volúmenes, mismas que son utilizadas en diversas disciplinas.
Uno de los pilares de cualquier organización es la información que necesita para su funcionamiento; así mismo, una de sus actividades principales es el tratamiento de dicha información el cual, ya se haga de forma manual o automática, tienen como objetivo proporcionar a las personas autorizadas la información que necesitan en el momento y el lugar adecuados. Por ello, uno de los componentes básicos de cualquier organización es su sistema de información.

Base de Datos.

Una base de datos (BD) es una colección estructurada de datos.
En esta colección, los datos deben estar estructurados de forma que reflejen fielmente los objetos, las relaciones y las restricciones existentes en el mundo real representada por la base de datos (propiedades estáticas). Para que esta

representación sea fiable, la base de datos debe reflejar los cambios que los sucesos puedan provocar en el mundo real (propiedades dinámicas).

Los mecanismos de estructuración de datos que se pueden utilizar dependen del sistema informático con el que se vaya a crear y manipular la base de datos, lo que se conoce como sistema de gestión de bases de datos.

Un Data Base Management System, en español, Sistema de Gestión de Base de Datos (SGDB), es una herramienta de software que permite la creación y manipulación de la base de datos.

Proporciona el método de organización necesaria para el almacenamiento y recuperación de grandes cantidades de datos.

Todo sistema de gestión de base de datos está basado en un modelo de datos. Estos proporcionan estructuras de datos predefinidas con sus operadores asociados (modelos clásicos), o bien mecanismos de estructuración de datos o constructores de tipos más generales.

E.F. Codd (1970) propuso el modelo relacional, el cual lo denomina como una "gestión de datos basadas en la lógica de predicados y la teoría de conjuntos utilizado para modelar problemas y administrar dinámicamente datos de forma estructurada", este modelo es elegido para la construcción de casi todo los SGDB comerciales.

En un modelo relacional se emplean tablas para poder organizar los elementos de datos. Cada fila representa una instancia de esa entidad y cada tabla corresponde a una entidad de aplicación.

Álgebra relacional.

El álgebra relacional es un conjunto de operaciones simples sobre tablas relacionales, a partir de las cuales se definen operaciones más complejas.

Dichas operaciones utilizan una o dos relaciones existentes para crear una nueva relación. Esta nueva relación puede entonces usarse como entrada para una

nueva operación. Se pueden aplicar las siguientes operaciones del algebra relacional.

- Selección: Consiste en recuperar un conjunto de registros de una tabla o de una relación indicando las condiciones que deben cumplir los registros recuperados. La selección también es conocida como consulta. En dichas consultas se utilizan diferentes operadores de comparación (=, <, >, <=, >=, <>) además de los operadores lógicos (and, or, xor) y la negación (not).
- Unión: Es utilizada para poder recuperar datos a través de varias tablas conectadas unas con otras mediante cláusulas de UNIÖN, en cualquiera de sus tres variantes, unión por la derecha, por la izquierda y unión completa (der. E izq.).
- Proyección: Es un caso específico de la selección. Es una segunda selección, en la que se seleccionan solo aquellos campos que se desean obtener.
- Asignación: Consiste en asignar un valor a uno o varios campos de una tabla.

Cálculo relacional.

El cálculo relacional es un lenguaje de consulta que describe la respuesta deseada sobre una Base de datos sin especificar cómo obtenerla.
La solución para toda consulta en este tipo de cálculo se define por:

- Una lista de resultados: Son aquellos registros que cumplen las condiciones que se desean.
- Una sentencia de cualificación: Contiene las condiciones que deseamos que cumplan los registros de la lista de resultados.

A diferencia del álgebra relacional que recurre a diferentes tablas y realiza varios pasos para obtener un resultado, el cálculo realiza la operación en un único paso gracias a lenguajes de interrogación de bases de datos.

El cálculo relacional incluye un concepto denominado cuantificador, los cuantificadores averiguan el número de registros afectados por una determinada operación, incluso antes de realizarla.

Podemos dividir a los cuantificadores en:

- Existenciales: Buscan el número de registros que devolverá una consulta.
- Universales: Indican que una condición se aplica a todas las filas de algún tipo.

Las siguientes operaciones del cálculo relacional:

- Reunión: Permite unir datos de varias relaciones.
- Diferencia: Identifica filas que están en una relación y no en otra.
- Intersección: Permite unir campos idénticos pertenecientes a las filas que son comunes en dos relaciones.
- Producto: Consiste en la realización de un producto cartesiano entre dos tablas dando como resultado todas las posibles combinaciones entre los registros de la primera y los registros de la segunda.

Cardinalidad.

La cardinalidad es la forma en la que se relacionan las entidades, o expresa cuantas entidades se relacionan con otras entidades.

- Uno a uno (1:1): Cada instancia o elemento de la entidad A está asociado solamente a un elemento de la entidad B.
- Uno a muchos (1:M): Cada instancia o elemento de la entidad A está asociado a varios elementos de la entidad B, entonces la clave que forma el vínculo entre ambas entidades, pasa hacia la entidad que tiene mayor grado de cardinalidad, es decir, el que posee la denominación “muchos”.
- Muchos a muchos (N:M): Los elementos de la entidad A están asociados a varios elementos de la entidad B, y los elementos de la entidad B están asociados a varios elementos de la entidad A, cuando esto sucede, se genera una nueva entidad denominada “Entidad Asociada”.

Normalización.

La normalización es el proceso de simplificar la relación entre los campos de un registro. En dicho proceso se somete un esquema de relación a una serie de pruebas para “certificar” si pertenece o no a una cierta forma normal. Esta se basa en las dependencias funcionales entre los atributos de una relación.
La normalización de los datos puede considerarse como un proceso de análisis de los esquemas de relación de datos basado en sus claves primarias para alcanzar las propiedades deseables de minimizar la redundancia y minimizar las anomalías de inserción, eliminación y actualización de los datos. Los esquemas de relación insatisfactorios que no cumplan determinadas condiciones, las pruebas de formas normales, se descomponen en esquemas de relación más pequeños que satisfagan dichas pruebas y que de este modo posean las propiedades deseables.

Forma normal.

La forma normal de una relación hace referencia a la condición de forma normal más alta y de este modo, indica el grado al que ha sido normalizada.

- Primera Forma Normal (1FN): Esta forma se definió para prohibir los atributos multivaluados, los atributos compuestos y sus combinaciones. Establece que el dominio de un atributo debe incluir solo valores atómicos. Así la 1FN prohíbe las “relaciones dentro de relaciones”. La 1FN también prohíbe los atributos compuestos que por sí mismos son multivaluados. Estos se denominan relaciones anidadas porque cada tupla puede tener una relación dentro de sí.
- Segunda Forma Normal (2FN): Se basa en el concepto de dependencia funcional total. Consiste en identificar que atributos no primos dependen funcionalmente de la llave primaria. La relación se encuentra en 2FN, cuando cumple con las reglas de la 1FN y todos sus atributos que no son

claves (llaves) dependen por completo de la clave. Cada tabla que tiene un atributo único como clave, está en 2FN.

Dependencia Transitiva: En una tabla (bidimensional) que tiene por lo menos tres atributos (A, B, C) en donde A determina a B y B determina a C pero este no determina a A.

- Tercera Forma Normal (3FN): Una relación estará en 3FN si y sólo si está en 2FN y todos sus atributos no primos dependen no transitivamente de la llave primaria. Se estará en 3FN si no existen dependencias transitivas entre los atributos, es decir, cuando existe más de una forma de llegar a referencias de un atributo en una relación.

Características de las bases de datos.

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios: Debido al carácter integrador que tiene la base de datos, ésta tendrá que estar compartida por distintos grupos de usuarios, lo que significa que éstos podrán acceder simultáneamente a los datos.
- Integridad de los datos: El sistema de bases de datos debe asegurar en todo momento la calidad de la información almacenada, evitando que ésta se deteriorare por un uso correcto (actualizaciones que no son válidas, accesos concurrentes no válidos, etc.).
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría: Los sistemas de bases de datos deben asegurar que la información almacenada solo la accedan las personas autorizadas y en la forma autorizada.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Ventajas de las bases de datos.

- Control sobre la redundancia de datos: Los sistemas de ficheros, almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.

  En los sistemas de bases de datos, todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

- Consistencia de datos: Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantengan consistentes.
- Compartición de datos: En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.
- Mantenimiento de estándares: Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

- Mejora en la integridad de datos: La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.
- Mejora en la seguridad: La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.
- Mejora en la accesibilidad a los datos: Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.
- Mejora en la productividad: El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.

  El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

- Mejora en el mantenimiento: En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

  Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena

en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

- Aumento de la concurrencia: En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.
- Mejora en los servicios de copias de seguridad: Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

  En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo pérdido cuando se produce un fallo.

Desventajas de las bases de datos.

- Complejidad: Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.
- Costo del equipamiento adicional: Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de

almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

- Vulnerable a los fallos: El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

2.2. Metodologías para el desarrollo de sistemas orientados a objetos

En cuanto a las metodologías orientadas a objetos, existen un gran número de métodos actualmente. Muchos de los métodos pueden ser clasificados como orientados a objetos porque soportan de manera central los conceptos de la orientación a objetos.

Se puede decir que:

- El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema.
- El diseño orientado a objetos es un método de diseño que abarca el proceso de descomposición orientado a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico del sistema que se diseña.

Algunas de las metodologías más conocidas y estudiadas se relacionan en la figura 2.2.1.

Año	Metodología
1968	Aparecen los primeros conceptos de la programación estructurada de DIJKSTRA
1974	Técnicas de programación estructurada WAGNER Y JACKSON. Jackson System Development (JSD).
1975	Primeros conceptos acerca del diseño estructurado MYERS Y JURIDOS
1977	Conceptos sobre el análisis estructurado GANE Y SARSON
1978	Conceptos sobre el análisis estructurado DEMARCO Y WEINBERG
1985	Análisis y diseño estructurado para sistemas en tiempo real de WARD Y MELLOR
1987	Análisis y Diseño Estructurado para sistemas en tiempo real de HATLEY y PIRHBAY
1991	Aparece OMT
1993	Aparece Objetory/ OOSE (Object Oriented Software Engineering) JACOBSON
1994	Aparece BOOCH Object Oriented Design (OOD)
1999	Concepto de proceso unificado de JACOBSON.
1999	Concepto de racional unified proces (RUP)

Figura 2.2.1 Metodologías orientadas a objetos

A continuación se detallan las principales metodologías de las antes mencionadas.

Metodología Jackson System Development.

La metodología JSD o desarrollo estructurado de Jackson fue desarrollada por Michael Jackson y fue especialmente popular en Europa. Sus modelos describen el mundo real en términos de entidades, acciones y secuencias de acciones. Las

entidades suelen aparecer como sustantivos en los documentos de especificaciones de requerimientos y las acciones aparecen como verbos.

JSD comienza ciertamente con unas consideraciones acerca del mundo real y en ese sentido, se podría decir que está orientada a objetos. Sin embargo, Jackson identifica pocas entidades (objetos) y muestra solo un poco de su estructura.

La aproximación JSD es compleja y difícil de comprender en su totalidad. Una razón de dicha complejidad es el fuerte uso que hace del pseudocódigo; los modelos gráficos son más fáciles de entender.

JSD es una metodóloga útil para las siguientes aplicaciones:

- Software concurrente en el cual los procesos deben sincronizarse entre sí.
- Software de tiempo real. El modelo JSD es extremadamente detallado y se centra en el tiempo.
- Programación de computadoras paralelas. El paradigma de múltiples procesos de JSD puede servir de ayuda. JSD no está bien adaptado para otras aplicaciones.
- Análisis de alto nivel. JSD no facilita una alta comprensión del problema. No es eficiente a efectos de abstracción y simplificación. Maneja meticulosamente los detalles, pero no ayuda a los desarrolladores a captar la esencia del problema.
- Bases de datos. El modelado JSD esta sesgado hacia las acciones, apartándose de las entidades y de los atributos. Por lo tanto, es una técnica poco adecuada para el diseño de bases de datos.
- Software convencional que corre en un sistema operativo. La abstracción de JSD formada por cientos o miles de procesos produce confusión y es innecesaria.

Metodología de Booch Object- Orienteed Design (OOD).

El método de Booch abarca un micro proceso de desarrollo y un macro proceso de desarrollo. El nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macro proceso.
Por eso se mantiene un enfoque evolutivo. El método Booch está soportado por una variedad de herramientas automatizadas. Booch explica que el desarrollo orientado a objetos es fundamentalmente diferente de las aproximaciones funcionales tradicionales al diseño, como pueden ser las basadas en flujo de datos.
La metodología de Booch incluye una variedad de modelos que abarcan los aspectos de objetos dinámico y funcional de un sistema de software.

Metodología de Coad y Yourdon Object Oriented Analysis (OOA).

El método de Coad y Yourdeon se considera con frecuencia como uno de los métodos más sencillos de aprender. La notación del modelado es relativamente simple y las reglas para desarrollar el modelo son evidentes.
A continuación se muestra una descripción resumida del proceso de análisis que proponen Coad y Yourdon:

- Identificar objetos usando el criterio de qué buscar.
- Definir una estructura de generalización-especificación.
- Definir una estructura de todo-parte.
- Identificar temas (representaciones de componentes de subsistemas)
- Definir atributos.

- Definir servicios. En lo referente al diseño, el método de Coad y Yourdon se desarrolló estudiando como realizan su trabajo de diseño Diseñadores especializados del software orientado a objetos.

El enfoque de diseño se dirige no solamente a la aplicación, sino también a la infraestructura para la aplicación. Una breve descripción del proceso de diseño orientado a objetos de Coad y Yourdon sigue a continuación:
Componente del dominio del problema:

- Agrupar todas las clases específicas al dominio.
- Diseña una jerarquía de clases apropiada para las clases de aplicación.
- Trabaja, cuando sea apropiado, para simplificar la herencia.
- Refina el diseño para mejorar el rendimiento
- Desarrolla una interfaz con el componente de gestión de datos
- Refina y añade objetos a bajo nivel, en caso de ser necesario
- Revisa el diseño y propone adiciones al modelo de análisis.

Componente para la gestión de tareas:

- Identificar tipos de tareas.
- Establecer prioridades
- Identificar la tarea que servirá de coordinadora para otras tareas
- Diseñar objetos apropiados para cada tarea.

La componente para la gestión de datos:

- Diseñar las estructuras de datos y su distribución.
- Diseñar servicios necesarios para manejar las estructuras de datos

- Identificar las herramientas que pueden ayudar en la implementación de la gestión de datos.
- Diseñar clases apropiadas y la jerarquía de clases.

Componente de interacción humana:

- Definir los actores humanos.
- Desarrollar escenarios para las tareas.
- Diseñar una jerarquía de órdenes de usuario.
- Refinar la secuencia de interacción del usuario.
- Diseñar clases relevantes y la jerarquía de clases.
- Integrar adecuadamente las clases de interfaz gráfica de usuario.

Metodología de Wirfs-Brock.

El método de Wirfs-Brock no hace distinción clara entre las tareas de análisis y diseño. En su lugar, propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño.
El análisis propuesto en este método se describe a continuación:

- Evaluar la especificación del cliente.
- Usar un análisis gramatical para extraer las clases candidatas de la especificación.
- Agrupar las clases en un intento de determinar superclases.
- Definir responsabilidades para cada clase.
- Asignar responsabilidades a cada clase.
- Identificar relaciones entre clases.

- Definir colaboraciones entre clases basándose en sus responsabilidades.
- Construir representaciones jerárquicas de clases para mostrar relaciones de herencia.
- Construir un diagrama de colaboraciones para el sistema. Wirfs- Brock define una secuencia de tareas técnicas en el cual el análisis conduce ineludiblemente al diseño.

Una breve descripción de las tareas relacionadas al diseño de Wirfs-Brock se indica a continuación:

- Construir protocolos para cada clase. Un protocolo es una descripción formal de los mensajes a los cuales la clase responderá.
- Refinar contratos entre objetos en protocolos refinados.
- Diseñar cada operación (responsabilidad).
- Diseñar cada protocolo (diseño de interfaz).
- Crear una especificación de diseño para cada clase.
- Describir, en detalle, cada contrato.
- Definir responsabilidades privadas.
- Especificar algoritmos para cada operación.
- Observar consideraciones y restricciones especiales.
- Crear una especificación de diseño para cada subsistema.
- Identificar todas las clases encapsuladas.
- Describir los contratos en detalle para los cuales el subsistema es un servidor.

- Observar consideraciones y restricciones especiales.

El proceso unificado.

El proceso unificado es un método de desarrollo de software configurable que se adapta a proyectos según su tamaño y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos, en el desarrollo de software de misión crítica en una variedad de industrias por la compañía Rational: Grady Booch, James Rumbaugh y Jacobson.

El proceso unificado describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura.

El proceso describe que entregables producir, como desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

El proceso unificado ha adoptado un enfoque que se caracteriza por:

- Interacción con el usuario continúa desde el inicio.
- Mitigación de riesgos antes de que ocurran.
- Liberaciones frecuentes.
- Aseguramiento de la calidad.
- Involucramiento del equipo en todas las decisiones del proyecto.
- Anticiparse al cambio de requerimientos.

Las características primordiales del proceso unificado son:

- Iterativo e incremental.
- Centrado en la arquitectura.

- Guiado por casos de uso.
- Confrontación de riesgos.

2.3 Características, ventajas y desventajas de Oracle 10g.

Oracle es un Manejador de Base de Datos Relacional ampliamente utilizado a nivel mundial y es considerado como uno de los manejadores más completos y confiables del mercado, es fabricado por Oracle Corporation cuyo logo aparece en la figura 2.3.1.

Figura 2.3.1 Logo de Oracle Corporation

La tecnología Oracle se encuentra prácticamente en todas las industrias del mundo y en las oficinas de 98 de las 100 empresas “Fortune”, es la primera compañía de software que desarrolla e implementa aplicaciones para empresas con activaciones 100% por internet y su dominio ha sido casi total teniendo entre sus competidores a Microsoft SQL Server, PostgreSQL y MySQL o Firebird antes de la compra de Sun Microsystems.

Oracle 10g salió al mercado en el 2004 en su revisión 1, y en el año 2005 en su revisión 2, no es la versión más reciente dado que la más reciente es la versión 11g, sin embargo, Oracle 10g en su revisión 2 fue la versión más innovadora de Oracle introduciéndola en un nuevo terreno en el clustering, automatización y alta disponibilidad, además del respeto y la confiabilidad ganada en la industria, esta

versión ha tenido el mayor impacto en la industria de las bases de datos inclusive provoco cambios importantes en la forma de operación de los centros de datos. Oracle 10g ofrece una arquitectura escalar única que posibilita a las aplicaciones de todo tipo aprovisionar dinámicamente servidores y recursos de almacenamiento adicionales según se requiera dependiendo de la dinámica y necesidades del negocio.

Las nuevas propiedades de automatización de Oracle 10g, redujeron los costos de su administración significativamente también por el hecho que se evita la necesidad de contar con costosas herramientas y utilidades de administración.

Características.

La figura 2.3.2 muestra los componentes principales del RDBMS Oracle 10g.

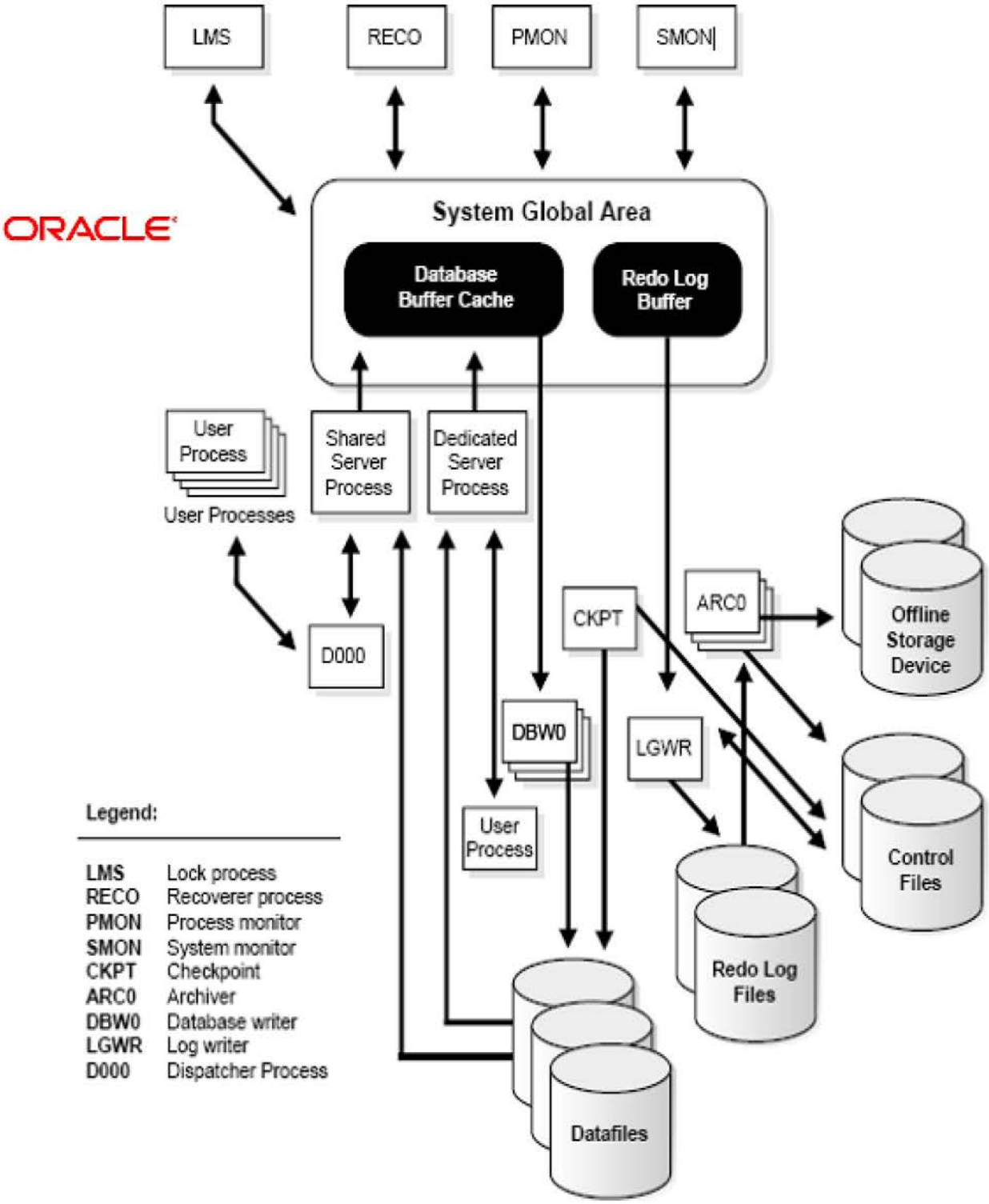

Figura 2.3.2 Componentes básicos de Oracle

- Características de desarrollo:

Característica	Descripción
Tiene múltiples ediciones	Oracle 10g cuenta con 6 ediciones: • Oracle Database Enterprise Edition (EE). • Oracle Database Standard Edition (SE). • Oracle Database Standard Edition One (SE1). • Oracle Database Express Edition (XE). • Oracle Database Personal Edition (PE). • Oracle Database Lite Edition (LE). Permitiendo seleccionar la más adecuada para las necesidades del negocio.
Es multiplataforma	Oracle 10g soporta todos los sistemas operativos productivos incluidos Windows, Unix, Linux.
Soporta cluster	Con la opción RAC toma también ventaja de la solución propuesta para clusterware, eliminando la complejidad de tener que instalar y configurar el cluster. Las capacidades de administración de almacenamiento automático distribuyen eficientemente los datos almacenados entre los discos disponibles asegurando un rendimiento óptimo.
Soporte para alto volumen de transacciones	Soporta el despliegue de un gran número de usuarios, permitiendo a la aplicación un rápido y fácil escalado desde decenas a miles de decenas de usuarios online.
Manejo de todos los tipos de datos	Soporta todos los tipos de datos relacionales estándar además del almacenamiento de XML, texto, documentos, imágenes, audio, video y datos de localización.
Múltiples formas de acceder a los datos	El acceso a los datos se realiza mediante interfaces estándar tales como SQL, JDBC, SQLJ, ODBC, OLE DB y ODP. NET, SQL/XML, XQuery y WebDAV. La lógica de negocio desarrollada en la BD puede ser escrita en Java o

	en PL/SQL .
Soporta consultas distribuidas	Soporta consultas distribuidas y transacciones entre dos o más BD, e incluye soporte para conectar vía ODBC las BDs más comunes (no de Oracle). Además, están disponibles entradas transparentes para BDs más específicas.
Particionamiento	La opción de particionamiento puede ser usada para simplificar las operaciones comunes de administración en grandes entornos de BD, con soporte para realizar estrategias de partición de datos.

- Características de administración:

Característica	Descripción
Mantenimiento sin Interrupción del servicio	Diseñado para proteger las operaciones de negocio criticas del impacto del mantenimiento de rutina. Pueden ser añadidos y usados por Oracle nuevo hardware, memoria y almacenamiento sin la necesidad de reiniciar el manejador. En la BD, las tablas pueden ser reubicadas o su tipo de almacenamiento puede ser cambiado, pueden ser añadidos o rehechos nuevos índices, pueden añadirse columnas y pueden ser eliminadas y cambiadas de nombre sin interrupciones para los usuarios finales que están accediendo a los datos.
Administración de cambios y de configuración	El paquete de diagnósticos proporciona un conjunto de diagnósticos automáticos y capacidades de monitoreo dentro de la BD, mientras que el paquete de puesta a punto (o “tuning”) ofrece una solución fácil que automatiza la compleja tarea que consume tanto tiempo del tuning de aplicaciones.

	El paquete de administración de cambios analiza las complejas dependencias asociadas con el cambio de la aplicación y automáticamente realiza los cambios requeridos en la BD, reduciendo errores, mientras que el paquete de administración de la configuración reduce la labor asociada al manejo de múltiples BD, automatizando la instalación, actualizando y clonando la BD.
Múltiples configuraciones de hardware	Soporta desde pequeñas máquinas de un solo procesador hasta entornos Symmetric Multi-Processing (SMP), los entornos cluster y grid son también soportados con la opción de Oracle Real Application Cluster.
Recuperación de errores	Proporciona un conjunto de capacidades únicas de flashback que ayudan al administrador a diagnosticar y deshacer de forma sencilla el efecto de los errores humanos incluyendo cambios en una sola fila, cambios hechos por una transacción malintencionada o bien todos los cambios hechos a una o más tablas (incluyendo la eliminación de una tabla), y todos los cambios hechos a una BD entera.
Replicación de datos	Proporciona un marco de trabajo para capturar, organizar y procesar eventos en la BD, tales como aquellos que son causados por cambios de datos o creados por las aplicaciones de negocio. Estos eventos, junto con los cambios en los datos asociados o los mensajes de aplicación, pueden ser automáticamente propagados y aplicados a una o más BDs o aplicaciones, proporcionando una solución integrada para la consulta de mensajes y la replicación de datos.

Seguridad avanzada	Proporciona características de seguridad a nivel de fila y de columna, así como características de encriptación de datos, la administración de llaves, la autentificación de proxy, el contexto de la aplicación y los roles seguros de la aplicación. A estas características se suman las características de seguridad comunes, como las revisiones, la comprobación de la complejidad de las contraseñas, los roles de la BD, los procedimientos almacenados y las funciones. Todas las comunicaciones con la BD Oracle pueden ser encriptadas.

- Características de inteligencia de negocio:

Motor de calculo OLAP	Para la inteligencia de negocio, esta versión de Oracle proporciona capacidades analíticas, estadísticas y de modelado que pueden ser accedidas directamente desde cualquier entorno basado en SQL. Estas capacidades pueden ser expandidas con el uso de las opciones de Oracle On-Line Analytical Processing (OLAP) y Data Mining (minería de datos), proporcionando un motor de calculo OLAP de gran rendimiento.
Indexación específica y optimización a consultas	Las aplicaciones de Inteligencia de negocio se beneficiarán particularmente de la indexación por mapeo de bits, de las capacidades de unión, de la re-escritura transparente de consultas y de las operaciones de consulta paralelas. Todo ello da como resultado una mejora significativa en el rendimiento de las consultas.

Ventajas

- Es el RDBMS más utilizado y documentado a nivel mundial.
- Puede ejecutarse en múltiples plataformas, desde una PC hasta arquitecturas de múltiples servidores.
- Puede ejecutarse en múltiples sistemas operativos, desde Windows hasta Unix.
- Soporta todas las funciones de un RDBMS “Formal” y de alto volumen, así como un lenguaje de diseño muy completo (PL/SQL).
- Mediante PL/SQL se pueden programar triggers y procedimientos almacenados, con una integridad referencial declarativa poderosa.
- Es ideal para empresas que necesita soporte de un alto volumen de transacciones y aplicaciones con almacenamiento intensivo de datos.
- Introduce nuevos conceptos como índices de mapas de bits, vistas materializadas, particionamiento, tablas externas y más.

Desventajas

- Su mayor inconveniente es el precio de sus licencias.
- Sus licencias además de caras pueden llegar a ser vendidas incluso a nivel procesador.
- Obliga a sus clientes a estarse actualizando dado que las versiones anteriores pierden soporte.
- Elevado costo en la formación oficial, mediante su división de “Educación” manejando incluso niveles de certificación.

2.4. Características, ventajas y desventajas del JDK 1.5.

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código de máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Componentes

Bibliotecas de Java, que son el resultado de compilar el código fuente desarrollado por quien implementa la JRE (Java Runtime Environment) y que ofrecen apoyo para el desarrollo en Java. Algunos ejemplos de estas bibliotecas son:

Las bibliotecas centrales, que incluyen:

- Una colección de bibliotecas para implementar estructuras de datos como listas, arrays, árboles y conjuntos.
- Bibliotecas para análisis de XML.
- Seguridad.
- Bibliotecas de internacionalización y localización.

Bibliotecas de integración, que permiten la comunicación con sistemas externos. Estas bibliotecas incluyen:

- La API para acceso a bases de datos JDBC (Java DataBase Conectivity).

- La interfaz JNDI (Java Naming and Directory Interface) para servicios de directorio.
- RMI (Remote Method Invocation) y CORBA para el desarrollo de aplicaciones distribuidas.

Bibliotecas para la interfaz de usuario, que incluyen:

- El conjunto de herramientas nativas AWT (Abstract Windowing Toolkit), que ofrece componentes GUI (Graphical User Interface), mecanismos para usarlos y manejar sus eventos asociados.
- Las Bibliotecas de Swing, construidas sobre AWT pero ofrecen implementaciones no nativas de los componentes de AWT.
- APIs para la captura, procesamiento y reproducción de audio.
- Una implementación dependiente de la plataforma en que se ejecuta de la máquina virtual de Java (JVM), que es la encargada de la ejecución del código de las bibliotecas y las aplicaciones externas.
- Plugins o conectores que permiten ejecutar applets en los navegadores Web.
- Java Web Start, para la distribución de aplicaciones Java a través de Internet.
- Documentación y licencia.

JDK/J2SE.

JDK (Java Development Kit) y J2SE (Java 2 Standard Edition) son nombres para el mismo componente utilizado en ambientes Java, el cual agrupa las diversas funcionalidades necesarias para desarrollar programas Java.

El conjunto de clases base proporcionadas por el JDK/J2SE incluyen clases de uso común tales como: manipulación de string's, fechas y números así como todas las funcionalidades *base* esperadas de un lenguaje computacional como:

ordenamiento y manipulación de arreglos, operaciones matemáticas complejas (trigonométricas y exponenciales) y escritura/lectura de "streams".

Además de estas clases base, el JDK/J2SE posee otra serie de clases especializadas que también ofrecen la base para la creación de otros componentes Java que incluyen: applets, objetos CORBA, fragmentos auditivos, soporte de criptografía (seguridad) y manipulación de XML entre otras funcionalidades.

J2SE 5.0 (30 de septiembre de 2004) — nombre clave: *Tiger*. (Originalmente numerado 1.5, esta notación aún es usada internamente) desarrollado bajo JSR 176, Tiger añadió un número significativo de nuevas características.

- Plantillas (genéricos) — provee conversión de tipos (type safety) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversión de tipos (type casting).
- Metadatos — también llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades.
- Autoboxing/unboxing — Conversiones automáticas entre tipos primitivos (como los int) y clases de envoltura primitivas (como integer).
- Bucle for mejorado — La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente “Iterable”, como la clase estándar “Collection”.

Características

- Lenguaje simple. Se lo conoce como lenguaje simple porque viene de la misma estructura de C y C++; ya que C++ fue un referente para la creación

de java por eso utiliza determinadas características de C++ y se han eliminado otras.

- Orientado a objeto. Toda la programación en java en su mayoría está orientada a objeto, ya que al estar agrupados en estructuras encapsuladas es más fácil su manipulación.
- Distribuido. Permite abrir sockets, establecer y aceptar conexiones con los servidores o clientes remotos; facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.
- Robusto. Es altamente fiable en comparación con C, se han eliminado muchas características con la aritmética de punteros, proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.
- Seguro. La seguridad es una característica muy importante en java ya que se han implementado barreras de seguridad en el lenguaje y en el sistema de ejecución de tiempo real.
- Indiferente a la arquitectura. Java es compatible con los más variados entornos de red, cualesquiera que sean estos desde Windows 95, Unix a Windows Nt y Mac, para poder trabajar con diferentes sistemas operativos.
- Java es muy versátil ya que utiliza byte-codes que es un formato intermedio que sirve para transportar el código eficientemente o de diferentes plataformas (Hardware - Software).
- Portable. Por ser indiferente a la arquitectura sobre la cual está trabajando, esto hace que su portabilidad sea muy eficiente, sus programas son iguales en cualquiera de las plataformas, ya que Java especifica tamaños básicos, esto se conoce como la máquina virtual de Java.
- Interpretado y compilado a la vez. Java puede ser compilado e interpretado en tiempo real, ya que cuando se construye el código fuente este se transforma en una especie de código de máquina.
- Multihebra o Multihilos. Java tiene una facilidad de cumplir varias funciones al mismo tiempo, gracias a su función de multihilos ya que por cada hilo

que el programa tenga se ejecutarán en tiempo real muchas funciones al mismo tiempo.

- Dinámico. El lenguaje java es muy dinámico en la fase de enlazado, sus clases solamente actuarán en la medida en que sean requeridas o necesitadas con esto permitirá que los enlaces se puedan incluir desde fuentes muy variadas o desde la red.
- Produce applets. En java se pueden crear aplicaciones independientes y applets.
- Alto rendimiento. Java es considerado de alto rendimiento por ser tan veloz en el momento de correr los programas y por ahorrase muchas líneas de código.

Ventajas

- El JDK es una herramienta libre de licencias (sin costo), creada por Sun, está respaldado por un gran número de proveedores.
- Existe soporte dado por Sun.
- Debido a que existen diferentes productos de Java, hay más de un proveedor de servicios.
- Sun saca al mercado cada 6 meses una nueva versión del JDK.
- Es independiente de la plataforma de desarrollo.
- Existen dentro de su librería clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.
- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.
- El conocimiento sobre tecnología Java está en alto crecimiento en el mercado.

- Se puede acceder a bases de datos fácilmente con JDBC, independientemente de la plataforma utilizada o el manejo de las bases de datos es uniforme, es decir transparente y simple.
- El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no hacer con los recursos críticos de una computadora.
- Es una fuente abierta, así que los usuarios no tienen que luchar con los gastos sobre patente y son Independientes de la plataforma.
- Java realiza la colección de basura de las ayudas, así que la gestión de memoria es automática.
- Usando Java podemos desarrollar aplicaciones web dinámicas.
- Permite que se pueda crear programas modulares y códigos reutilizables.

Desventajas

- La velocidad. Los programas hechos en Java no tienden a ser muy rápidos, supuestamente se está trabajando en mejorar esto. Como los programas de Java son interpretados nunca alcanzan la velocidad de un verdadero ejecutable.

- Java es un lenguaje de programación. Esta es otra gran limitante, por más que digan que es orientado a objetos y que es muy fácil de aprender sigue siendo un lenguaje y por lo tanto aprenderlo no es cosa fácil. Especialmente para los no programadores.
- Java es nuevo. En pocas palabras todavía no se conocen bien todas sus capacidades.
- Hay diferentes tipos de soporte técnico para la misma herramienta, por lo que el análisis de la mejor opción se dificulta.
- Para manejo a bajo nivel deben usarse métodos nativos, lo que limita la portabilidad.

- El diseño de interfaces gráficas con awt y swing no es simple o existen herramientas como el JBuilder que permiten generar interfaces gráficas de manera sencilla, pero tienen un costo adicional.
- Puede ser que no haya JDBC para bases de datos poco comerciales.
- Algunas herramientas tienen un costo adicional.
- El código Java puede ser a veces redundante en comparación con otros lenguajes.
- Java no dispone de operadores de sobrecarga definidos por el usuario., esta característica podía complicar la lectura y mantenimiento de los programas.

Pero en general Java posee muchas ventajas y se pueden hacer cosas muy interesantes con esto. Hay que prestar especial atención a lo que está sucediendo en el mundo de la computación, a pesar de que Java es relativamente nuevo, posee mucha fuerza y es tema de moda en cualquier medio computacional. Muchas personas apuestan a futuro y piensan en Java.

2.5. Conceptos básicos de mediación y comunicaciones.

Conceptos de comunicaciones.

- Comunicación de datos: Es el proceso de comunicar información en forma binaria entre dos o más puntos. Requiere cuatro elementos básicos (figura 2.5.1) que son:
    - Emisor: Dispositivo que transmite los datos.
    - Mensaje: Lo conforman los datos a ser transmitidos.
    - Medio: Consiste en el recorrido de los datos desde el origen hasta su destino.
    - Receptor: Dispositivo de destino de los datos.

DIAGRAMA DEL PROCESO DE LA COMUNICACIÓN.

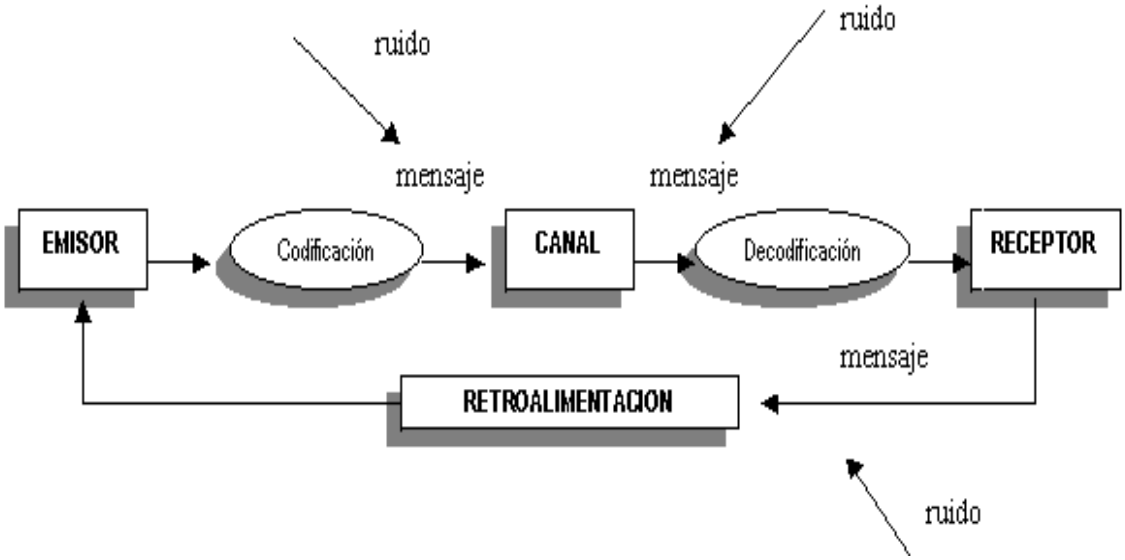

Figura 2.5.1 D Diagrama del proceso de la comunicación

- Codificación: Es la conversión de un sistema de datos a otro sistema llamado destino. De esto se desprende que la información resultante es equivalente a la información de origen.
- Decodificación: Es el proceso del cual el receptor interpreta el mensaje y lo traduce en información significativa. El receptor debe recibir primero el mensaje y luego interpretarlo.
- Retroalimentación: Es el último eslabón del proceso de comunicación, es el paso que cierra el circuito, poniendo el mensaje de respuesta devuelta en el sistema. La forma en que se puede saber si la comunicación se logró efectivamente es a través de la retroalimentación que nos dé el receptor, por medio de su reacción o respuesta.
- Protocolos: Conjunto de reglas que posibilitan la transferencia de datos entre dos o más computadoras.
- FTP: En inglés File Transfer Protocol, protocolo de conexión para extraer archivos entre equipos.
- Trama: Grupo de bits con un formato predefinido usado en protocolos orientados a bits.

- Parser: Proceso de clasificación que genera el CDR.
- Paquetes: Cantidad mínima de datos que se transmite en una red o entre dispositivos. Tiene una estructura y longitud distinta según el protocolo al que pertenezca. También llamado TRAMA.
- Códigos: Acuerdo previo sobre un conjunto de significados que definen una serie de símbolos y caracteres. Toda combinación de bits representa un carácter dentro de la tabla de códigos. Las tablas de códigos más reconocidas son las del código ASCII y las del código EBCDIC (Extended Binary Coded Decimal Interchange Code, Código Ampliado de Intercambio decimal codificado en binario).
- ASCII: Es un código de siete bits, usa cadenas de bits representables con siete dígitos binarios (que van de 0 a 127 en base decimal), para representar información de caracteres.
- Hilos de transmisión: Para describir el circuito que compone un canal se utiliza el término “pares”. Uno de los hilos del par sirve para transmitir o recibir datos y el otro es la línea de retorno electrónico.
- Switch: Equipo que permite realizar la conmutación de llamadas entre dos equipos de comunicación de telefonía celular.
- Interconexión: Sistema que genera las facturas correspondientes para otros operadores de telefonía con los que se tengan firmados contratos de interconexión o intercambio de capacidad.
- Plataforma de Tarificación: Sistema que maneja el saldo al cliente.
- AAA (Authentication, Authorización, Accounting): Sistema utilizado para prestar el servicio de envío y recepción de datos, ya sea para Web o PTT.
- Multimedia: Sistema que proporciona el servicio para extraer los archivos de Ringtones e imágenes de los diferentes proveedores.
- Proceso de facturación: Es la etapa en la que se realiza el conjunto de actividades mediante la cual se generan las facturas correspondientes a los consumos de los usuarios o suscriptores de los servicios públicos de telecomunicaciones.

- Proceso de tarificación: Es la etapa en la que se realiza el conjunto de actividades mediante la cual se le aplica un valor monetario a los consumos medidos en el proceso de tasación.
- Proceso de tasación: Es la etapa en la que se realiza el conjunto de actividades mediante la cual se mide el consumo de los usuarios o suscriptores de los servicios públicos de telecomunicaciones.
- Servidores: Computadoras dedicadas para proporcionar servicios a otras computadoras tales como servicios web, bases de datos, almacenamiento en discos, acceso a las impresoras, unidades para respaldo de archivos, acceso a otras redes o computadoras centrales.

El CDR en sus siglas en inglés es Call Details Record o en español Registro con el detalle de una llamada.

El CDR está compuesto por los siguientes campos:

- El número que hace la llamada (el que llama).
- El número que recibe la llamada (interlocutor).
- Cuando se inició la llamada (fecha y hora).
- La duración de la llamada.
- El cobro al número de teléfono por la llamada.
- El identificador de la central telefónica.
- El número de serie que identifica el registro.
- Dígitos adicionales en el número de llamada.
- El resultado de la llamada (si fue respondida, ocupado, etc.).
- La ruta por la que la llamada entró.
- Tipo de llamada.
- Condiciones de falla encontrados.

Proceso de mediación.

La mediación implica recolectar y dar formato a los registros a utilizar. El sistema de mediación actúa como un buffer valorizado, desacoplando la infraestructura de los switches desde el sistema de soporte operacional y permitiendo a cada uno ser configurado independientemente de los demás.

Los Sistemas de Mediación implementan las siguientes funciones genérales:

- Recepción de datos de CDRs desde las centrales.
- Control en la recepción de archivos, secuencia de generación de archivos, recepción de archivos duplicados, etc.
- Planificación del procesamiento de archivos.
- Conversión de los archivos con CDRs originales, en archivos en el formato requerido por el sistema de facturación. Esto implica validación y filtrado de ticket de acuerdo a los escenarios de registro definidos para facturar.
- Creación de copias de respaldos de los archivos con CDRs recibidos.
- Envío de los archivos con CDRs a sistemas de facturación, de análisis de fraude, de control de tráfico, búsquedas, control de consumo telefónico, etc.

La conexión con las centrales telefónicas y nodos, desde el Sistema de Mediación se realiza por medio de módulos que resuelven las particularidades de comunicación con cada tipo de central o nodo (ver figura 2.5.2).

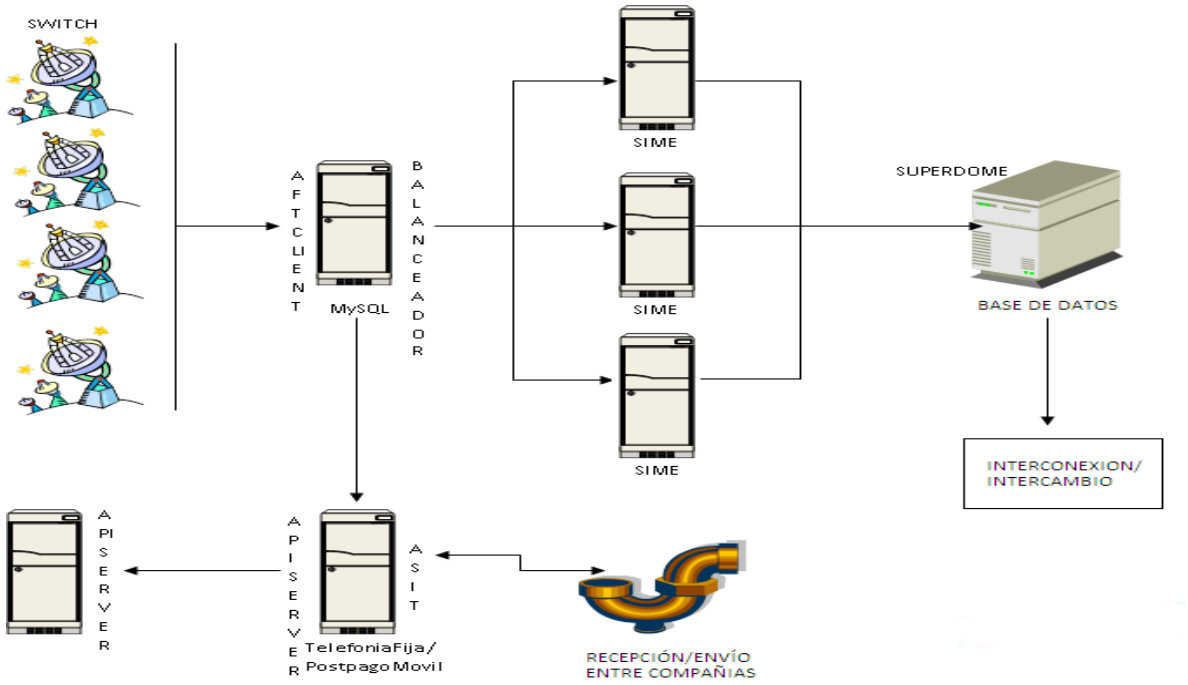

Figura 2.5.2 Proceso de mediación.

Dependiendo del tipo de fuente de datos, varía la forma en que se reciben los archivos. En algunos casos la comunicación es iniciada por la central y en otros la misma se inicia desde el Sistema de Mediación.

El sistema cuenta con un módulo de control que se encarga de procesar los archivos provenientes de las centrales, revisar desfasamiento en la recepción de archivos, controlar y autorizar la recepción de reenvíos así como generar alarmas.

Una vez recibidos los archivos de tasación originales se planifica su procesamiento por los módulos de validación y formateo de CDRs, con el objetivo de enviarlos al sistema de facturación de la empresa.

Por lo que el Sistema de Mediación es un proceso en donde los datos son transformados a datos formateados. Esta transformación precisa una plataforma distinta que conecta los dos componentes de un sistema de telecomunicaciones, la red switcheada y los sistemas de información del negocio, mientras se asegura que estas funcionen independientes unas de otras.

El Sistema de Mediación recolecta una gran cantidad de datos útiles en un tiempo determinado, en forma segura y confiable desde varias tecnologías distintas.

CAPÍTULO

3

ANÁLISIS Y PLANTEAMIENTO DEL PROBLEMA

3.1. Análisis del proceso actual de transferencia de CDRs.

Actualmente el Sistema de Mediación de la compañía, se encuentra a su máxima capacidad de procesamiento, darle mantenimiento resulta muy costoso, presenta diversos problemas operativos y de funcionamiento, es necesario invertir una gran cantidad de horas hombre en su operación.

Los tiempos de entrega de la información hacia sus sistemas cliente no son los ideales, lo que en casi la totalidad de casos atenta directamente con la captación de ingresos de forma oportuna a la compañía.

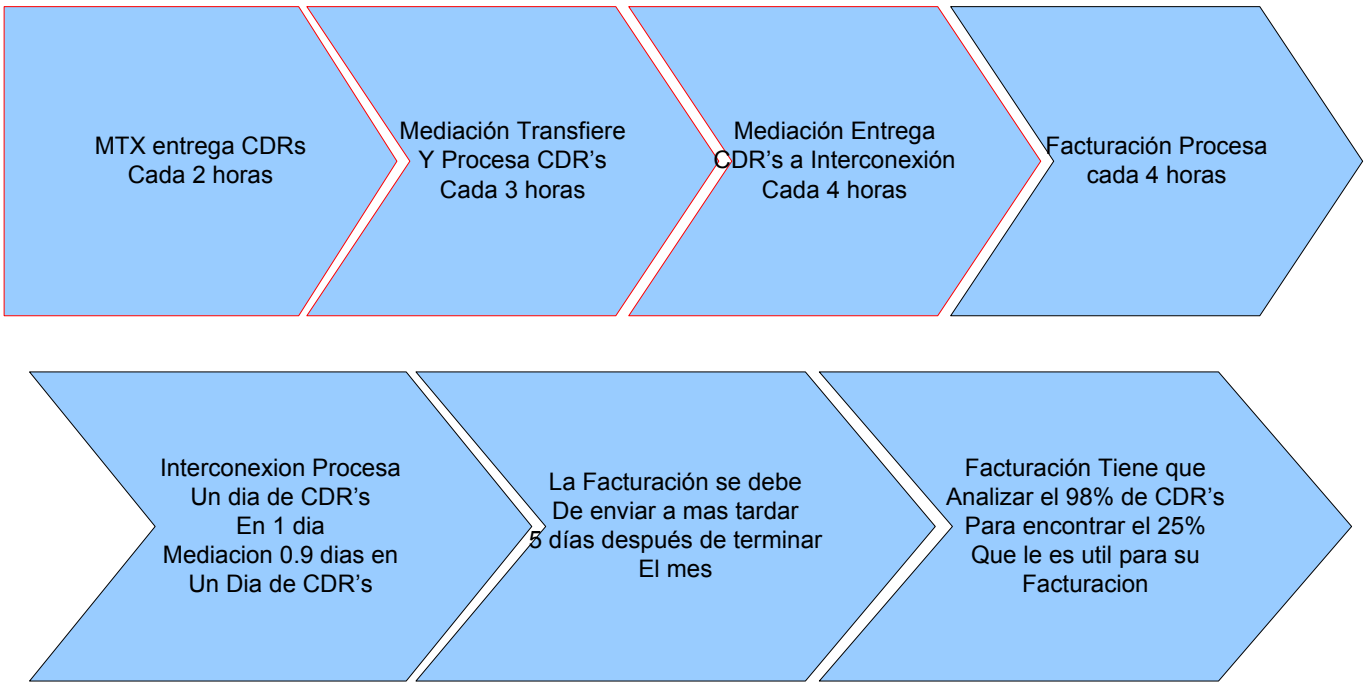

Figura 3.1.1 Proceso actual de transferencia de CDRs

En la figura 3.1.1 se muestra en un cronograma el procesamiento actual de CDRs. El Sistema de Mediación actualmente en operación no se ajusta a las políticas tecnológicas definidas por la compañía además de que se incrementará considerable la base de datos de clientes activos provocando una mayor demanda de procesamiento de CDRs y que el sistema de mediación quede obsoleto, puesto que de no tener esta capacidad de procesamiento se estarían limitando las posibilidades de crecimiento de la compañía.

Las implicaciones que tiene el proceso de transferencia así como el sistema de mediación actual se describen brevemente a continuación:

- Evitar la manipulación y posible alteración de los datos.

  Debido a la alta intervención humana en el proceso, este se presta a que los datos puedan sufrir alteraciones sea por error o intencionalmente, lo que nos deja con un sistema vulnerable desde el punto de vista operativo.
- El tiempo de recolección de la información es muy alto.

  En razón de que los protocolos de transporte no están unificados y de que el grado de automatización es pobre, el tiempo que se lleva la recolección de los datos es mucho más alto del deseable
- El período medio de pago es muy largo.

  Al no contar con una plataforma que nos permita contar con la información de manera oportuna, el periodo medio de pago se alarga porque en lugar de contabilizar las llamadas el mismo día que se realizan, a veces pasan semanas y para cuando el cliente recibe su facturación, tal vez ya no se le cobran todas las llamadas del periodo y si las que quedan pendientes se le cobran en el periodo siguiente, es mucho más probable que no las reconozca en su totalidad.
- No se obtienen los datos para tarifar NRT (Near Real Time). En particular, para la facturación a otros operadores.

Abundando el punto anterior, se ha hecho más que evidente con el sistema actual que la disponibilidad de los datos de facturación de manera casi inmediata es vital para la competitividad de la operadora, pues, otras operadoras competidoras ya lo hacen o están por hacerlo y esto les reporta beneficios visibles para el cliente y para la liquidez de la operadora, además de que al contar con la información al momento, es posible supervisar los procesos de manera oportuna.

- No se tiene un buen control de fraude de celulares.

Como actualmente la supervisión no puede hacerse día a día por la lentitud en el proceso y la vulnerabilidad de los datos es muy difícil detectar patrones de fraude en un corto lapso de tiempo para tomar acciones correctivas.

- No se consolidan en un formato común de CDR la información de las diversas plataformas que componen la red celular.

Debido a que la operadora cuenta con equipos de diferentes marcas y especificaciones en los diversos puntos de servicio final y a que estos equipos cuentan con diferentes formatos de CDR, no se cuenta con un formato único de CDR, por lo que al momento es necesario realizar laboriosas tareas semiautomáticas para dar proceso a los CDRs con un consumo excesivo de horas hombre.

- No se garantiza el correcto transporte y almacenamiento de los CDR's generados en las plataformas de conmutación.

Dado que el sistema es semiautomático en algunos de sus procesos, el transporte y almacenamiento de los datos se ve comprometido por la posibilidad de que existan errores u omisiones de parte del personal operativo; además de que las tareas de transporte y almacenamiento no son fácilmente rastreables. Lo anterior, se agrava si tomamos en consideración que contamos con al menos un punto de llegada por cada una de las centrales nacionales y que el personal está de alguna manera rebasado

- No se provee a los clientes del sistema de mediación con una mayor disponibilidad de información.

Como el sistema en su forma actual no genera registros adecuados de seguimiento de las tareas, no existen las interfaces de servicio adecuadas que provean a los clientes de mayor información respecto a los procesos y a sus resultados.

- No se facilita el acceso y procesamiento de la información colocándola en un formato estándar disponible en una base de datos.

Como no se cuenta con un formato estándar de CDRs, los registros de la base de datos tampoco son estándares del todo y deben contar con un dinamismo excesivo que entorpece el proceso y complica las tareas de seguimiento y documentación de la operación.

- El sistema actual se encuentra a su máxima capacidad de procesamiento.

La operadora está expandiendo su oferta de servicios a nivel nacional y está creciendo de manera significativa en su número de clientes y solicitudes de servicios por lo que el sistema en su forma actual se está quedando muy corto para atender a la creciente demanda.

3.2. Recolección y análisis de la información.

El Sistema de Mediación de la compañía actualmente obtiene los archivos AMA que contienen los CDRs generados en los MSC por medio de una conexión vía FTP replicándolos dentro de los servidores de mediación. Estos archivos cumplen con la nomenclatura establecida, lo que permite mantener un control de estos archivos.

Estos archivos pueden rotar ya sea cumpliendo con el tamaño permitido o con un tiempo de duración configurado.

En la figura 3.2.1 se muestra cómo está organizado el sistema de mediación actual.

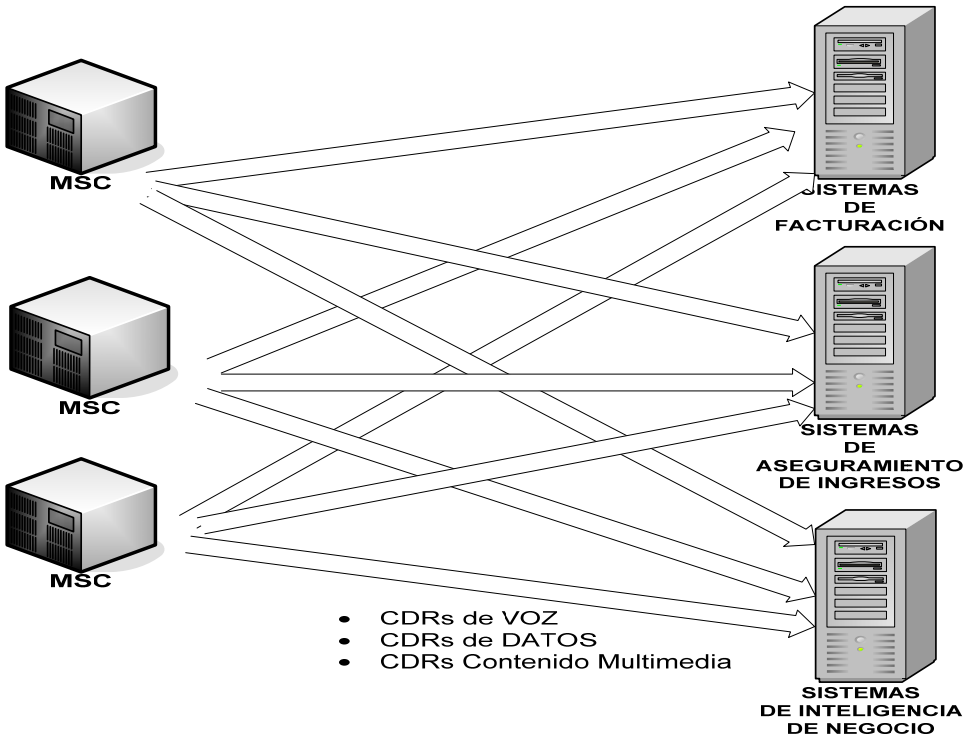

Figura 3.2.1 Sistema de Mediación actual

Este sistema realiza solamente la recolección, validación y distribución de CDR's, maneja solamente archivos y los entrega en su formato original (BCD) a los diferentes sistemas cliente.

Uno de los sistemas cliente de mayor importancia para la compañía es el sistema de facturación de interconexión, a continuación se describe como se realiza el análisis de la información de CDRs para esta facturación.

Principios y tipos de interconexión.

El campo de la interconexión de redes de telecomunicación es vasto e interdisciplinar, pues en él confluyen elementos económicos y regulatorios que determinan la viabilidad del servicio, además de los elementos propiamente

técnicos (topología, calidad, seguridad,..) dichos elementos han de configurarse en base a una serie de principios, que conforman la interconexión de redes de telecomunicación.

Sin duda, el primero de tales principios es el de la universalidad, es decir, que cualquier usuario pueda comunicarse con cualquier otro usuario, con independencia del operador de telecomunicaciones contratado por cada uno de ellos y con independencia, asimismo, de la ubicación geográfica de ambos. Otros principios, básicos, que conforman la interconexión de redes de telecomunicación son la privacidad que garantiza el secreto de las comunicaciones y la neutralidad que avala un tratamiento no-discriminatorio y transparente a los distintos operadores.

Se distinguen dos tipos de tarifas involucradas en la interconexión de redes: las tarifas de interconexión propiamente dichas, que amparan el coste (equipos de conmutación, medios de transmisión,..) de la interconexión de las redes, y las tarifas de interconexión por tráfico cursado (por llamadas telefónicas completadas) entre dichas redes. Estas últimas podrían definirse como el precio que un operador A debe abonar a un operador B por la utilización de la red de este último.

Interconexión de redes fijas.

Primeramente, se establece el plano topológico de la interconexión: el operador dominante u operador con poder significativo en el mercado ofrece a los demás operadores la posibilidad de acceder, al menos, a un punto de interconexión en cada provincia, punto que será elegido por el operador que solicita la interconexión.

Por otra parte, se detallan también los servicios a interconectar: el servicio telefónico básico: metropolitano, nacional, e internacional.

Tarifas de interconexión por tráfico cursado

Dichas tarifas se facturaban sobre tráfico efectivo cursado, midiendo la duración de cada llamada en minutos redondeados por exceso.

Bajo las anteriores premisas, se estipularon las tarifas de conexión, expresadas entonces en pesos/minuto, que tenían carácter de máximas (es decir, que podrían ser sustituidas por otras, inferiores, que acordaran los operadores de las redes a interconectar), para cada tipo de conexión: metropolitana, nacional e internacional.

Facturación entre operadores, y a los usuarios

Como premisa de partida, se establece que cada llamada individual deberá ser facturada al usuario por una única entidad habilitada; dicha entidad dependerá del tipo de llamada. Así, por ejemplo, en llamadas metropolitanas y nacionales de telefonía básica, se facturará únicamente al usuario origen de la llamada, lo cual hará el operador al cual tiene contratado el acceso dicho usuario. Y, por otra parte, dicho operador tendrá que abonar la correspondiente tasa de conexión al operador que sustenta el acceso del usuario destino de la llamada. Vea figura 3.2.1

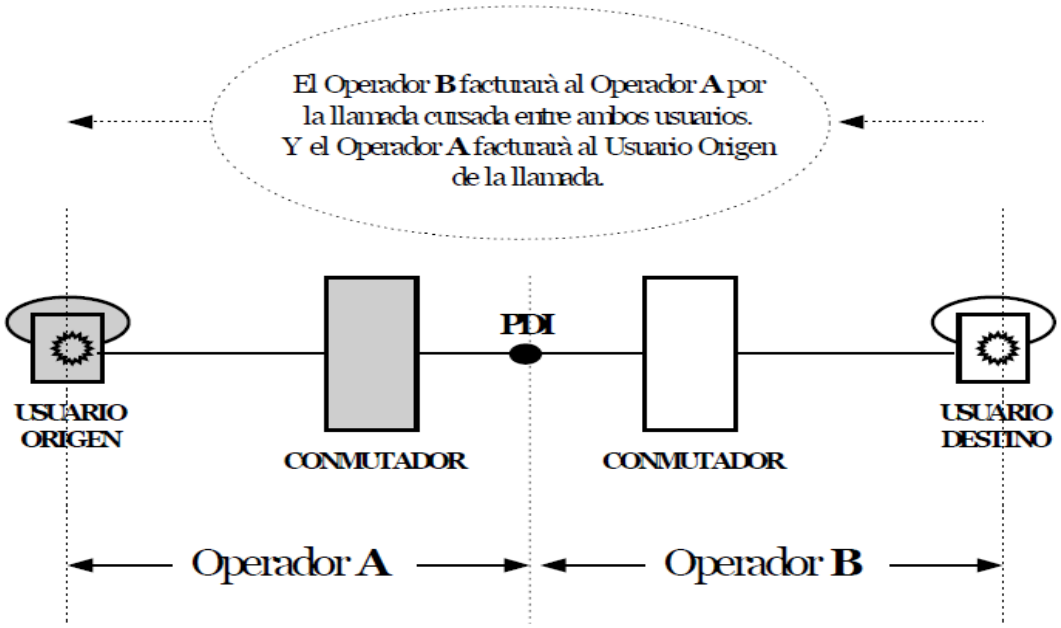

Figura 3.2.1 Facturación de llamadas caso de dos operadores

En caso de llamadas nacionales es aplicable el mismo procedimiento anterior. No obstante, en este caso cabe la posibilidad, además, de que el usuario origen de la llamada decida cursar ésta por la red de un operador distinto de aquel con el que tiene contratado el acceso. Vea figura 3.2.2

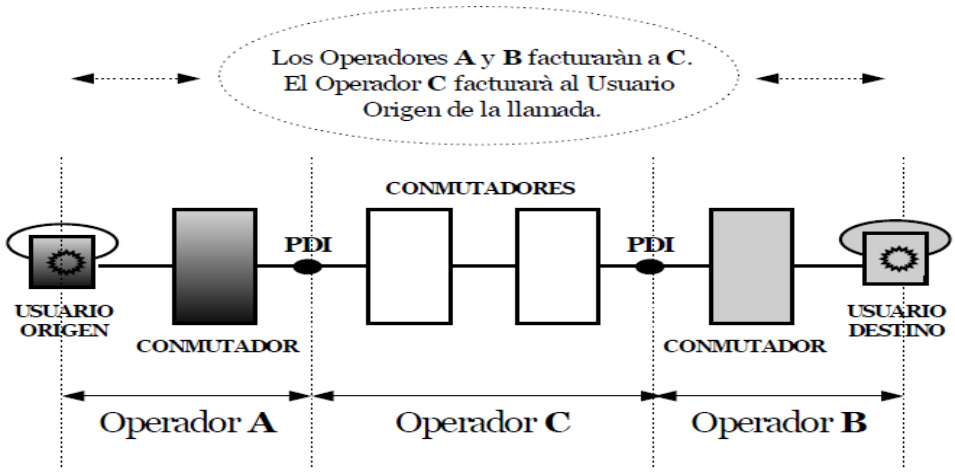

Figura 3.2.2 Facturación de llamadas caso de dos operadores nacional.

En el caso de llamadas internacionales, el operador seleccionado por el usuario origen de la llamada será el responsable de facturar a dicho usuario por la totalidad del servicio, en función de sus precios. Y, por otra parte, dicho operador tendrá que abonar la correspondiente tarifa de interconexión a la entidad con la cual tiene contratado el acceso el usuario en cuestión.

Conexión de redes de móviles.

Los concesionarios nacionales del servicio GSM deberán interconectarse entre sí, interconexión obligatoriamente de tipo directo e igualmente, dichos concesionarios podrán conectarse directamente con otras redes fijas o móviles, públicas o privadas, nacionales o extrajeras. En el primero de los casos se requerirá una propuesta conjunta de tarifas, y en el segundo, obviamente, éstas deberán ser concertadas entre los distintos operadores.

Tarifas de interconexión por tráfico cursado.

A título de referencia, a continuación se reflejan la tarifas, procediendo a señalar que las mismas sólo válidas. Vea figura 3.2.3

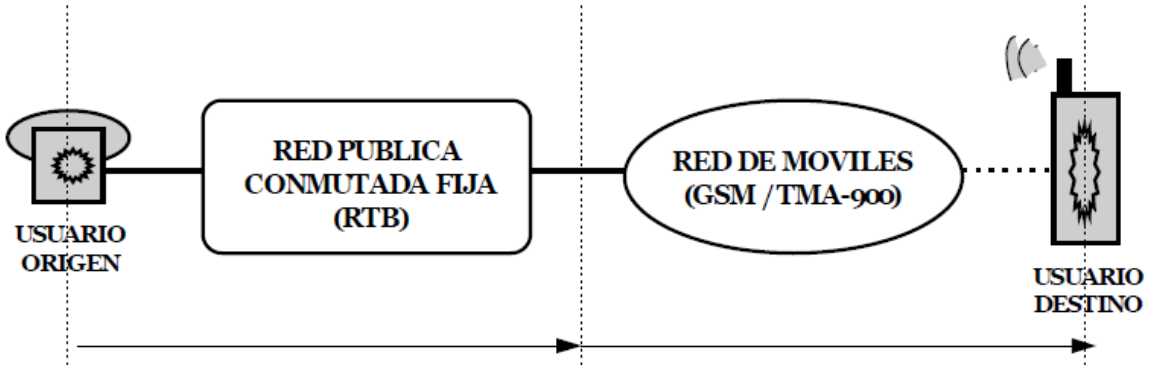

Figura 3.2.3 Llamada nacional, de fijo a móvil

(1) Tarifa (sin IVA) “normal”: de lunes a viernes, de 08.00 a 22.00 horas, y sábados, de 08.00 a 14.00 horas.

(2) Tarifa (sin IVA) “reducida”: de lunes a viernes (de 22.00 a 08.00 horas), sábados (a partir de las 14.00 horas), y domingos y festivos de ámbito nacional durante todo el día.

Las tarifas a aplicar a los abonados de la red de móviles se establecerán libremente por los operadores del servicio GSM con la limitación de que, en ningún caso, podrán ser inferiores a las establecidas para los abonados de la red fija. Vea figura 3.2.4

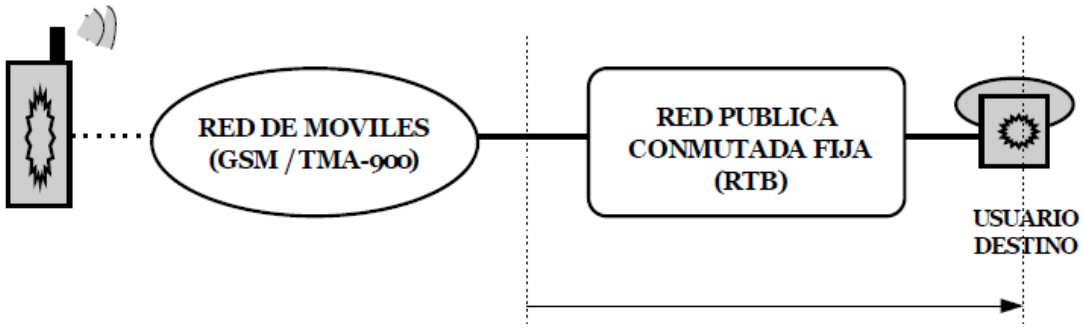

Figura 3.2.4 Llamada Nacional, de Móvil a Fijo.

(1) Tarifa (sin IVA) “normal”: de lunes a viernes, de 08.00 a 22.00 horas, y sábados, de 08.00 a 14.00 horas.

(2) Tarifa (sin IVA) “reducida”: de lunes a viernes (de 22.00 a 08.00 horas), sábados (a partir de las 14.00 horas), y domingos y festivos de ámbito nacional durante todo el día.

La facturación entre ambos operadores se realizará con la periodicidad y la forma en que ambos hayan acordado.

3.3. Requerimientos generales y particulares para el nuevo sistema.

Un requerimiento es un conjunto de propiedades o restricciones definidas con precisión, que un sistema de software debe satisfacer.

Los requerimientos son la pieza fundamental en un proyecto de desarrollo de software, ya que es la base para:

- Planear el proyecto y los recursos que se usarán en él.
- Especificar el tipo de pruebas que se habrán de realizar al sistema.
- Planear la estrategia de prueba a la que habrá de ser sometido el sistema.
- Son el fundamento del ciclo de vida del proyecto.

El Sistema de Mediación recopilará los archivos de voz, datos y de multimedia que sean enviados a través de la red para su tasación, facturación y estadística.
Debemos de puntualizar que el cliente no requiere de un front end porque la información se enviará para su procesamiento a diferentes sistemas de facturación o estadísticas internos, de esta forma sólo se mencionarán los requerimientos generales y particulares acerca de la funcionalidad del mismo y no sobre como el usuario interactuará con él.
Por lo anterior, a continuación se explicarán los requerimientos para el back end.

Requerimientos generales.

Los requerimientos generales para el SIMED son los siguientes:

- El sistema tendrá que unificar la información a partir de distintas fuentes de datos y distintas formas de comunicación con éstas.
- El sistema tendrá que complementar los CDRs, agregando la siguiente información a partir del CDR Original.
    - IRM, agregara el correspondiente IRM (International Roaming MIN) a los respectivos MIN (Mobile Identification Number) que aparezcan en el CDR.
    - Indicará si alguno de los números que intervienen en la llamada corresponden a operadores con convenios especiales con la compañía.
    - Indicará toda la información útil para las troncales y celdas que aparezcan en el CDR, indicará si es una troncal de interconexión, de LD, LDI. Etc.

  - Indicará si algunos de los números que intervienen en el CDR corresponden a los clientes corporativos de la compañía.
  - Indicará si algunos de los números que intervienen en la llamada es un cliente de pospago.
  - Indicará a que operador corresponden todos los números que intervienen en la llamada.
  - Indicará si algunos de los números que intervienen en la llamada corresponde a números de carriers internacionales
- Deberá tener la capacidad de procesar aproximadamente 4 millones de CDRs es decir, la cantidad de CDRs prevista para un día) en menos de medio día.
- Se requiere que tenga la capacidad de procesar un mes de CDRs en una semana o menos.
- En la base de datos se requiere tener almacenados tres meses de información de CDR de voz, aproximadamente 360 millones de CDRs.
- Se solicita una confiabilidad de un 99.9% en el procesamiento de CDRs.
- No debe haber intervención manual para la clasificación y homologación de los CDRs a procesar.
- Debe de implementar los protocolos dispuestos por los proveedores de las MSC para transferir CDRs por bloques y además mantener la transferencia de archivos para conciliación.
- Un CDR escrito por la MSC deberá encontrarse en la base de datos de CDRs, clasificado, homologado, enriquecido y distribuido a los diferentes sistemas cliente.
- Se requiere que el sistema sea de alta disponibilidad de al menos un 99.9% mensual.

Requerimientos particulares.

Los requerimientos particulares para este sistema son:

- Sistema de facturación a otros operadores (Interconexión)
    - Se deberán entregar solamente los CDRs, homologados, clasificados y enriquecidos que sean necesarios para la facturación de interconexión según los criterios que se definan, se estima que se entregará únicamente un 30% de la totalidad de CDRs al sistema de facturación y deberá de realizarse en tiempo real, es decir, prácticamente en el momento que se inserta en la base de datos del SIMED se deberá también de estar entregando a este sistema de facturación.
- Para el sistema de facturación a clientes el sistema deberá disponer de un mecanismo para enviar los CDRs con los criterios necesarios. Este mecanismo deberá ser fácilmente configurable y deberá disponer de mecanismos de reintentos y manejo de errores.
- Para los sistemas de aseguramientos de ingresos y de inteligencia de negocios se enviarán reportes diarios y mensuales, y se depositarán en los servidores especificados por cada aplicación, así como los formatos y contenido solicitado por cada uno.
- Para los sistemas de facturación a clientes de telefonía fija el sistema deberá disponer de un mecanismo para enviar los CDRs con los criterios necesarios hacia los sistemas de facturación de clientes de telefonía fija. Este mecanismo deberá ser fácilmente configurable y deberá disponer de mecanismos de reintentos y manejo de errores.
- Un proceso independiente conciliará los CDRs procesados por el protocolo a bloques y los archivos recolectados, para en su caso realizar los envíos necesarios al nuevo Sistema de Mediación SIMED.

3.4. Posibles módulos del nuevo sistema.

De acuerdo al análisis realizado para el SIMED se construirán los módulos representados en la figura 3.4.2 y que se describen a continuación:

- Módulo de recolección
- Módulo de parseo
- Módulo de catálogos
- Base de datos
- Módulo de explotación de base de datos

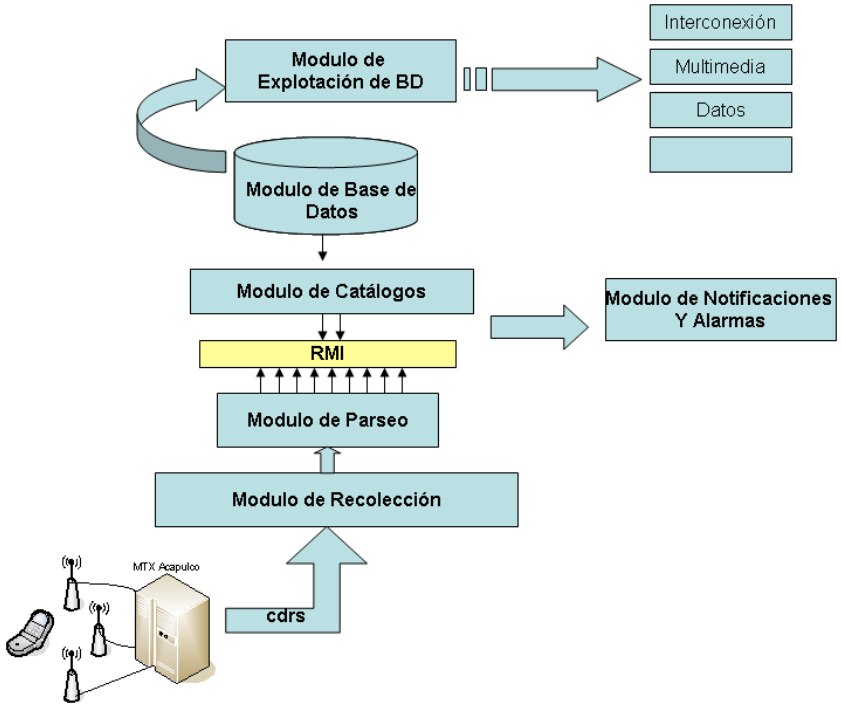

Figura 3.4.1. Módulos de la aplicación.

Módulo de recolección

Es el módulo de la aplicación encargado de establecer conectividad con las plataformas de conmutación celular y de implementar el protocolo de la central para recolección por bloques de CDRs en lugar de archivos.

El proveedor de la central de conmutación celular o MSC, dispuso de librerías de Java para programar su protocolo propietario, dentro de las responsabilidades de este módulo está la de garantizar el correcto transporte de los CDRs validando la consistencia de la información transferida así como su formación correcta.

Este módulo estará programado en Java de acuerdo con las políticas tecnológicas de la compañía.

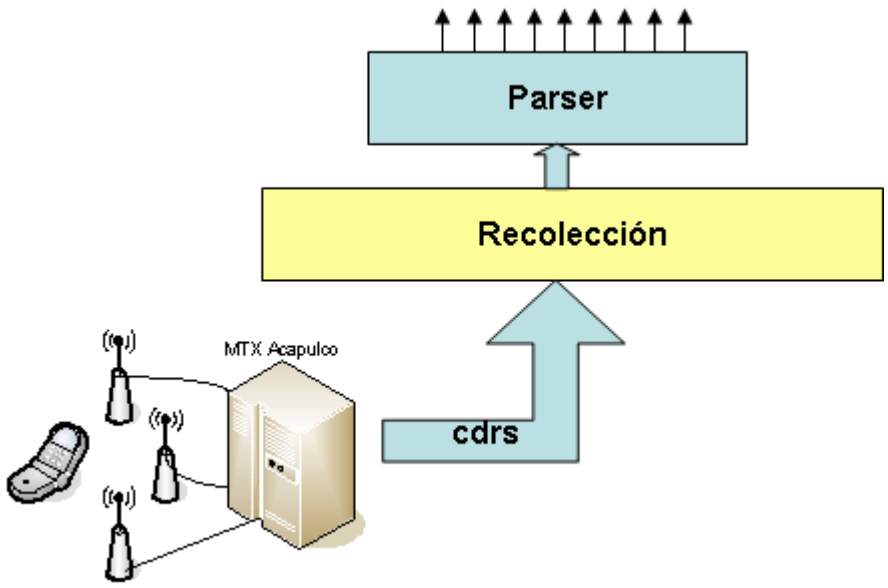

Figura 3.4.2. Módulo de recolección.

En la figura 3.4.2 se resalta el módulo de recolección dentro del Sistema de Mediación por bloques de CDRs y no por archivos. Como se observa el módulo de recolección establecerá una conexión dedicada con la MSC y se comunicará con esta mediante el protocolo propietario del proveedor de la MSC, se establecerá una sesión dedicada y estará enviando bloques de CDRs al módulo de parseo y a su vez los estará almacenando los bloques en archivos, manejando cifras de control.

Posteriormente trasferirá por FTP el archivo de CDRs ya cerrado por la MSC y lo comparará con el archivo escrito por el protocolo a bloques, notificando cualquier

anormalidad e incluso seleccionando los bloques faltantes y enviándolos al módulo de parseo para su reproceso.

Módulo de parseo

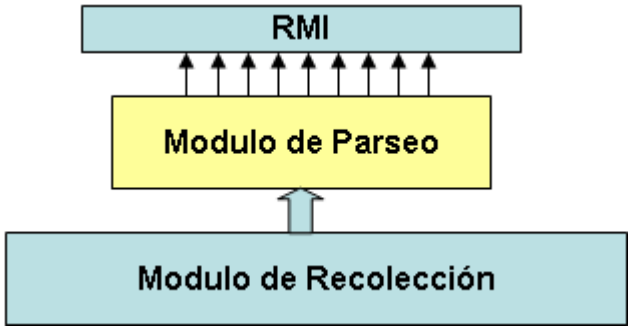

Figura 3.4.3. Módulo de parseo.

Como se observa en la figura 3.4.3 el módulo de Parseo es el que va a recibir los bloques de CDRs desde el módulo de recolección, será el encargado de leer cada CDR, clasificarlo, formatearlo y enriquecerlo con información de negocio, para su posterior inserción en la base de datos para su explotación y entrega manual y/o automática.

Este módulo de parseo necesitará información de catálogos de base de datos para complementar e enriquecer la información de cada CDR para ello implementará un cliente que consulte un servicio publicado por el módulo de catálogos mediante RMI.

En la figura 3.4.4 se muestra a alto nivel los submódulos que van a componer el módulo de parseo:

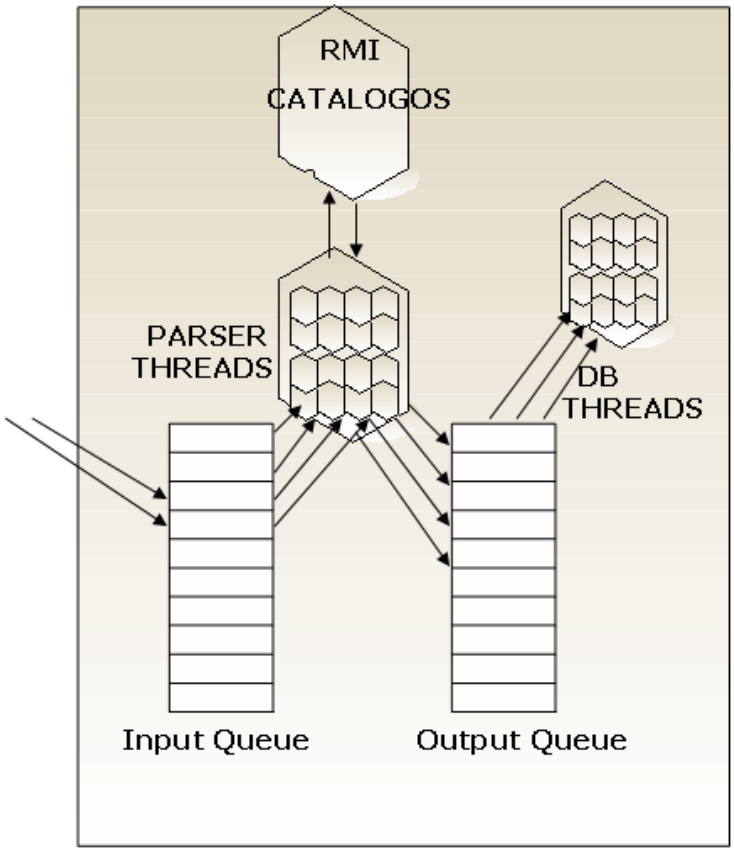

Figura 3.4.4. Submódulos que van a componer el módulo de parseo.

Este módulo también contendrá las reglas de negocio necesarias para enriquecer y complementar la información de cada CDR, utilizará el módulo de catálogos para consultar información respecto a numeración y respecto a troncales.

RMI

RMI (Java Remote Method Invocation) es una funcionalidad ofrecida por Java para compartir procesos o métodos de forma remota, ofreciendo un mecanismo simple para la comunicación de procesos en ambientes distribuidos basados en Java exclusivamente.

A través de RMI un programa Java puede exportar un objeto que estará disponible en la red y el programa permanece a la espera de peticiones en un puerto TCP, pudiéndose conectar un cliente e invocar los métodos publicados.

Se decidió hacerlo con RMI dado que las políticas definidas por la compañía obligaban a compartir funcionalidades a otras aplicaciones que pudieran utilizarlas con el propósito de reutilizar código agilizando los futuros desarrollos en Java para la compañía.

Módulo de catálogos

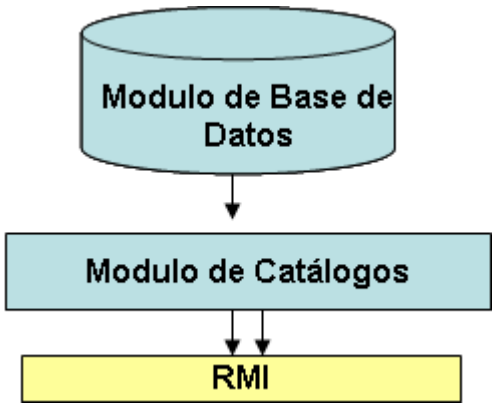

Figura 3.4.5. Módulo de catálogos.

En la figura 3.4.5 se muestra el módulo de catálogos, cuya finalidad será la de proporcionar al módulo de parseo y cualquier aplicación que lo requiera información de la base de datos. Publicará mediante RMI dos métodos, uno asociado a numeración y otro asociado a troncales o celdas, el módulo que consumirá estos métodos, tendrá que invocar estos colocando un número telefónico o bien un número de troncal y su respectiva MSC, con esto el módulo de catálogos entregará toda la información relacionada con ese número o troncal.

Módulo de base de datos y de explotación de BD.

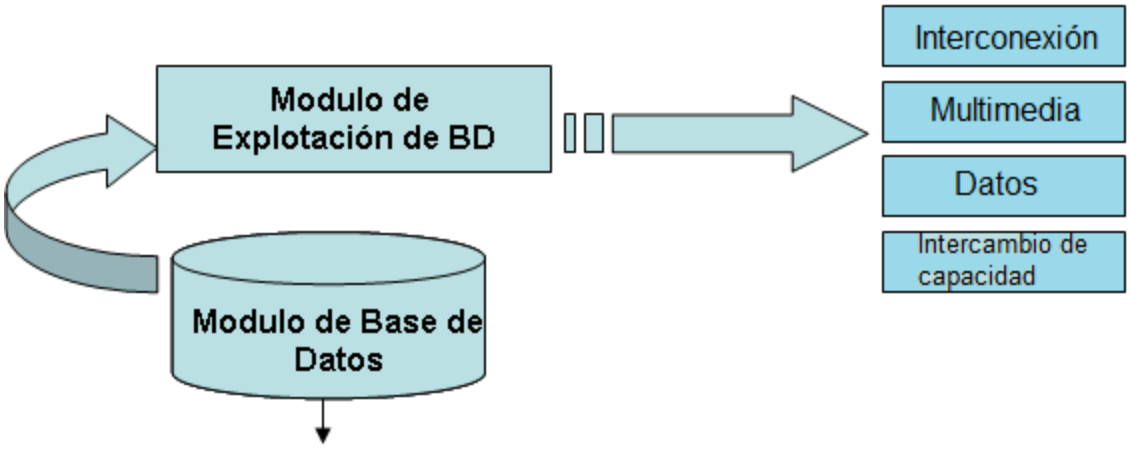

Figura 3.4.6. Módulo de base de datos y de explotación de BD.

La base de datos de la aplicación será donde se almacenen los catálogos necesarios para la clasificación de la información así como el repositorio donde se almacenarán los CDRs clasificados, homologados y enriquecidos, tendrá disparadores y Links hacia las BDs de otra aplicaciones que recibirán estos CDRs o el extracto que de ellos requieran, a su vez filtraran solamente la información necesaria para cada aplicación lo que permitirá entregar la información a las diferentes aplicaciones clientes en tiempo cercano al real.

En la figura 3.4.7 se muestra a un alto nivel como se organizará la Base de datos así como módulo de explotación de esta, uno de los requerimientos para la aplicación es que se tengan 3 meses de CDRs on line, es decir, aproximadamente 360 millones de CDRs almacenados y disponibles en la base de datos. Aprovechando la funcionalidad del manejador se particionarán las tablas de CDRs físicamente, siendo las llaves de partición los campos de mes y de día. Adicionalmente se agregó un campo numérico que almacena la hora lo que permite realizar búsquedas por mes, día y hora.

El diseño de la base incluirá 1440 particiones repartidas en 360 tablespaces, además para la explotación de la base se desarrollaran paquetes (Packages) así como triggers escritos en PL/SQL que estarán entregando, clasificando y transformando la información necesaria de CDRs para cada aplicación cliente que así lo requiera.

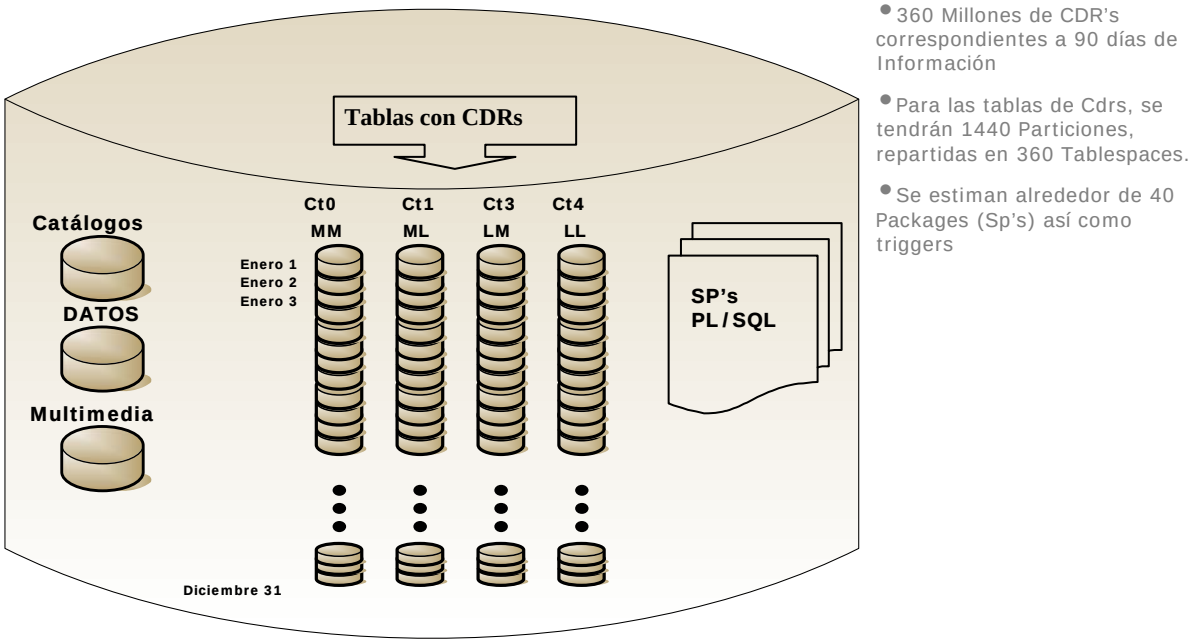

Figura 3.4.7. Organización de la base de datos

3.5. Justificación de la metodología y del software a utilizar.

Para la realización de este proyecto, se nos han especificado la tecnología y la metodología a utilizar para la construcción del SIMED. Estos estándares son determinados por las áreas de tecnología de la compañía cuya función es alinear los desarrollos así como definir la política tecnológica que se seguirá a futuro.

Dada la importancia del desempeño de la nueva aplicación, haremos un breve comparativo entre algunas herramientas existentes en el mercado y que se piensa podrían dar buenos resultados para la aplicación sin que esto signifique que podamos modificar las herramientas a utilizar ya definidas, puesto que uno de los

requerimientos es que el sistema se encuentre alineado a la política tecnológica de la compañía, a continuación se compararan los siguientes RDBMS:

- Informix.
- Progress.
- Oracle 10g.

INFORMIX

Es creado por Informix Software Inc., en el año de 1980, después fue adquirida por IBM en 2001. Informix es un lenguaje de cuarta generación, es considerado como uno de los manejadores de datos más flexibles y potentes. Posee como base para el desarrollo de aplicaciones, el manejo de bases de datos relacionales. Algunas de las características que posee son:

- Utiliza plataforma NT y UNIX.
- Se especializa en aplicaciones tipo GIS (datos geográficos).
- Dispone de herramientas gráficas.
- Gestiona múltiples bases de datos remotas de una consola centralizada.
- Capacidad de relación de datos en múltiples lugares físicos.
- Conecta datos relacionales en páginas web.
- Ocupa menos memoria y recursos que Oracle.
- Se integra con Linux y Oracle.
- Ofrece herramientas para crear menús, formularios de entrada de datos y generadores de listados.
- Soporte para Datawarehouse y Datamining.

PROGRESS

Es un potente sistema de gestión de base de datos relacional, de la compañía de software, llamada "Progress Software Corporation" (o PSC). Progress tiene un

lenguaje de programación 4GL con todas las funciones para trabajar con los datos y la programación lógica. El mismo lenguaje se utiliza para la programación de la interfaz de usuario (GUI y terminal). Se alcanza un mejor rendimiento cuando se trabaja con los datos, en comparación con SQL, ya que la manipulación de datos se hace en un nivel inferior, y con más comandos sin problemas de base de datos de acceso.

Algunas de las características que posee son:

- Solución altamente escalable que proporciona un rendimiento óptimo.
- Hasta 10,000 usuarios concurrentes.
- Soporte para SMP.
- Servidor multienlazado; múltiples servidores por cada base de datos (hasta 256).
- Soporte para múltiples sistemas operativos y plataformas de hardware.
- Conformidad con el ANSI SQL-92.
- Características de seguridad, incluyendo recuperación automática de caídas y transacciones y capacidades de avance "roll-forward".
- Respaldo en línea, soporte a prueba de fallas y reorganización de tablas e índices para alta disponibilidad
- Almacenamiento virtual de datos ilimitado, su única limitante será el hardware.

En la tabla 3.5.1 se muestran las comparaciones de los diferentes RDBMS.

Característica	Informix	Progress	Oracle
Variedad de plataforma en las que opera (Multiplataforma)	Excelente	Excelente	Excelente
Robustez	Excelente	Excelente	Excelente
Facilidad de uso	Bueno	Excelente	Excelente
Compartir los datos con otras bases de datos	Bueno	Bueno	Excelente
Disponibilidad de herramientas	Bueno	Excelente	Excelente
Restauración del sistema	Bueno	Bueno	Excelente
Capacidad de herramientas de información	Bueno	Bueno	Excelente
Cantidad de clientes que pueden atender (concurrencia)	Bueno	Bueno	Excelente
Niveles de seguridad	Bueno	Excelente	Excelente
Costo	Bueno	Bueno	Bueno

Tabla 3.5.1. Comparativa de manejadores de bases de datos.

Algunas de las características que destacan del RDBMS Oracle a utilizar son:

- Facilidad para el uso de la herramienta.
- Multiplataforma.
- Rapidez de la aplicación.
- Concurrencia.
- Seguridad.

Como en el caso de los RDBMS también se compararán algunos lenguajes de programación, asumiendo de igual manera que las políticas de la compañía ya nos indicaron el lenguaje a utilizar, los lenguajes a comparar son los siguientes:

- PHP
- ASP
- Java

PHP (Hipertext Preprocessor)

Fue creado en 1994 por Rasmus Lerdorf. Es un lenguaje de programación interpretado de alto nivel que puede ser incluido en las páginas HTML, así como ser ejecutado por un servidor web.
Algunas de las características que posee son:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).

- El programador puede aplicar cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable.

ASP (Active Server Pages)

Es una tecnología creada por Microsoft para la creación dinámica de páginas web ofrecida junto a su servidor IIS (Internet Information Services***).*** Apareció por primera vez en diciembre de 1996. No se trata de un lenguaje de programación sino de un marco en donde construir aplicaciones basadas en internet. La tecnología ASP se emplea principalmente para crear aplicaciones interactivas que funcionan en internet.

Algunas de las características que posee son:

- Es totalmente gratuito para Microsoft Windows NT o Windows 95/98.
- El código ASP se puede mezclar con el código HTML en la misma página.
- El código ASP se puede escribir con un simple editor de textos como el bloc de notas de Windows o UltraEdit.
- Cómo el código ASP se ejecuta en el servidor, y produce como salida código HTML puro, su resultado es entendible por todos los navegadores existentes.
- Mediante ASP se pueden manipular bases de datos (consultas, actualizaciones, borrados, etc.) de prácticamente cualquier plataforma, con tal de que proporcione un driver OLEDB u ODBC.
- Permite usar componentes escritos en otros lenguajes (C++, Visual Basic, Delphi), que se pueden llamar desde los guiones ASP.
- Sin modificar la instalación, los guiones ASP se pueden programar en JScript o VBScript, pero también existen otros lenguajes, como Perlscript y Rexx, que se pueden emplear para programar ASP.
- Se ha portado a la plataforma Java por Chili!Soft y Halcyon Software, lo que permite que ASP sea usado en casi cualquier sistema operativo.

- Permite acceder a bases de datos de una forma sencilla y rápida.
- Las páginas se generan dinámicamente mediante el código de scripts, (guiones).
- El código de script se ejecuta en el servidor, y no se depende del navegador que se emplee.
- Desde una página ASP se pueden ejecutar servidores OLE en el servidor de web, lo que abre un abanico de nuevas posibilidades sólo accesibles previamente usando CGI y filtros ISAPI: acceso a base de datos, acceso a ficheros, logging en el sistema, envío de correo, etc.

En la tabla 3.5.2 se muestran las comparaciones de las diferentes herramientas de desarrollo.

Característica	**PHP**	**ASP**	**Java**
Facilidad para manejo de interfaz de usuario	Bueno	Bueno	Bueno
Rapidez	Bueno	Bueno	Excelente
Componentes Active X	Excelente	Excelente	Bueno
Manejo de bases de datos	Bueno	Bueno	Excelente
Herramientas para el desarrollo de servicios web	Bueno	Excelente	Excelente
Elaboración de pantallas	Bueno	Bueno	Excelente
Costo	Bueno	Bueno	Bueno
Desempeño	Bueno	Bueno	Excelente
Facilidad de aprendizaje	Bueno	Bueno	Bueno
Documentación	Excelente	Bueno	Excelente

Tabla 3.5.2. Comparativa de herramientas de desarrollo.

Al tener en cuenta las características de las diferentes herramientas de desarrollo, Java definitivamente cumple con los requisitos necesarios para el proyecto.
Algunas de las características que destacan de Java son las siguientes:

- Es un lenguaje independiente de la plataforma, ya que puede funcionar en cualquier ordenador.
- Posee gran variedad de documentación en internet,
- No requiere licencia para poder utilizarlo.
- Facilidad de manejo de interfaz de usuario.
- Desempeño.

CAPÍTULO

4

DISEÑO Y CONSTRUCCIÓN DE LA APLICACIÓN

4.1 Arquitectura de la aplicación.

Modelo cliente/servidor.

El modelo cliente/servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, solicitan requerimientos a uno o más servidores centrales, como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma.

En el modelo usual, un servidor (daemon, se traduce como “demonio”), se activa y espera las solicitudes de los clientes. En este modelo se intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones.

En este modelo, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio), figura 4.1.1, En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de clientes para otras.

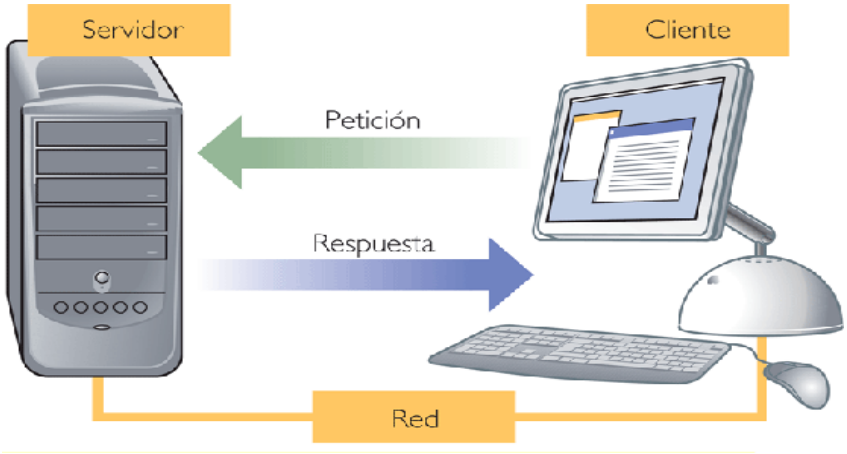

Figura 4.1.1 Modelo cliente/servidor

De aquí se deduce que el proceso cliente es quien inicia el diálogo, el proceso servidor es quien pasivamente espera a que lleguen peticiones de servicio y el middleware corresponde a la interfaz que provee la conectividad entre el cliente y el servicio para poder intercambiar mensajes.

Elementos Principales.

- Cliente

  Es el proceso que permite al usuario formular requerimientos y pasarlos al servidor, se le conoce con el nombre de front-end.

  Es el que maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de la red. Sus funciones son:

  - Administrar la interfaz de usuario.
  - Interactuar con el usuario.
  - Procesar la lógica de la aplicación y hacer validaciones locales.
  - Generar requerimientos de base de datos.
  - Recibir resultados del servidor.
  - Formatear resultados.

- Servidor

  Es todo proceso que proporciona un servicio a otros. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Sus principales funciones son:

  - Aceptar los requerimientos de bases de datos que hacen los clientes.
  - Procesar los requerimientos de bases de datos.
  - Formatear datos para transmitirlos a los clientes.
  - Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

- Middleware

  Es un módulo intermedio que actúa como conductor entre sistemas permitiendo a cualquier usuario de sistemas de información comunicarse con varias fuentes de información que se encuentran conectadas por una red, siendo así el intermediario entre el cliente y el servidor y se ejecuta en ambas partes.

  Sus principales características son:

  - Simplifica el proceso de desarrollo de aplicaciones al independizar los entornos propietarios.
  - Permite la interconectividad de los sistemas de información del organismo.
  - Proporciona mayor control del negocio al poder contar con información procedente de distintas plataformas sobre el mismo soporte.
  - Facilita el desarrollo de sistemas complejos con diferentes tecnologías y arquitecturas.

Características de la arquitectura cliente/servidor.

Las características básicas de una arquitectura cliente/servidor son:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de “servicio”, que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicio a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente/servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

Ventajas de la arquitectura cliente/servidor.

Entre las principales ventajas del sistema cliente/servidor están:

- La existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes lo cual contribuye a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.
- Facilita la integración entre sistemas diferentes y comparte información permitiendo, que las máquinas ya existentes puedan ser utilizadas, pero utilizando interfaces más amigables al usuario. De esta manera podemos integrar PC´s con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.
- Al favorecer el uso de interfaces graficas interactivas, los sistemas construidos bajo este esquema tienen mayor interacción con el usuario. En el uso de interfaces gráficas, no es siempre necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.
- Es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (servidores de SQL).

- Facilita la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- Proporciona a los diferentes departamentos de una organización, soluciones locales pero permitiendo la integración de la información relevante a nivel global.

Desventajas de la arquitectura cliente/servidor.

Entre las principales desventajas están:

- El mantenimiento de los sistemas es más difícil pues implica la interacción de diferentes partes del hardware y del software, distribuidas por diferentes proveedores, lo cual dificulta el diagnostico de fallas.
- Se cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.
- Es importante que los clientes y servidores utilicen el mismo mecanismo, lo cual implica que deben tener mecanismos generales que existan en diferentes plataformas.
- Hay que tener estrategias para el manejo de errores y para mantener la consistencia de los datos.

Arquitectura cliente/servidor de dos capas.

Consiste en una capa de presentación y lógica de la aplicación; y la otra de la base de datos figura 4.1.2. Normalmente esta arquitectura se utiliza en las siguientes situaciones:

- Cuando se requiera poco procesamiento de datos en la organización.
- Cuando se tiene una base de datos centralizada en un solo servidor.
- Cuando la base de datos es relativamente estática.
- Cuando se requiere un mantenimiento mínimo.

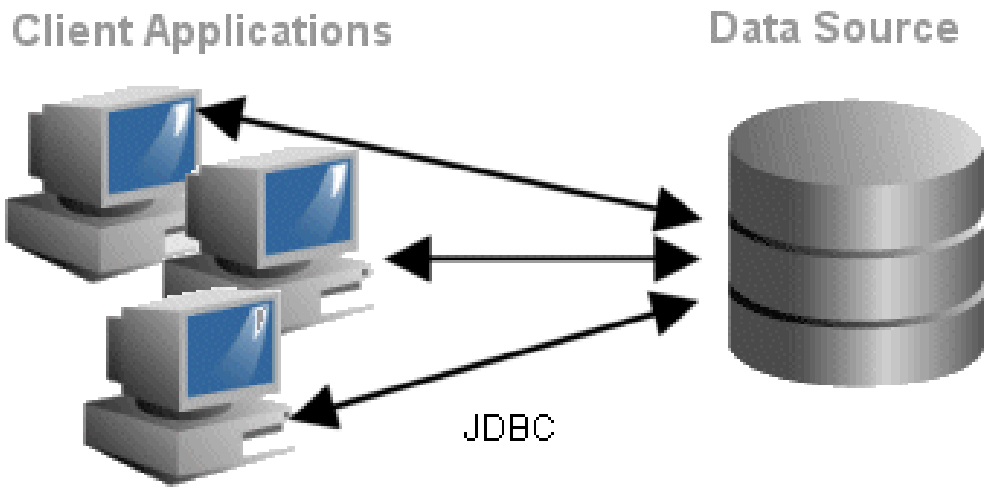

Figura 4.1.2 Cliente/servidor de dos capas.

UML (Lenguaje unificado de modelado).

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un leguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, también aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los elementos o componentes en el sistema y para documentar y construir. Es el lenguaje en el que esta descrito el modelo, figura 4.1.3.

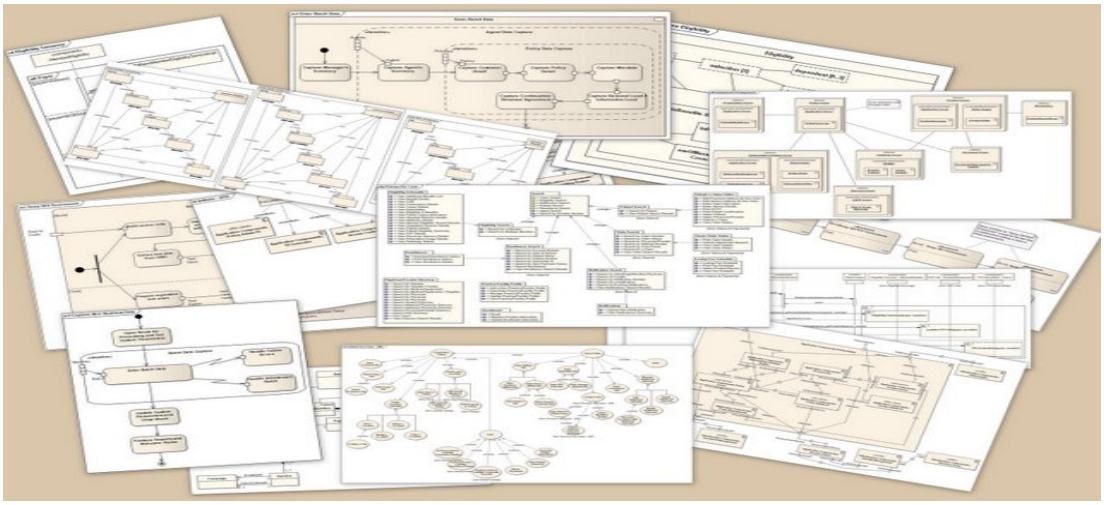

Figura 4.1.3 Diagramas UML.

Se puede aplicar en el desarrollo de software, entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica qué metodología o proceso usar.

Los diagramas a representar dependerán del sistema a desarrollar. Está recomendaciones se deberán adaptar a las características de cada desarrollo.

En UML hay trece tipos de diagramas diferentes, los cuales son:

- Los diagramas de estructura: Enfatizan en los elementos que deben existir en el sistema modelado.
    - Diagrama de clases.
    - Diagrama de componentes.
    - Diagrama de objetos.
    - Diagrama de estructura compuesta.
    - Diagrama de despliegue.
    - Diagrama de paquetes.
- Los diagramas de comportamiento: Enfatizan en lo que debe suceder en el sistema modelado.
    - Diagrama de actividades.
    - Diagrama de casos de uso.
    - Diagrama de estados.

- Los diagramas de interacción: Son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:
    - Diagrama de secuencia.
    - Diagrama de comunicación.
    - Diagrama de tiempos.
    - Diagrama global de interacciones o diagrama de vista de interacción.

Diagrama de casos de uso.

Un caso de uso es la descripción de la secuencia de interacciones que se realizan entre los actores y el sistema. Presenta las siguientes características:

- Es siempre iniciado por un actor y provee de valores a un actor.
- Es una tarea específica que se realiza tras una orden de algún agente externo, ya sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.
- El nombre del caso de uso siempre esta expresado desde el punto de vista del actor y no desde el punto de vista del sistema.
- Cada caso de uso está acotado al uso de una determinada funcionalidad en el sistema.

Sus elementos (figura 4.1.4), son:

- Actores: Es algo con comportamiento, como una persona, un sistema informático u organización y que realiza algún tipo de interacción con el sistema. Se representa mediante una figura humana.
- Casos de Uso: Es una descripción de la secuencia de interacciones que se producen entre un actor y un sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad y se representa mediante una elipse con el nombre del caso de uso en su interior.

- Inclusión (include o use): Es una forma de interacción o creación, un caso de uso dado puede “incluir” otro. El primer caso de uso a menudo depende del resultado del caso de uso incluido. Esto es útil para extraer comportamientos verdaderamente comunes desde múltiples casos de uso a una descripción individual. Se puede decir que va desde padre a hijo.
- Extensión (Extend): Es otra forma de interacción, un caso de uso dado, puede extender a otro. Esta relación indica que el comportamiento del caso de la extensión se utiliza en el caso de uso. Un caso de uso a otro caso siempre debe tener extensión o inclusión.
- Generalización: Es la actividad de identificar elementos en común entre conceptos y definir las relaciones de una superclases y subclase. Es una relación de generalización/especialización.

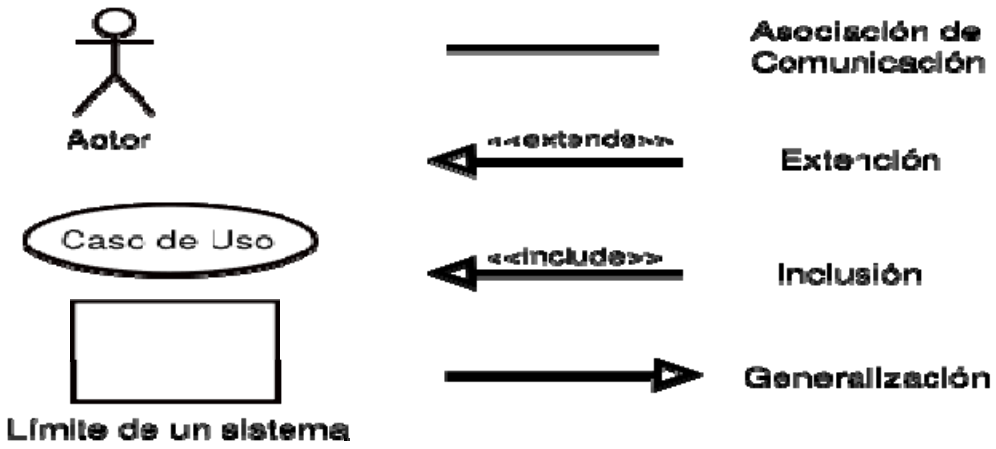

Figura 4.1.4 Componentes de un diagrama de caso de uso.

Justificaciones.

El Sistema de Mediación como ya se ha dicho, utiliza los CDRs que se extraen de las centrales telefónicas, los cuales entran en una serie de procesos que no son tangibles, ni visibles para el usuario. Simplemente es una aplicación que tiene como objetivo procesar los CDRs, estandarizarlos y guardarlos en una base de

datos, sin que el usuario intervenga, además de que no arroja resultados que el usuario pueda manipular, ya que los CDRs procesados solamente se guardan en una base de datos para después ser utilizados por otros sistemas, para fines propios.

Es por esto que se utilizará la arquitectura cliente/servidor de dos capas simplemente porque el sistema no tiene interacción con el usuario de ningún tipo. Se podría decir que el sistema es automático, ya que el usuario en ningún momento introduce datos ni peticiones, en este sistema el cliente no existe, solo existe comunicación entre el servidor y el sistema, alimentados de las centrales y aterrizados en una base de datos.

Para modelar el sistema de mediación, se utilizará UML, con el diagrama de casos de uso, ya que este diagrama nos permitirá visualizar un entorno general de cómo interactúan los actores del sistema, de esta forma, se puede conocer cómo se comporta cierta parte del sistema, mas no, como está implantada la estructura que lo define. Indica lo que cierta parte del sistema debe de hacer cuando ocurra cierto proceso. También permitirá centrarse en lo que se espera lograr al utilizar el sistema así como una estimación más exacta para determinar tiempo, recursos y prioridades en la dosificación de esfuerzo de desarrollo y un mayor control para mantener las sucesivas revisiones de los programas.

4.2 Diagramación

4.2.1 Diagrama de casos de uso.

Estos son los casos de uso más relevantes:

- Recepción de CDRs
- Análisis de CDRs
- Balanceador de tramas.

- Parser SIMED.
- Monitoreo de disponibilidad.
- Procesamiento facturación a clientes.
- Generar información para facturación a clientes corporativos.
- Generar información para facturación a otros operadores.
- Carga información externa.
- Diagrama General.

Recepción de CDRs.

La conexión con las MSC es necesaria para recolectar los CDRs que se generan en el momento de realizar las llamadas entre equipos celulares o teléfonos fijos, es importante obtener la información mediante el protocolo que permite la transferencia por bloques, así como la recolección de los archivos mediante FTP Ver figura 4.2.1

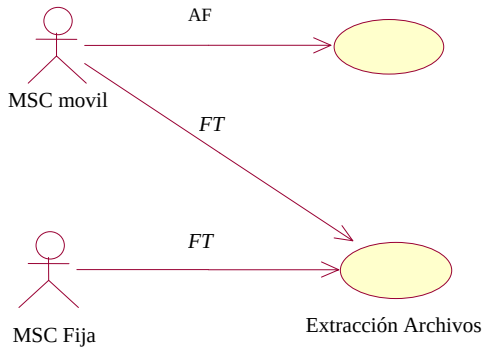

Figura 4.2.1 Diagrama de recepción de CDRs

Análisis de CDRs.

Es necesario identificar el tipo de CDR, para en su caso poder informar a otras compañías de telefonía sobre algún CDR de un equipo que utiliza nuestra red. Se verifica que cumpla con el requerimiento de la versión que establece el MSC. En la figura 4.2.2 se muestra el diagrama de caso de uso para el análisis de CDRs.

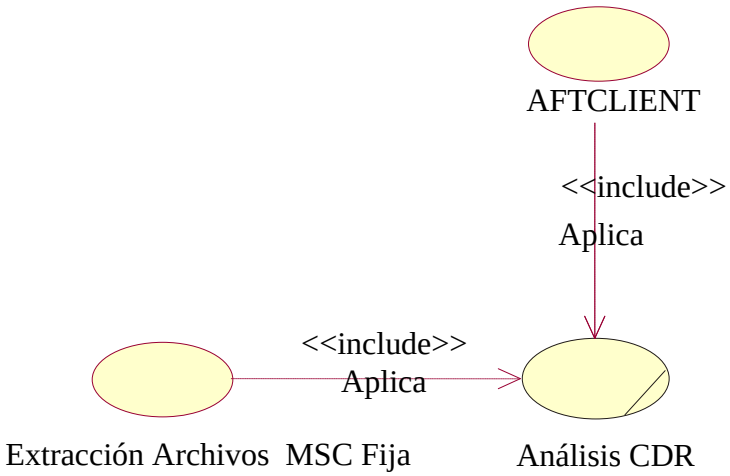

Figura 4.2.2 Diagrama de análisis de CDRs

Balanceador de tramas.

Esta herramienta funciona manteniendo la conexión con los diferentes servidores y también detecta automáticamente cuando un socket destino se encuentra sin poder recibir transacciones, distribuyendo los CDRs a los otros destinos disponibles permitiendo que se mantenga el servicio. La figura 4.2.3 muestra el diagrama de caso de uso para el balanceador de tramas.

Figura 4.2.3 Diagrama de balanceador de tramas

Parser SIMED.

El parser SIMED compone el proceso medular del Sistema de Mediación, debido a que realiza las operaciones de analizar el CDR que se obtuvo de los diferentes medios de conmutación. El parser SIMED analiza los CDRs agregando información adicional al CDR permitiendo clasificar la trama para su inserción en la base de datos. Véase la figura 4.2.4 con el diagrama de parser SIMED.

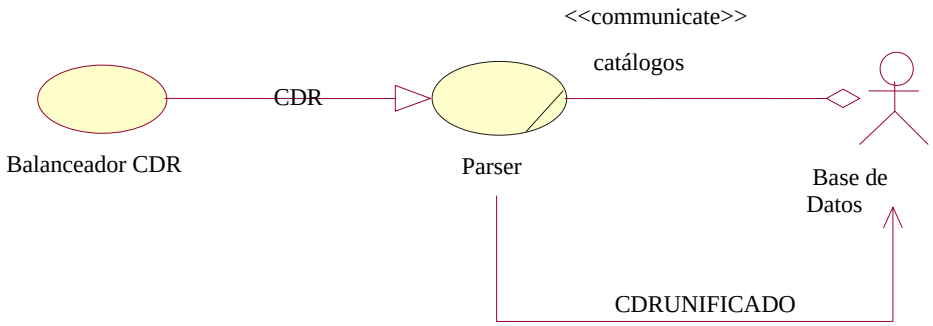

Figura 4.2.4 Diagrama de parser SIMED

Monitoreo de disponibilidad.

Existen diversas herramientas propias que permiten realizar verificaciones constantes con el fin de mantener la disponibilidad de los diversos objetos que intervienen en el flujo del CDR, desde su llegada al AFTCLIENT hasta su impacto a la base de datos. La figura 4.2.5 muestra el diagrama de caso uso de este caso.

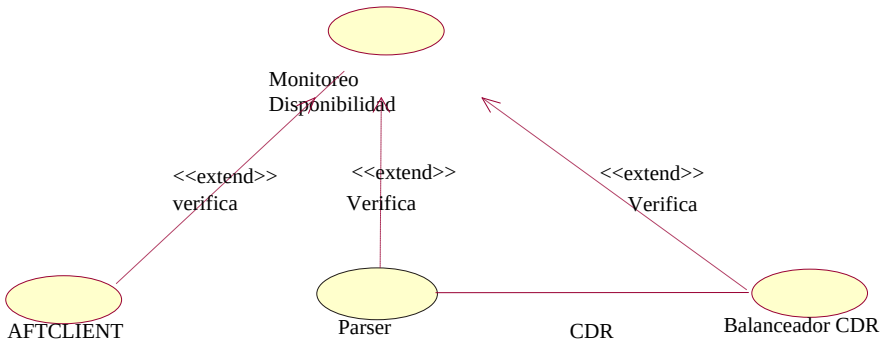

Figura 4.2.5 Diagrama de monitoreo de disponibilidad

Procesamiento facturación a clientes.

La facturación a clientes tiene un formato especificado por otras compañías telefónicas derivado del Intercambio de capacidad y de la interconexión, para efecto de conciliación y facturación, es necesario traducir este formato, a un formato propio y en una base de datos. En la figura 4.2.6 se muestra su diagrama de caso de uso.

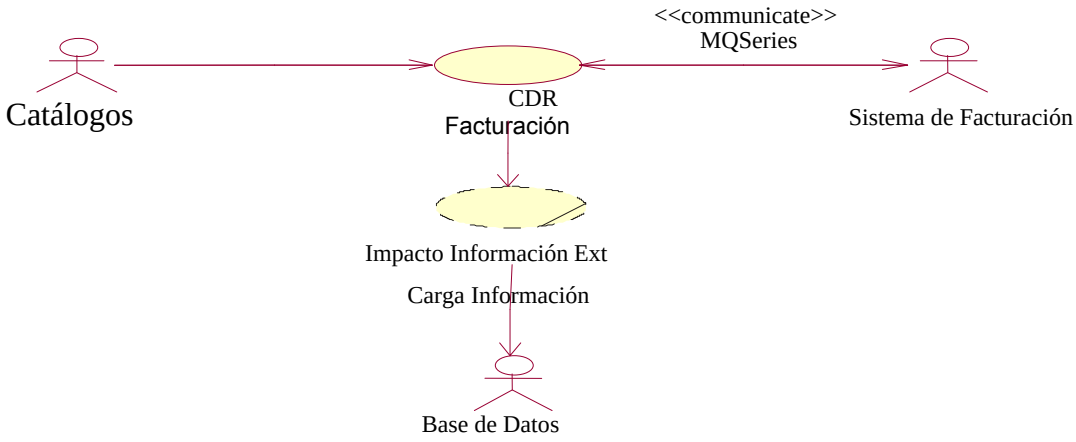

Figura 4.2.6 Diagrama de procesamiento facturación a clientes

Generar Información para facturación a clientes corporativos

Existen registros de CDRs los cuales no son informados a la caja de prepago y es necesario que se le reporten estas llamadas. La información que es necesario impactar de manera mensual es para aquellas series que correspondan al servicio de telefonía fija o post pago móvil. Ver figura 4.2.7 con el diagrama de caso de uso.

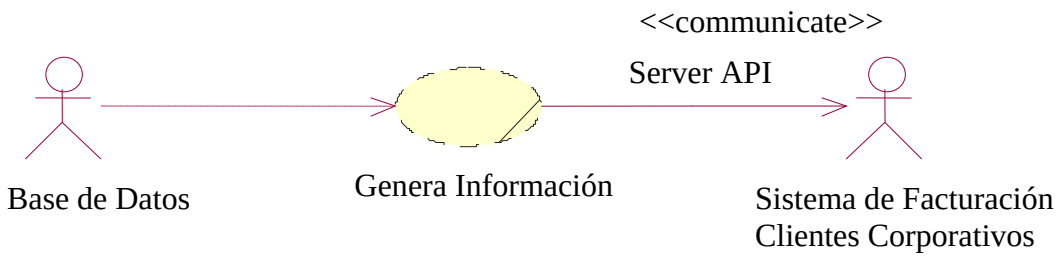

Figura 4.2.7 Diagrama de información facturación a clientes corporativos

Generar información para facturación a otros operadores

Con el fin de cumplir con los requerimientos para la interconexión e intercambio de tecnología cubriendo el servicio de red celular, es necesario que la información que se genera de los diferentes mecanismos de conmutación de servicios, sea de utilidad para la generación de facturas las cuales se utilizan para el cobro o conciliación con otros operadores. La figura 4.2.8 muestra el diagrama correspondiente.

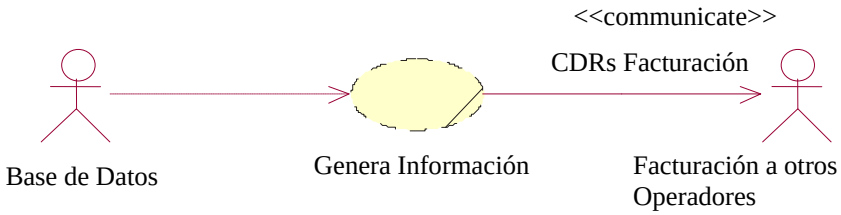

Figura 4.2.8 Diagrama de facturación a otros operadores

Carga de información externa

Con el objetivo de cumplir los requerimientos de información para los distintos usuarios que tiene el Sistema de Mediación es necesario obtener esta información de diferentes plataformas como puede ser el proceso de facturación de clientes con otras compañías telefónicas o las plataformas de envío de datos o multimedia, esto puede seguir creciendo dependiendo de los diferentes equipos que soportan

los nuevos productos que se ofrecen. La figura 4.2.9 muestra el diagrama de carga de información externa.

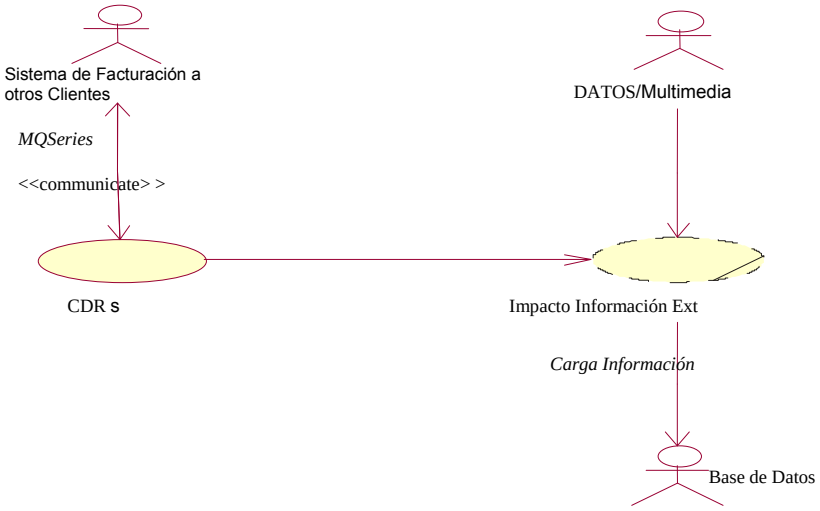

Figura 4.2.9 Diagrama de carga de información externa

Diagrama general.

A continuación la figura 4.2.10 muestra el diagrama unificado con todos los casos de uso mencionados.

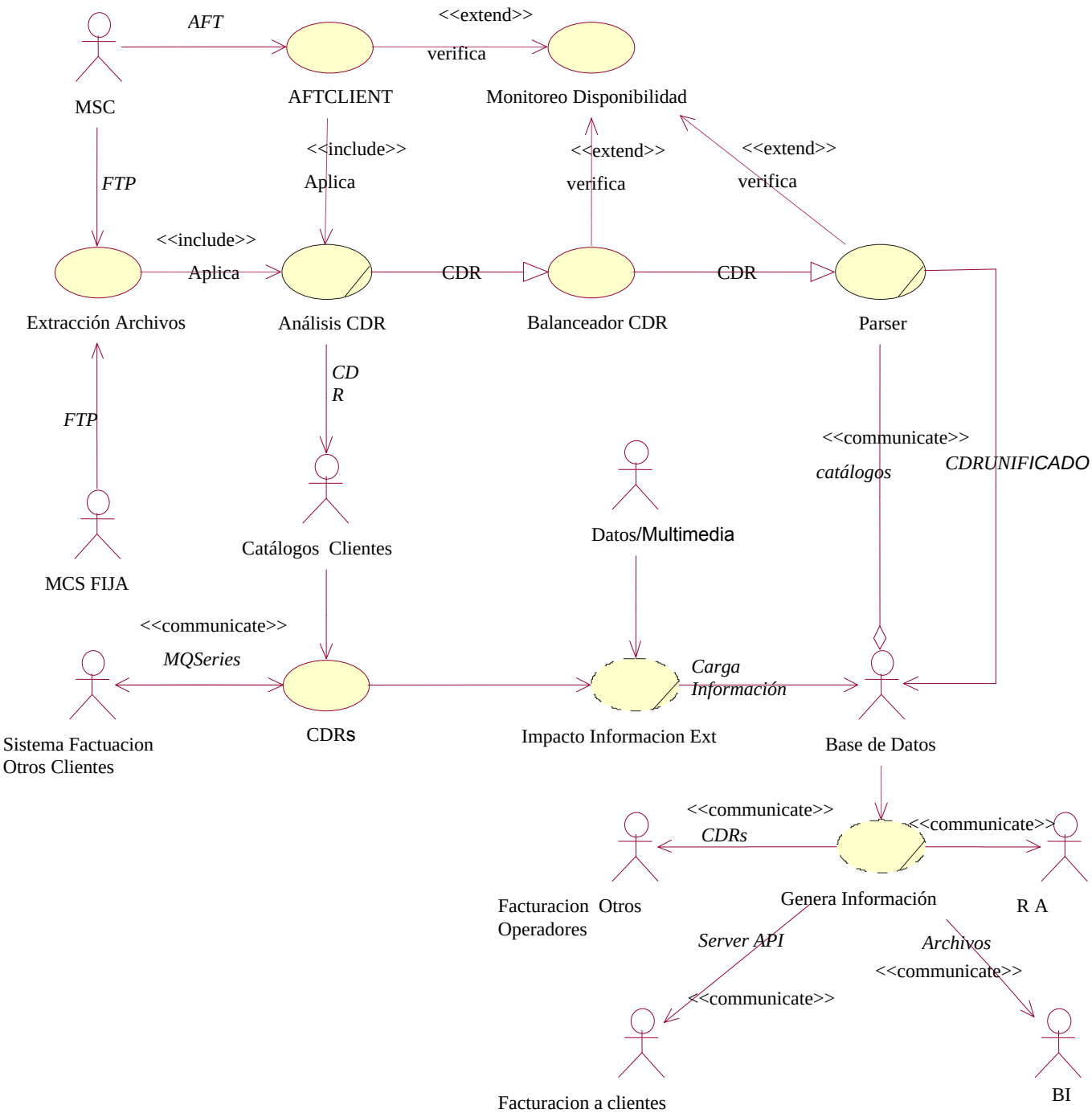

Figura 4.2.10 Diagrama general

4.2.2 Diagramas de secuencia.

Nombre del caso de uso: Recepción de trama de CDR.

El caso de uso comienza cuando se ejecuta la aplicación AFTCLIENT especificando el MSC Móvil con el cual va a interactuar posteriormente establece una conexión permanente mediante un socket y comienza a verificar cual es el archivo AMA que está procesando. En la figura 4.2.3.1 se muestra el diagrama de secuencia de la recepción de trama de CDR.

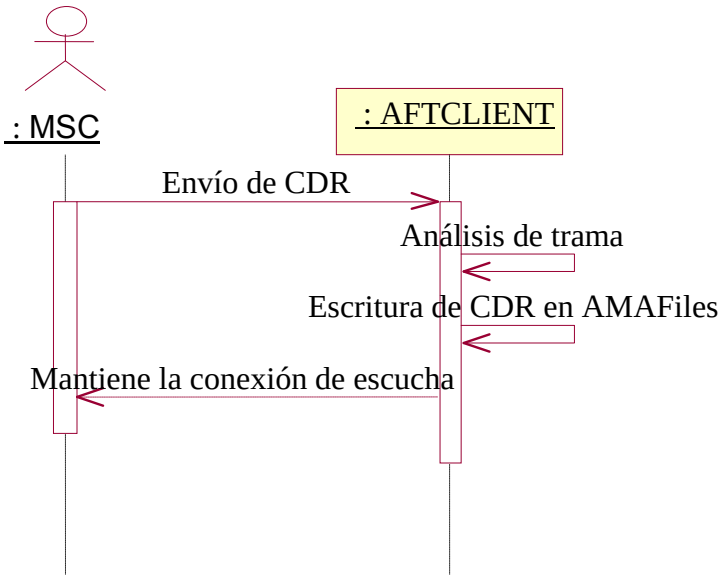

Figura 4.2.2.1 Diagrama de recepción de trama CDR

Nombre del caso de uso: Recepción de CDR MSC fija.

El caso de uso comienza cada hora cuando se realiza la conexión por FTP con el depositario de archivos del MSC fija, el proceso verifica cual es el último archivo extraído en la anterior conexión y verifica que exista un nuevo archivo para extraerlo por FTP. Revisa el formato de los registros que contiene el archivo y solo se analizan aquellos que tienen estatus de “STOP”, debido a que es el último

registro que se genera cuando se conmuta una llamada. En la figura 4.2.2.2 se muestra el diagrama de secuencia.

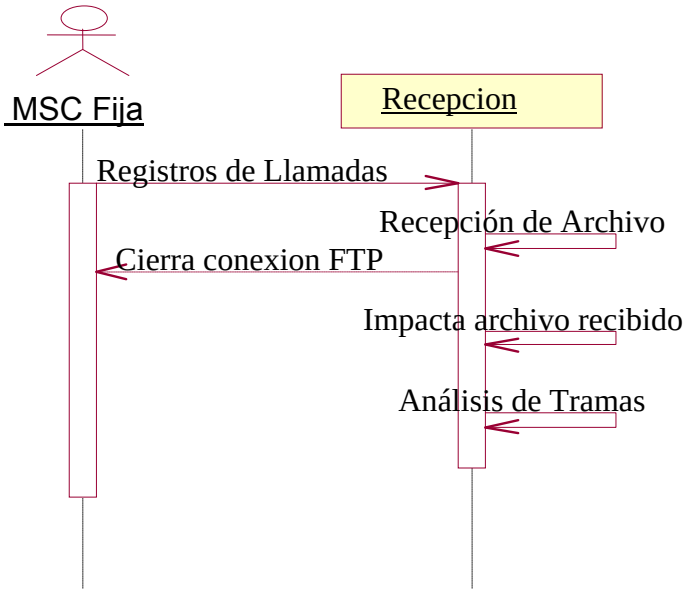

Figura 4.2.2.2 Diagrama de secuencia de recepción de CDRs MSC fija

Nombre del caso de uso: Balanceador de trama CDR.

Levanta la conexión del proceso hacia el parser, el análisis que está incluido en el proceso del AFTCLIENT envía la trama de CDR al balanceador, posteriormente el algoritmo de balanceador toma la trama de CDR en el formato enviado. El balanceador toma una dirección de destino para su envío y almacena la trama de manera temporal en una tabla asignada exclusiva para el parser SIMED destino.
Cuando el balanceador reciba respuesta de recepción por parte del parser SIMED elimina la trama de la tabla temporal, cuando el balanceador detecta que un parser SIMED se encuentra no disponible realiza otra revisión de parser SIMED disponibles para reenviar la trama CDR. Ver el diagrama de secuencia de la figura 4.2.2.3

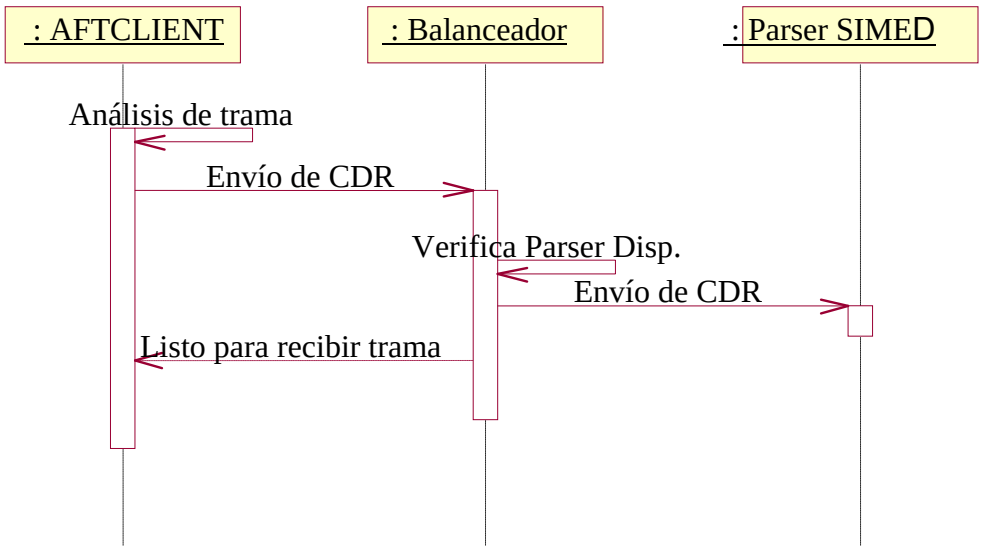

Figura 4.2.2.3 Diagrama de balanceador de trama CDR

Nombre del caso de uso: Parser SIMED.

Al levantar el sistema se cargan los catálogos necesarios para completar la información de los CDRs que ayudan en la clasificación del mismo posteriormente recibe la trama de CDR por parte del balanceador, la trama CDR de entrada se almacena en la cola de entrada El sistema del parser SIMED tiene configurados 50 hilos de conexión para la cola de entrada.

El proceso del parser SIMED descompone la trama de CDR para realizar el análisis de aquellos CDR que correspondan a llamadas completadas analizando el campo de duración el cual deberá de ser 0 o mayor. Después de realizar el proceso de incrementar la información con los catálogos y de realizar la clasificación del CDR, se deposita en la cola de salida para su impacto en la base de datos. En la figura 4.2.2.4 se muestra el diagrama de secuencia del parser SIMED.

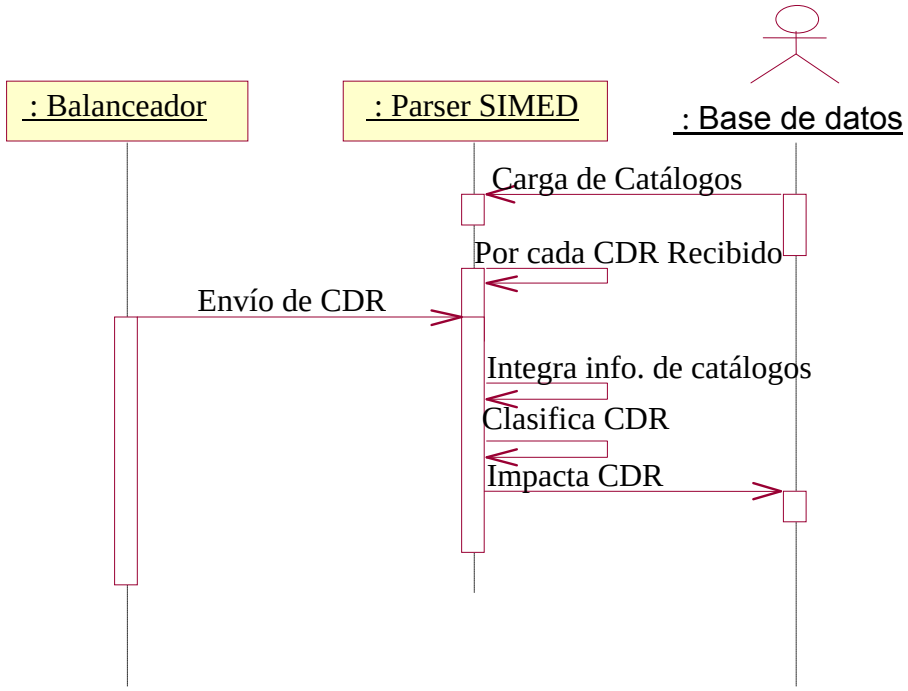

Figura 4.2.2.4 Diagrama de parser SIME

Nombre del caso de uso: Monitoreo de disponibilidad.

Las herramientas que monitorean el adecuado funcionamiento y disponibilidad del proceso de AFTCLIENT verificaran que se encuentre en ejecución el proceso. La salida se deposita en un archivo log para su análisis y el proceso analiza el archivo para verificar que se encuentre en ejecución el AFTCLIENT asociado a cada uno de los MSC fija. El monitoreo realiza el borrado de los archivos AMA para mantener el espacio disponible, solo se mantiene un periodo de 10 días de histórico.

La herramienta de monitoreo para el balanceo de las tramas CDR es necesario verificar que se encuentre en ejecución, comienza cuando analiza el archivo log de salida para verificar si se encuentra actualmente en ejecución el balanceador.

La herramienta de monitoreo de disponibilidad para el parser SIMED es necesario asegurar que se encuentre en ejecución, así como verificar el número de registros encolados en las colas de entrada y salida del parser SIMED.

La herramienta tendrá que analizar que los tres procesos se encuentren en ejecución, en caso contrario deberá de levantar el proceso faltante. Ver figura 4.2.2.5 con el diagrama de secuencia de este proceso.

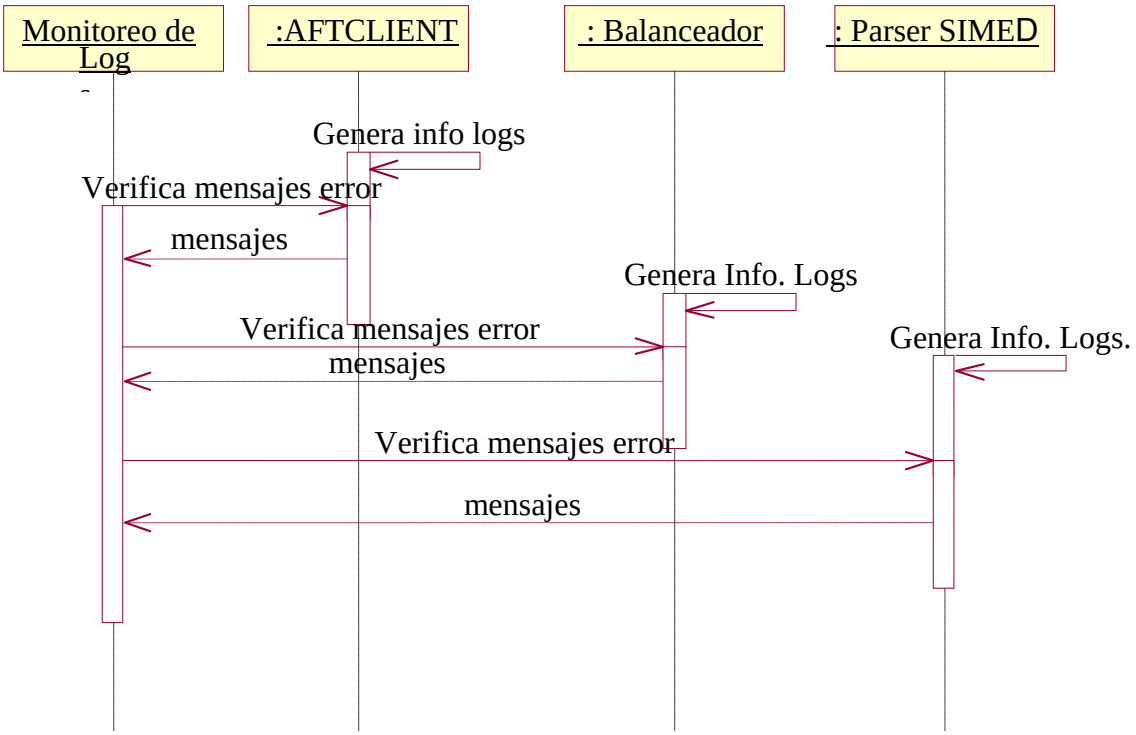

Figura 4.2.2.5 Diagrama de monitoreo de disponibilidad

Nombre del caso de uso: Carga de información externa.

El caso de uso comienza cuando se cuenta con los archivos correspondientes al proceso de facturación a clientes, estos contienen la información que se genera en nuestra red y la de otras compañías telefónicas. Estos archivos son depositados en el servidor en donde se encuentra la base de datos, el proceso tiene objetos de la base de datos indexados a archivos con nombres genéricos para su lectura. Por lo que los archivos recibidos se renombran para que se puedan acceder por la base de datos, el caso de uso termina con la inserción de datos en la base de datos. Para el caso de los datos y multimedia comienza con la recepción de los

archivos que corresponden a la información generada por estas plataformas, el formato que se deben de entregar deberá cumplir con el contrato de formato. El proceso realiza un análisis de los registros para ser insertados en tablas de la base de datos. La figura 4.2.2.6 muestra el diagrama de secuencia para la carga de información externa.

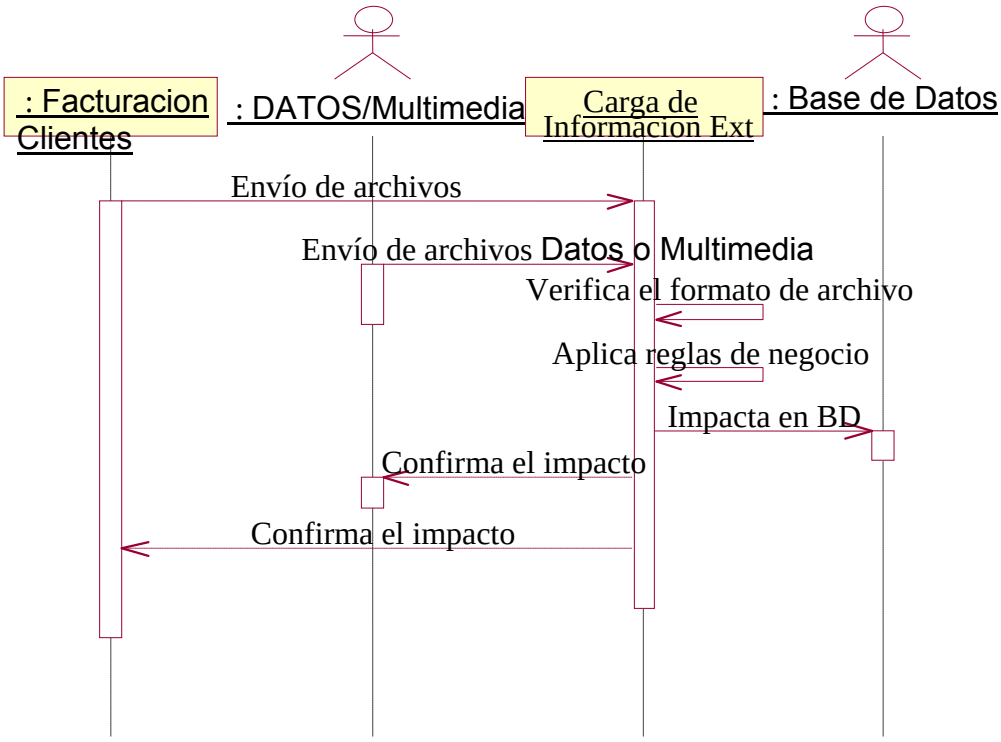

Figura 4.2.2.6 Diagrama de carga de información externa

Nombre del caso de uso: Generación de Información para facturación a clientes corporativos.

Se verifica cifras de la información impactada en la base de datos de los archivos de MSC fija, a partir de la información de los corporativos que se tienen registrados en la base de datos se extraen las llamadas generadas por los mismos, después se crean las tramas con el formato específico para su impacto al servidor, las tramas se depositan en la BD del SIMED en donde se clasifica por mes. Las tramas se generan de entrada y de salida de llamadas una vez que crea

la información, se forma un archivo con las tramas que se impactaran al servidor. El archivo se divide en tantos hilos de impacto que haya definido el área de facturación a clientes corporativos, para no afectar a la producción diaria. Se programa el socket que envía las tramas e impacta al servidor. Para la generación de las tramas de pospago, es necesario verificar cifras de ingresos en la base de datos de los archivos recibidos de los CDRs, con la información que se tiene registrada de los usuarios de pospago, se extraen todas las llamadas realizadas en el periodo del mes a procesar. Ver figura 4.2.2.7 con el diagrama de secuencia de este proceso.

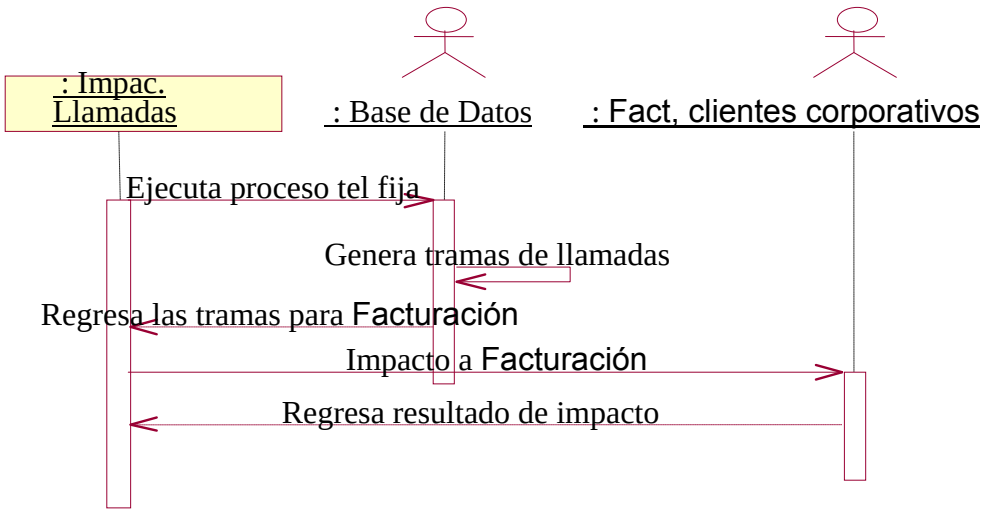

Figura 4.2.2.7 Diagrama de Generación de Información para Pospago

Nombre del caso de uso: Generación de información para facturación a otros operadores.

La información que se inserta en las tablas de mediación se replica aplicando reglas de negocio a las tablas de facturación a otros operadores e Intercambio de capacidad. La información es clasificada basándose en las reglas definidas por cada una de las áreas, la información se clasifica y deposita directamente en sus BD, teniendo la información generada el proceso deposita en archivos los diferentes anexos. Los archivos deberán de cumplir con el formato establecido en los contratos que se hayan firmado. Finaliza el proceso generando todos los

anexos necesarios por el área de interconexión. La figura 4.2.2.8 muestra el diagrama de secuencia para la carga de información externa.

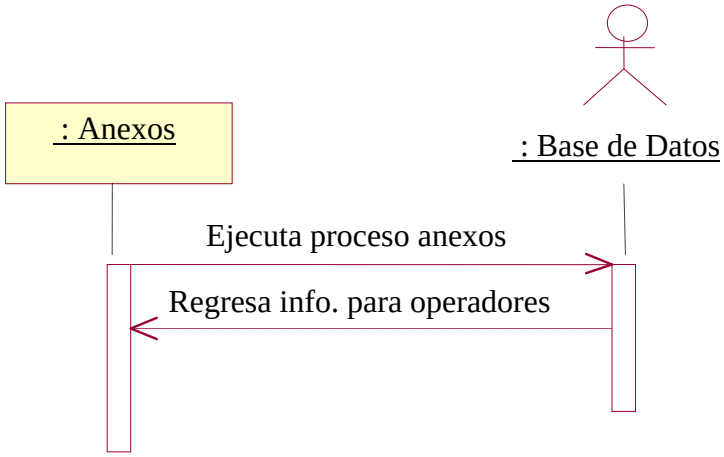

Figura 4.2.2.8 Diagrama de generación información para interconexión.

4.2.3. Diagrama entidad- relación.

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido, figura 4.2.3.1.

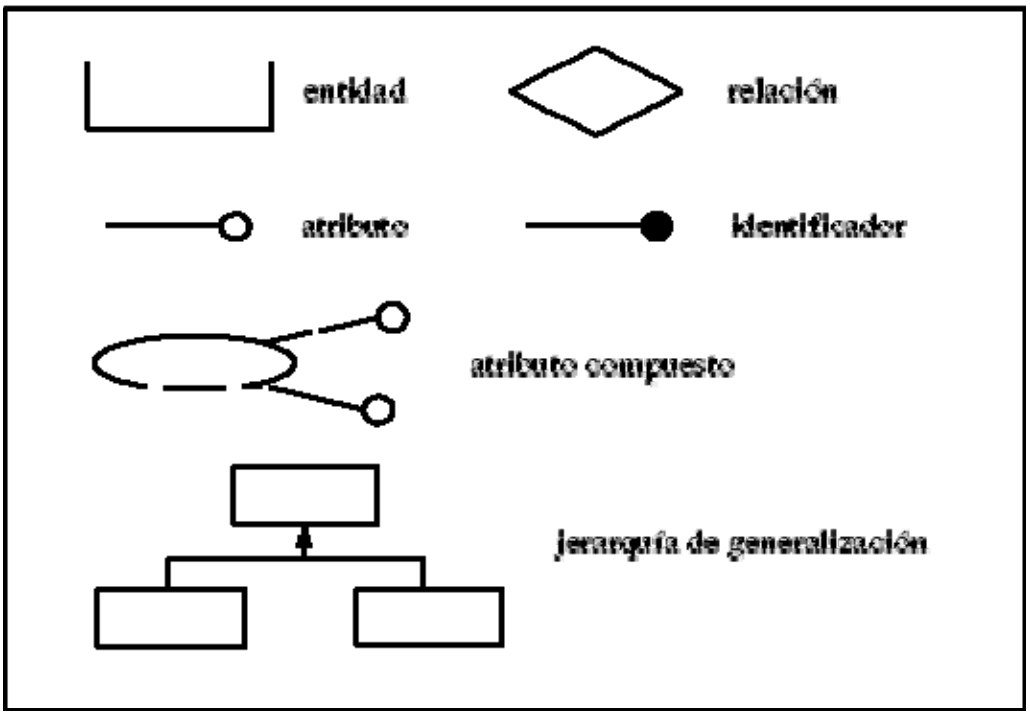

Figura 4.2.3.1 Conceptos del modelo entidad-relación extendido.

Entidad.

Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual.

Hay dos tipos de entidades: fuertes y débiles. Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil.

Relación (interrelación).

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc.

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

La cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial). Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio.

A veces, surgen problemas cuando se está diseñado un esquema conceptual. Estos problemas, denominados trampas, suelen producirse a causa de una mala interpretación en el significado de alguna relación, por lo que es importante comprobar que el esquema conceptual carece de dichas trampas. En general, para encontrar las trampas, hay que asegurarse de que se entiende completamente el significado de cada relación. Si no se entienden las relaciones, se puede crear un esquema que no represente fielmente la realidad.

Una de las trampas que pueden encontrarse ocurre cuando el esquema representa una relación entre entidades, pero el camino entre algunas de sus ocurrencias es ambiguo. El modo de resolverla es reestructurando el esquema para representar la asociación entre las entidades correctamente.

Otra de las trampas sucede cuando un esquema sugiere la existencia de una relación entre entidades, pero el camino entre una y otra no existe para algunas de sus ocurrencias. En este caso, se produce una pérdida de información que se puede subsanar introduciendo la relación que sugería el esquema y que no estaba representada.

Atributo.

Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos. Gráficamente, se representan mediante bolitas que cuelgan de las entidades o relaciones a las que pertenecen.

Cada atributo tiene un conjunto de valores asociados denominado dominio. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio.

Los atributos pueden ser simples o compuestos. Un atributo simple es un atributo que tiene un solo componente, que no se puede dividir en partes más pequeñas que tengan un significado propio. Un atributo compuesto es un atributo con varios componentes, cada uno con un significado por sí mismo. Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso. Un atributo compuesto se representa gráficamente mediante un óvalo.

Los atributos también pueden clasificarse en monovalentes o polivalentes. Un atributo monovalente es aquel que tiene un solo valor para cada ocurrencia de la entidad o relación a la que pertenece. Un atributo polivalente es aquel que tiene varios valores para cada ocurrencia de la entidad o relación a la que pertenece. A estos atributos también se les denomina multivaluados, y pueden tener un número máximo y un número mínimo de valores. La cardinalidad de un atributo indica el número mínimo y el número máximo de valores que puede tomar para cada ocurrencia de la entidad o relación a la que pertenece. El valor por omisión es (1,1).

Por último, los atributos pueden ser derivados. Un atributo derivado es aquel que representa un valor que se puede obtener a partir del valor de uno o varios atributos, que no necesariamente deben pertenecer a la misma entidad o relación.

Identificador.

Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad. Un identificador de una entidad debe cumplir dos condiciones:

1. No pueden existir dos ocurrencias de la entidad con el mismo valor del identificador.
2. Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.

Toda entidad tiene al menos un identificador y puede tener varios identificadores alternativos. Las relaciones no tienen identificadores.

Jerarquía de generalización.

Una entidad *E* es una generalización de un grupo de entidades E^1, E^2, ... E^n, si cada ocurrencia de cada una de esas entidades es también una ocurrencia de *E*.

Todas las propiedades de la entidad genérica *E* son heredadas por las subentidades.

Cada jerarquía es total o parcial, y exclusiva o superpuesta. Una jerarquía es total si cada ocurrencia de la entidad genérica corresponde al menos con una ocurrencia de alguna subentidad. Es parcial si existe alguna ocurrencia de la entidad genérica que no corresponde con ninguna ocurrencia de ninguna subentidad. Una jerarquía es exclusiva si cada ocurrencia de la entidad genérica corresponde, como mucho, con una ocurrencia de una sola de las subentidades. Es superpuesta si existe alguna ocurrencia de la entidad genérica que corresponde a ocurrencias de dos o más subentidades diferentes.

Un subconjunto es un caso particular de generalización con una sola entidad como subentidad. Un subconjunto siempre es una jerarquía parcial y exclusiva.

En la figura 4.2.3.2, se muestra el diagrama entidad-relación del SIMED.

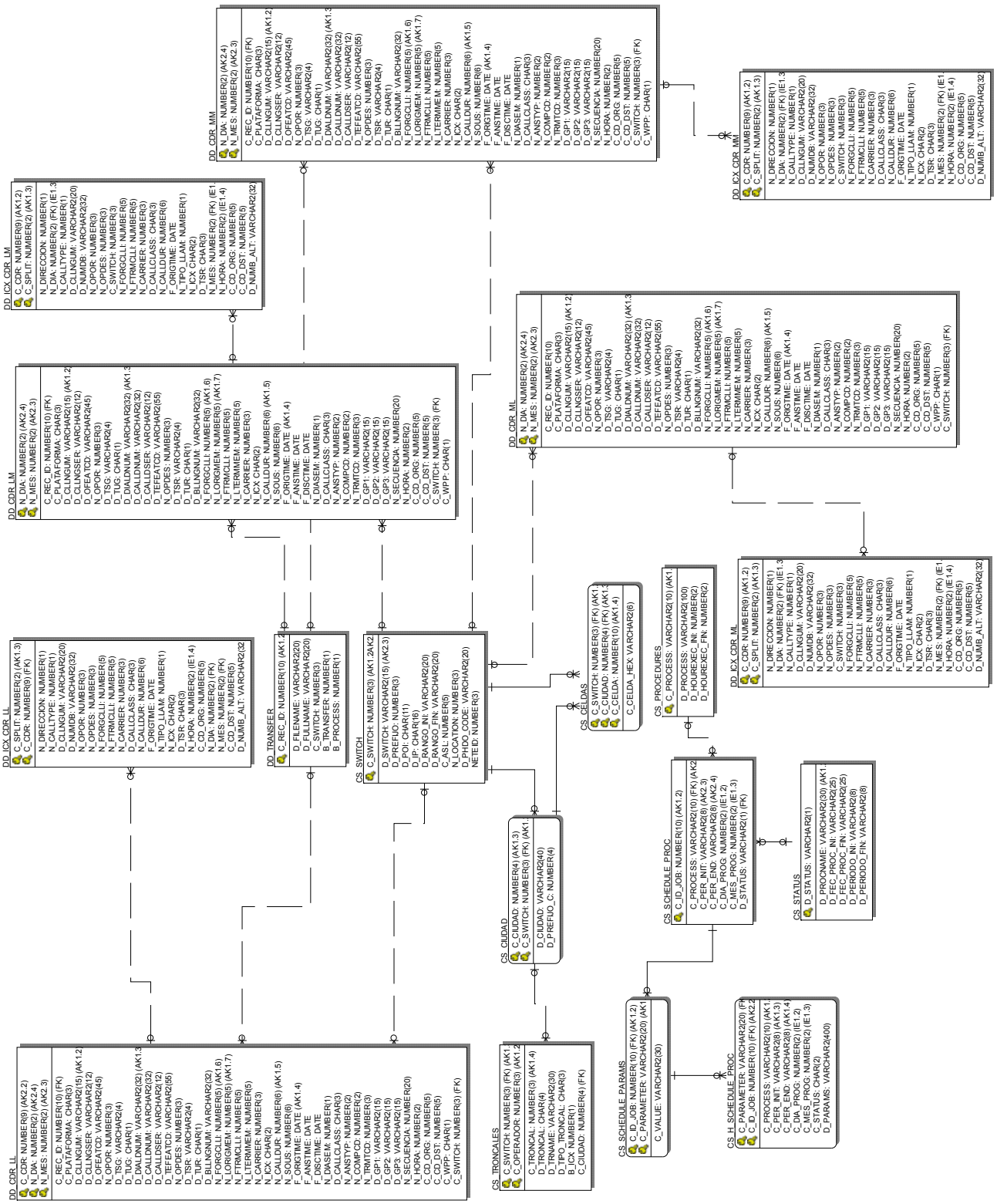

Figura 4.2.3.2 Diagrama entidad-relación

A continuación, se hace una breve descripción de las tablas contenidas en el diagrama entidad relación.

TABLA	DESCRIPCION
CS_ASIT_PHDO	Catálogo que contiene las claves necesarias para el envío de trafico de intercambio de capacidad
CS_CALLCLASS	Catalogo que almacena los diferentes tipos de llamadas a clasificar
CS_CARRIER	Catalogo que contiene todos los carriers de entrega local como de larga distancia con sus respectivas claves
CS_CARRIER_TARIFAS	Catálogo que contiene los cobros de cada carrier
CS_CELDAS	Catálogo que contiene todas las celdas de la compañía, su clave hexadecimal y su ubicación geográfica
CS_CIUDAD	Catálogo que contiene todas las ciudades donde la compañía tiene presencia, identificados por un ID único en toda la compañía
CS_CIUDAD_DAPLACE	Catálogo con las diferentes claves de región para todas las ciudades donde la compañía tiene presencia
CS_CORPORATIVOS	Catálogo que contiene a todos los clientes corporativos o empresas de la compañía, con su respectiva serie manejada así como su cabeza de serie necesaria para la facturación
CS_CORPORATIVOS_VRT	Catálogo que contiene otra modalidad de clientes corporativos de la compañía
CS_EXTRACTOS	Catálogo que almacena los diferentes extractos que se deben generar para otras aplicaciones
CS_H_SCHEDULE_PROC	Tabla para programar reportes y/o entregas periódicas
CS_INT_CARRIER	Catálogo que contiene todos los carriers internacionales con los que tiene convenio la compañía

CS_IRM_ASIT	Catálogo que contiene toda la numeración de IRM para el envío de intercambio de capacidad a facturación
CS_IRM_IU	Catálogo que contiene toda la numeración de IRM para intercambio de capacidad
CS_IRM_MIN	El catálogo de todos los IRM's y su respectivo MIN de todos los clientes de la compañía
CS_IRM_RANGE	El catálogo que contiene todos los rangos de IRM's manejados por la compañía
CS_IRM_UN	El catálogo que contiene todos los IRM's para una de las marcas de la compañía
CS_MIN_POST	Catálogo que contiene todos los mins de todos los clientes de pospago de la compañía
CS_PREFIX_REG	Catálogo con los prefijos para cada región
CS_PROCEDURES	Catálogo de los procedimientos almacenados existentes en la aplicación
CS_REPORTE_ROAMING	Tabla donde se almacenan los reportes de roaming entregados mensualmente
CS_REPORTE_TELFIJ	Tabla que contiene el reporte de telefonía fija entregado mensualmente
CS_REPORTE_TELFIJ_DETALLE	Tabla que contiene el reporte de telefonía fija a detalle
CS_REPORTE_TELFIJ_RESUMEN	Tabla que contiene el reporte de telefonía fija en resumen
CS_RETRO_PHDO	Tabla donde se almacenan todos los CDR's que van a ser enviados para su cobro a la plataforma de tarificación
CS_ROAMING	Tabla que almacena todas las tarifas de roaming para la compañía
CS_ROAMING_TMP	Tabla temporal para el proceso de roaming mensual

CS_RPM_NACIONAL	Tabla que contiene toda la numeración nacional entregada por COFETEL
CS_SCHEDULE_PARAMS	Catálogo con los parámetros necesarios para cada proceso de ejecución programado
CS_SCHEDULE_PROC	Catálogo de procesos programados
CS_SONUS	Catálogo que contiene información respecto al SoftSwitch SONUS
CS_SPECIAL_NUMBERS	Catálogo que contiene números especiales de los clientes de la compañía
CS_STATUS	Catálogo que contiene todos los posibles estatus manejados durante el proceso
CS_SWITCH	Catálogo que contiene todos los MSC de la compañía
CS_TELEFONIA_FIJA	Catálogo que contiene todos los clientes de telefonía fija de la compañía
CS_TRONCALES	Catálogo que contiene todas las troncales que utiliza la compañía para la entrega de trafico
DD_CDR_ASIT_LL	Tabla que contiene todo el tráfico de intercambio de capacidad de la compañía, que sea de calltype 4 LL o Land to Land
DD_CDR_ASIT_ML	Tabla que contiene todo el tráfico de intercambio de capacidad de la compañía, que sea de calltype 1 ML o Mobile to Land
DD_CDR_ASIT_MM	Tabla que contiene todo el tráfico de intercambio de capacidad de la compañía, que sea de calltype 0 MM o Mobile to Mobile
DD_CDR_LL	Tabla que contiene todos los CDR's recolectados de todas las MSC de la compañía de calltype 4, LL o Land to Land
DD_CDR_LM	Tabla que contiene todos los CDR's recolectados de

	todas las MSC de la compañía de calltype 3, LM o Land to Mobile
DD_CDR_ML	Tabla que contiene todos los CDR's recolectados de todas las MSC de la compañía de calltype 1, ML o Mobile to Land
DD_CDR_MM	Tabla que contiene todos los CDR's recolectados de todas las MSC de la compañía de calltype 0, MM o Mobile to Mobile
DD_ICX_CDR_LL	Tabla que contiene todos los CDR's recolectados de las MSC de la compañía de calltype 4, LL o Land to Land y que corresponden a interconexión
DD_ICX_CDR_LM	Tabla que contiene todos los CDR's recolectados de las MSC de la compañía de calltype 3, LM o Land to Mobile y que corresponden a interconexión
DD_ICX_CDR_ML	Tabla que contiene todos los CDR's recolectados de las MSC de la compañía de calltype 1, ML o Mobile to Land y que corresponden a interconexión
DD_ICX_CDR_MM	Tabla que contiene todos los CDR's recolectados de las MSC de la compañía de calltype 0, MM o Mobile to Mobile y que corresponden a interconexión
DD_TRANSFER	Tabla que almacena todos los archivos de CDR's de todas las centrales de la compañía
DD_UDR	Tabla que almacena todos los CDR's de datos
DD_UDR_TRANSFER	Tabla que almacena todos los archivos de CDR's de datos generados por la central

Como se puede observar, la mayoría de las tablas son catálogos necesarios para la clasificación de CDRs (CS_XX) en las tablas operativas (DD_XX). Esta validación y/o referencia se realiza con el módulo de catálogos de la aplicación, para su posterior inserción en la base de datos.

Estas tablas operativas contienen un índice UNIQUE definido sobre los campos D_CLLNGUM, D_DIALDNUM, F_ORIGTIME, N_CALLDUR, N_FORGCLLI, y N_LORIGMEM.. La combinación de estos campos asegura que no podrá insertarse un CDR duplicado. Adicionalmente la llave primaria de estas tablas será una secuencia de Oracle.

- D_CLLNGUM = Número que origina la llamada
- D_DIALDNUM = Número que marca el usuario
- F_ORIGTIME = Fecha y hora de origen de la llamada
- N_CALLDUR = Duración de la llamada expresada en segundos
- N_LORIGMEM = Campo de banderas de Features de la MSC utilizados en la llamada.

Dado que los campos que se insertan en las tablas operativas son complementados y determinados en el módulo de parseo, se decidió que esta validación fuera la única a realizarse y omitir una validación por integridad referencial dentro de la base de datos, cuidando solo la integridad de entidad.

4.2.4 Diagrama de clases.

Diagrama de clases para el módulo de parseo.

La clase principal es la clase ParseThread que a su vez utiliza las clases phone y Trunk para clasificar cada uno de los CDR's, estas clases son las que implementan las interfaces de intercambio mediante RMI entre el módulo de catálogos y el de parseo.
A su vez la clase status control le permite a la clase ParseThread enviar a base de datos el CDR final para su inserción, mediante una queue (cola) de salida, de igual manera la clase CmdServer permite interactuar con las queue's de la aplicación

tanto del módulo de parseo como de la de base de datos, e incluso compartir la configuración con la que se encuentra funcionando, figura 4.2.4.1.

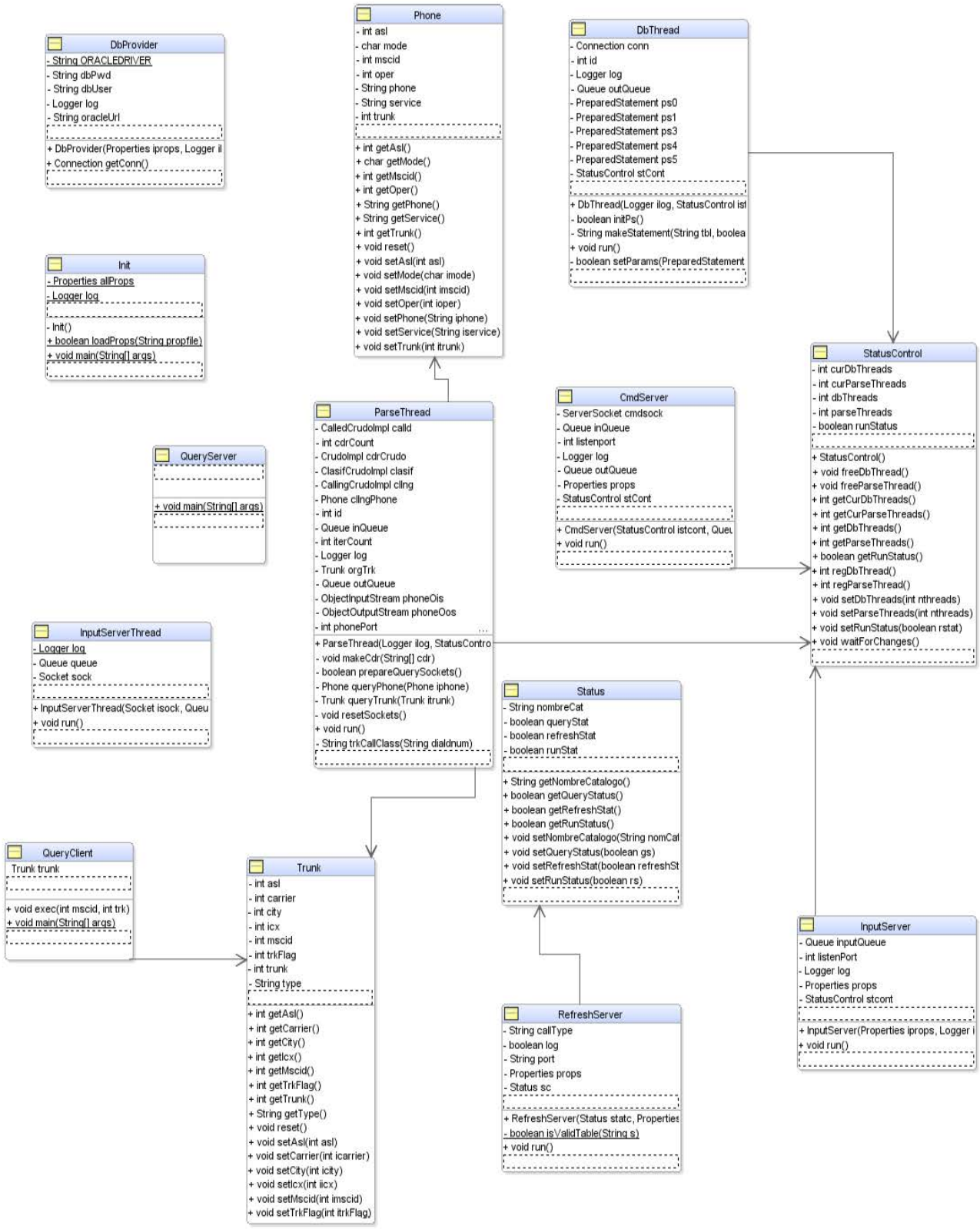

Figura 4.2.4.1 Diagrama de clases para el módulo de parseo.

Diagrama de clases para el módulo de catálogos.

La clase principal InitCat, es la encargada de establecer la conexión a la base de datos e implementa clases genéricas como catálogos y DBProvider, publica las interfaces de los objetos phone y trunk mediante RMI, figura 4.2.4.2.

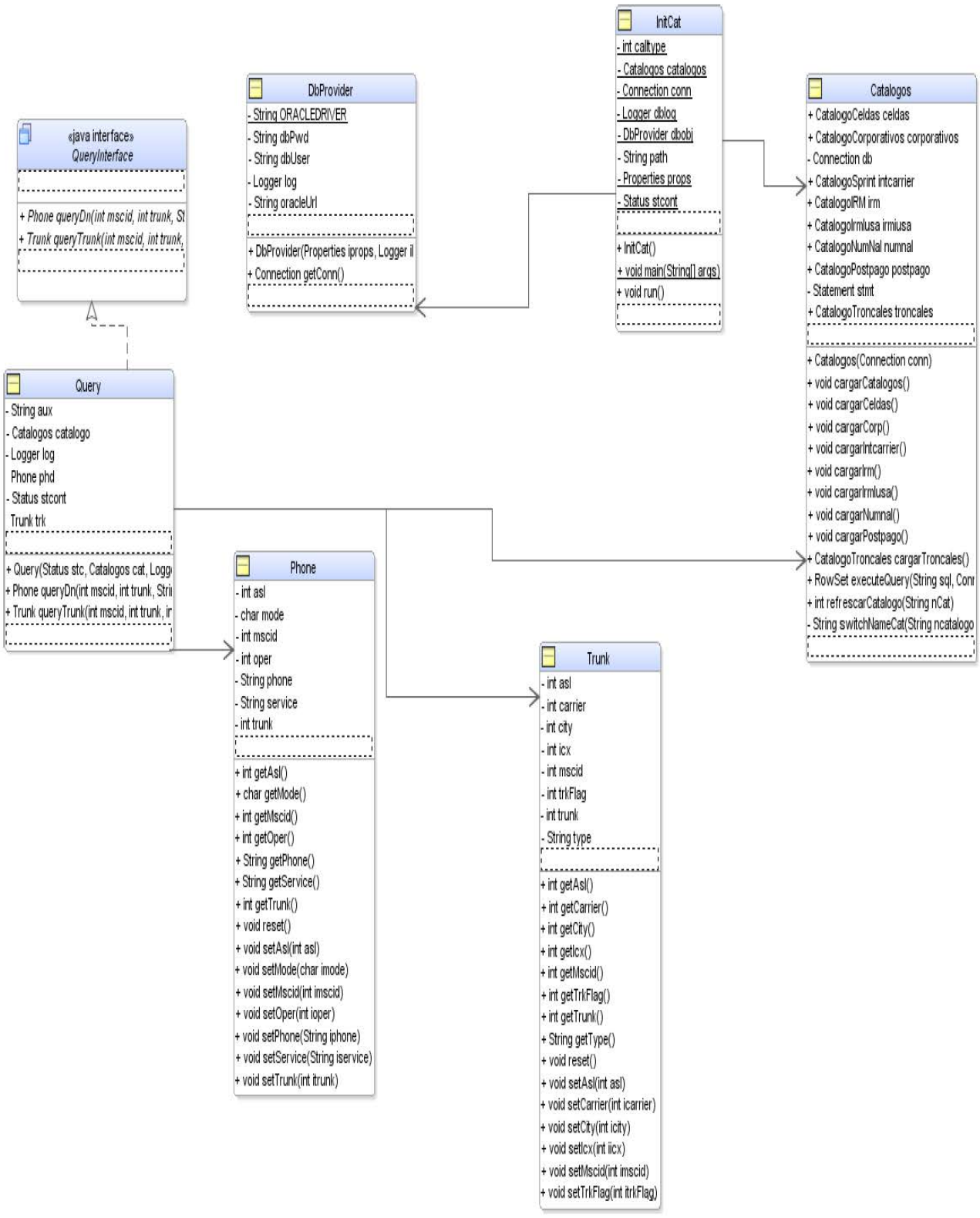

Figura 4.2.4.2 Diagrama de clases para el módulo de catálogos.

4.2.5 Diccionario de datos.

Un diccionario de datos es un conjunto de metadatos que contiene las características lógicas y puntuales de los datos que se van a utilizar en el sistema que se programa, incluyendo nombre, descripción, alias, contenido y organización.

Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

TABLA: CS_CORPORATIVOS_VRT
Catálogo que contiene otra modalidad de clientes corporativos de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_COMP	VARCHAR2(10)	NOT NULL	PK		
D_INT_TRUNK	VARCHAR2(20)	NOT NULL	PK		
D_HEADER	VARCHAR2(10)	NOT NULL			
C_SWITCH	NUMBER(3)	NOT NULL			

Tabla 4.2.5.1. CS_CORPORATIVOS_VRT

TABLA: CS_CORPORATIVOS
Catálogo que contiene a todos los clientes corporativos o empresas de la compañía, con su respectiva seria manejada así como su cabeza de serie necesaria para la facturación.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_COMP	VARCHAR2(10)	NOT NULL	PK		

D_INT_TRUNK	VARCHAR2(50)	NOT NULL			
D_HEADER	VARCHAR2(10)	NOT NULL			
C_SWITCH	NUMBER(3)	NOT NULL			

Tabla 4.2.5.2. CS_CORPORATIVOS

TABLA: CS_INT_CARRIER
Catálogo que contiene todos los carriers internacionales con los que tiene convenio la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
D_COMPANY	VARCHAR2(25)	NOT NULL	PK		
D_REGION	VARCHAR2(50)	NOT NULL	PK		
N_INICIO	VARCHAR2(10)	NOT NULL			
N_FIN	VARCHAR2(10)	NOT NULL			

Tabla 4.2.5.3. CS_INT_CARRIER

TABLA: DD_CDR_LL
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 4, LL o Land to Land.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
N_DIA	NUMBER(2)	NOT NULL	PK		
N_MES	NUMBER(2)	NOT NULL	PK		
C_REC_ID	NUMBER(10)	NOT NULL			
C_PLATAFORMA	CHAR(3)	NOT NULL			
D_CLLNGUM	VARCHAR2(15)	NOT NULL			
D_CLLGSER	VARCHAR2(12)	NOT NULL			

Tabla 4.2.5.4. DD_CDR_LL

TABLA: CS_CIUDAD
Catálogo que contiene todas las ciudades donde la compañía tiene presencia, identificados por un ID único en toda la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
CS_CIUDAD	NUMBER(4)	NOT NULL	PK		
CS_SWITCH	NUMBER(3)	NOT NULL	PK/FK		
D_CIUDAD	VARCHAR2(40)	NOT NULL			
D_PREFIJO_C	NUMBER(4)	NOT NULL			

Tabla 4.2.5.5. CS_CIUDA

TABLA: CS_ROAMING
Tabla que almacena todas las tarifas de roaming para la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
N_MES	NUMBER(2)	NOT NULL			
N_DIA	NUMBER(2)	NOT NULL			
D_CLIENTE	CHAR(10)	NOT NULL			
D_STATUS	CHAR(1)	NOT NULL			
D_STATUS_PROC	CHAR(1)	NOT NULL			
C_ROAMING	NUMBER(1)	NOT NULL			

Tabla 4.2.5.6. CS_ROAMING

TABLA: CS_RETRO_PHDO
Tabla donde se almacenan todos los CDRs que van a ser enviados para su cobro a la plataforma de tarificación.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		

N_MES	NUMBER(2)	NOT NULL			
N_DIA	NUMBER(2)	NOT NULL			
D_CHAIN	VARCHAR2(160)	NOT NULL			
D_STATUS	CHAR(1)	NOT NULL			
D_STATUS_PROC	CHAR(1)	NOT NULL			
C_RESPUESTA	NUMBER(6)	NOT NULL			
C_SWITCH	NUMBER(3)	NOT NULL			

Tabla 4.2.5.7. CS_RETRO_PHDO

TABLA: CS_SCHEDULE_PROC
Catálogo de procesos programados.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_ID_JOB	NUMBER(10)	NOT NULL	PK		
C_PROCESS	VARCHAR2(10)	NOT NULL			
C_PER_INIT	VARCHAR2(8)	NOT NULL			
C_PER_END	VARCHAR(8)	NOT NULL			
C_DIA_PROG	NUMBER(2)	NOT NULL			
C_MES_PROG	NUMBER(2)	NOT NULL			

Tabla 4.2.5.8. CS_SCHEDULE_PROC

TABLA: DD_ICX_CDR_LL
Tabla que contiene todos los CDRs recolectados de las MSC de la compañía de calltype 4, LL o Land to Land y que corresponden a interconexión.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK/FK		
C_SPLIT	NUMBER(2)	NOT NULL	PK		

N_DIRECCION	NUMBER(1)	NOT NULL			
N_CALLTYPE	NUMBER(1)	NOT NULL			
D_CLLNGUM	VARCHAR2(20)	NOT NULL			
D_NUMDB	VARCHAR2(32)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			
N_OPDES	NUMBER(3)	NOT NULL			

Tabla 4.2.5.9. DD_ICX_CDR_L

TABLA: CS_SWITCH
Catálogo que contiene todos los MSC de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_SWITCH	NUMBER(3)	NOT NULL	PK		
D_SWITCH	VARCHAR2(15)	NOT NULL			
D_PREFIJO	NUMBER(3)	NOT NULL			
D_POI	CHAR(11)	NOT NULL			
D_IP	CHAR(16)	NOT NULL			
D_RANGO_INI	VARCHAR2(20)	NOT NULL			
D_RANGO_FIN	VARCHAR2(20)	NOT NULL			
C_ASL	NUMBRE(5)	NOT NULL			

Tabla 4.2.5.10. CS_SWITCH

TABLA: CS_CELDAS
Catálogo que contiene todas las celdas de la compañía, su clave hexadecimal y su ubicación geográfica.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_SWITCH	NUMBER(3)	NOT NULL	PK/FK		
C_CIUDAD	NUMBER(4)	NOT NULL	PK/FK		
C_CELDA	NUMBER(10)	NOT NULL	PK		
C_CELDA_HEX	VARCHAR2(5)	NOT NULL			

Tabla 4.2.5.11. CS_CELDAS

TABLA: DD_TRANSFER
Tabla que almacena todos los archivos de CDRs de todas las centrales de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_REC_ID	NUMBER(10)	NOT NULL	PK		
D_FILENAME	VARCHAR2(20)	NOT NULL			
D_FULLNAME	VARCHAR2(20)	NOT NULL			
C_SWITCH	NUMBER(3)	NOT NULL			
B_TRANSFER	NUMBER(1)	NOT NULL			
B_PROCESS	NUMBER(1)	NOT NULL			

Tabla 4.2.5.12. DD_TRANSFER

TABLA: CS_TRONCALES
Catálogo que contiene todas las troncales que utiliza la compañía para la entrega de tráfico.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_SWITCH	NUMBER(3)	NOT NULL	PK		
C_OPERADOR	NUMBER(3)	NOT NULL	PK		
C_TRONCAL	NUMBER(3)	NOT NULL			
D_TRONCAL	CHAR(4)	NOT NULL			

D_TRNAME	VARCHAR2(30)	NOT NULL			
D_TIPO_TRONCAL	CHAR(3)	NOT NULL			
B_ICX	NUMBER(3)	NOT NULL			
C_CIUDAD	NUMBER(3)	NOT NULL			

Tabla 4.2.5.13. CS_TRONCALES

TABLA: IRM_MIN.
El catálogo de todos los IRM´s y su respectivo MIN de todos los clientes de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
MIN	CHAR(10)	NOT NULL	PK		
IRM	CHAR(10)	NOT NULL			

Tabla 4.2.5.14. IRM_MIN.

TABLA: CS_CARRIER
Catálogo que contiene los cobros de cada carrier.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
CARRIER	VARCHAR2(5)	NOT NULL	PK		
TARIFA	NUMBER(4,2)	NOT NULL			

Tabla 4.2.5.15. CS_CARRIER

TABLA: CS_PROCEDURES
Catálogo de los procedimientos almacenados existentes en la aplicación.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_PROCESS	VARCHAR2(10)	NOT NULL	PK		
D_PROCESS	VARCHAR2(100)	NOT NULL			
D_HOUREXEC_INI	NUMBER(2)	NOT NULL			

D_HOUREEXEC_FIN	NUMBER(2)	NOT NULL			

Tabla 4.2.5.16. CS_PROCEDURES

TABLA: CS_RPM_NACIONAL
Tabla que contiene toda la numeración nacional entregada por COFETEL.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
MIN	CHAR(25)	NOT NULL	PK		
CITY	CHAR(25)	NOT NULL			

Tabla 4.2.5.17. CS_RPM_NACIONAL

TABLA: CS_PREFIX_REG
Catálogo con los prefijos de cada región.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_REGION	VARCHAR2(50)	NOT NULL	PK		
D_PREFIX	NUMBER(5)	NOT NULL			
C_TARIFA	NUMBER(4,2)	NOT NULL			
D_CIUDAD	VARCHAR2(55)	NOT NULL			

Tabla 4.2.5.18. CS_PREFIX_REG

TABLA: CS_MIN_POST
Catálogo que contiene todos los MIN´s de todos los clientes de postpago de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_NUM_POST	CHAR(10)	NOT NULL	PK		
N_CUENTA	NUMBER(10)	NOT NULL			

Tabla 4.2.5.19. CS_MIN_POST

TABLA: CS_REPORTE_TELFIJ
Tabla que contiene el reporte de telefonía fija entregado mensualmente.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_MES	NUMBER(2)	NOT NULL	PK		
N_DIA_INI	NUMBER(2)	NOT NULL	PK		
N_DIA_FIN	NUMBER(2)	NOT NULL	PK		
C_RESPUESTA	NUMBER(10)	NOT NULL	PK		
D_CHAIN	VARCHAR2(1000)	NOT NULL			

Tabla 4.2.5.20. CS_REPORTE_TELFIJ

TABLA: CS_H_SCHEDULE_PROC
Tabla para programar reportes y/o entregas periódicas.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_ID_JOB	NUMBER(10)	NOT NULL			
C_PROCESS	VARCHAR2(10)	NOT NULL			
C_PER_INIT	VARCHAR2(8)	NOT NULL			
C_PER_END	VARCHAR2(8)	NOT NULL			
C_DIA_PROG	NUMBER(2)	NOT NULL			
C_MES_PROG	NUMBER(2)	NOT NULL			
C_STATUS	CHAR(2)	NOT NULL			
D_PARAMS	VARCHAR2(400)	NOT NULL			

Tabla 4.2.5.21. CS_H_SCHEDULE_PROC

TABLA: CS_SCHEDULE_PARAMS
Catálogo con los parámetros necesarios para cada proceso de ejecución programado.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_ID_JOB	NUMBER(10)	NOT NULL	PK		

C_PARAMETER	VARCHAR2(20)	NOT NULL	PK		
C_VALUE	VARCHAR2(30)	NOT NULL			

Tabla 4.2.5.22. CS_SCHEDULE_PARAMS

TABLA: CS_STATUS
Catálogo que contiene todos los posibles estatus manejados durante el proceso.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
D_PROCNAME	VARCHAR2(30)	NOT NULL	PK		
D_STATUS	VARCHAR2(25)	NOT NULL			
D_FEC_PROC_INI	VARCHAR2(25)	NOT NULL			
D_FEC_PROC_FIN	VARCHAR2(25)	NOT NULL			

Tabla 4.2.5.23. CS_STATUS

TABLA: DD_CDR_LM
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 3, LM o Land to Mobile.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_DIA	NUMBER(2)	NOT NULL	PK		
N_MES	NUMBRE(2)	NOT NULL	PK		
C_REC_ID	NUMBER(10)	NOT NULL			
C_PLATAFORMA	CHAR(3)	NOT NULL			
D_CLLNGUM	VARCHAR2(15)	NOT NULL			
D_CLLNGSER	VARCHAR2(12)	NOT NULL			
D_OFEATCD	VARCHAR2(45)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.24. DD_CDR_LM

TABLA: DD_CDR_ML
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 1, ML o Mobile to Land.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_DIA	NUMBER(2)	NOT NULL	PK		
N_MES	NUMBRE(2)	NOT NULL	PK		
C_REC_ID	NUMBER(10)	NOT NULL			
C_PLATAFORMA	CHAR(3)	NOT NULL			
D_CLLNGUM	VARCHAR2(15)	NOT NULL			
D_CLLNGSER	VARCHAR2(12)	NOT NULL			
D_OFEATCD	VARCHAR2(45)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.25. DD_CDR_ML

TABLA: CS_SPECIAL_NUMBERS
Catálogo que contiene números especiales de los clientes de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_NUMBER	VARCHAR2(10)	NOT NULL	PK		
D_NUMBER	VARCHAR2(500)	NOT NULL			

Tabla 4.2.5.26. CS_SPECIAL_NUMBERS

TABLA: CS_TELEFONIA_FIJA
Catálogo que contiene todos los clientes de Telefonía fija de la compañía.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
N_MES	NUMBRE(2)	NOT NULL			

N_DIA	NUMBER(2)	NOT NULL			
D_CLIENTE	CHAR(10)	NOT NULL			
D_STATUS	CHAR(1)	NOT NULL			
D_STATUS_PROC	CHAR(1)	NOT NULL			
C_ROAMING	NUMBER(1)	NOT NULL			
C_RESPUESTA	NUMBER(5)	NOT NULL			

Tabla 4.2.5.27. CS_TELEFONIA_FIJA

TABLA: DD_CDR_MM
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 0, MM o Mobile to Mobile.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_DIA	NUMBER(2)	NOT NULL	PK		
N_MES	NUMBRE(2)	NOT NULL	PK		
C_REC_ID	NUMBER(10)	NOT NULL			
C_PLATAFORMA	CHAR(3)	NOT NULL			
D_CLLNGUM	VARCHAR2(15)	NOT NULL			
D_CLLNGSER	VARCHAR2(12)	NOT NULL			
D_OFEATCD	VARCHAR2(45)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.28. DD_CDR_MM

TABLA: CS-REPORTE_ROAMING
Tabla donde se almacenan los reportes de roaming entregados mensualmente.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
N_MES	NUMBER(2)	NOT NULL	PK		

N_DIA_INI	NUMBRE(2)	NOT NULL	PK		
N_DIA_FIN	NUMBER(2)	NOT NULL	PK		
C_RESPUESTA	NUMBER(10)	NOT NULL	PK		
D_CHAIN	VARCHAR2(1000)	NOT NULL			

Tabla 4.2.5.29. CS-REPORTE_ROAMING

TABLA: CS_CARRIER
Catálogo que contiene todos los carriers de entrega local como de larga distancia con sus respectivas claves.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CARRIER	VARCHAR2(4)	NOT NULL	PK		
D_CARRIER	VARCHAR2(50)	NOT NULL			
C_CUENTA	VARCHAR2(7)	NOT NULL			
D_DN	VARCHAR2(10)	NOT NULL			
D_MIN	VARCHAR2(10)	NOT NULL			
D_RATEPLAN	VARCHAR2(10)	NOT NULL			

Tabla 4.2.5.30. CS_CARRIER

TABLA: DD_ICX_CDR_ML
Tabla que contiene todos los CDRs recolectados de las MSC de la compañía calltype 1, ML o Mobile to Land y que corresponden a interconexión.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
C_SPLIT	NUMBRE(2)	NOT NULL	PK		
N_DIRECCION	NUMBER(1)	NOT NULL			
N_DIA	NUMBER(2)	NOT NULL	FK		

N_CALLTYPE	NUMBER(1)	NOT NULL			
D_CLLNGUM	VARCHAR2(20)	NOT NULL			
D_NUMDB	VARCHAR2(32)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.31. DD_ICX_CDR_ML

TABLA: DD_ICX_CDR_MM
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 0, MM o Mobile to Mobile y que corresponden a interconexión.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
C_SPLIT	NUMBRE(2)	NOT NULL	PK		
N_DIRECCION	NUMBER(1)	NOT NULL			
N_DIA	NUMBER(2)	NOT NULL	FK		
N_CALLTYPE	NUMBER(1)	NOT NULL			
D_CLLNGUM	VARCHAR2(20)	NOT NULL			
D_NUMBD	VARCHAR2(32)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.32. DD_ICX_CDR_MM

TABLA: DD_ICX_CDR_LM
Tabla que contiene todos los CDRs recolectados de todas las MSC de la compañía de calltype 3, LM o Land to Mobile y que corresponden a interconexión.

Columna	Tipo de dato Longitud	Acepta Nulos	PK o FK	Tabla de Referencia	Breve Descripción
C_CDR	NUMBER(9)	NOT NULL	PK		
C_SPLIT	NUMBRE(2)	NOT NULL	PK		

N_DIRECCION	NUMBER(1)	NOT NULL			
N_DIA	NUMBER(2)	NOT NULL	FK		
N_CALLTYPE	NUMBER(1)	NOT NULL			
D_CLLNGUM	VARCHAR2(20)	NOT NULL			
D_NUMBD	VARCHAR2(32)	NOT NULL			
N_OPOR	NUMBER(3)	NOT NULL			

Tabla 4.2.5.33. DD_ICX_CDR_LM

4.3 Diseño y construcción del back end

A continuación haremos una breve descripción del proceso que se sigue para el diseño y construcción de una base de datos en Oracle SQL Developer 1.2.1 de agosto de 2007. Describiremos como crear una tabla, una consulta y los triggers necesarios considerando que la aplicación en cuestión está conceptualizada como un daemon.

El objetivo fundamental es proporcionar una interfaz más amigable para la consulta y programación de la base de datos Oracle. La funcionalidad disponible es sólo parte de la disponibilidad a través de comandos en SQL*Plus, pero se corresponde con las tareas más habituales de interacción, programación y depuración de código sobre la base de datos.

Ejecución de sentencias SQL.

Escribir las sentencias SQL en la ventana de edición SQL (SQL Worksheet), figura 4.3.1.

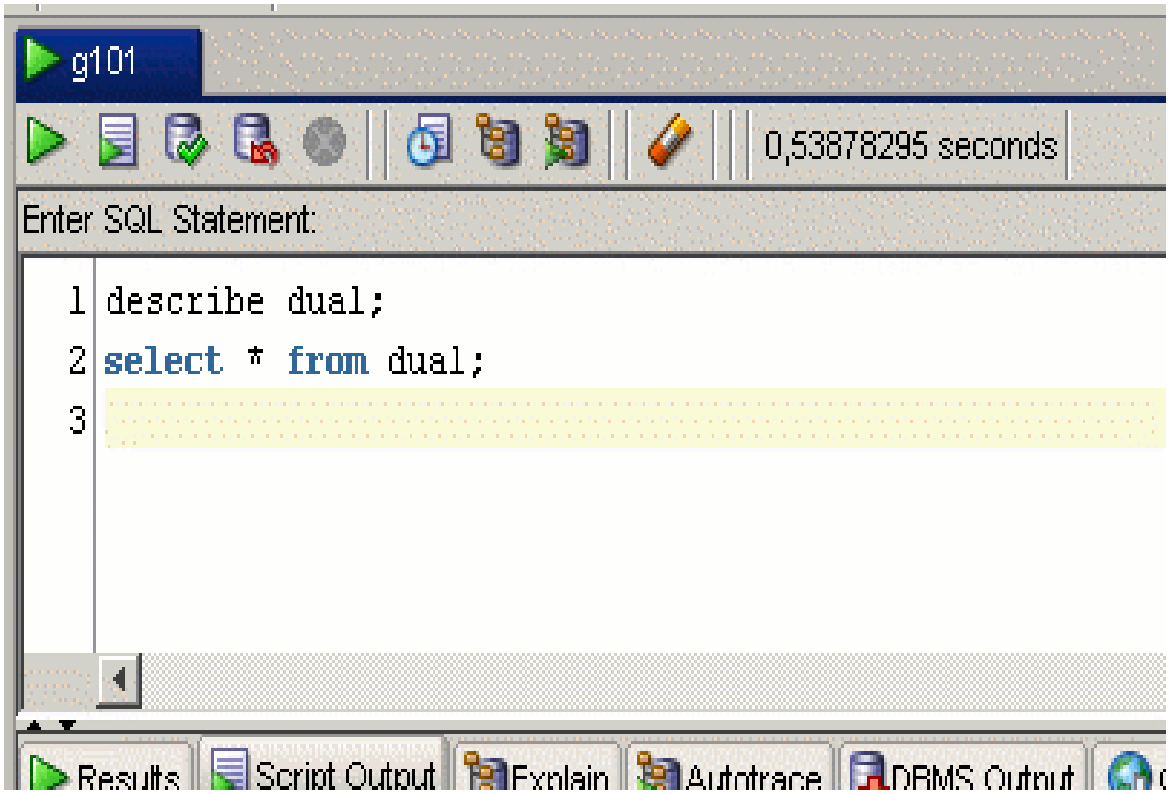

Figura 4.3.1 Ejecución de sentencias.

Para ejecutar sólo una sentencia, se sitúa el cursor sobre la sentencia y se pulsa el icono o la tecla F9.

Para ejecutar todas las sentencias, se pulsa el icono o la tecla F5.

Los resultados de la ejecución de las sentencias SQL se muestran en las pestañas “Results” y “Script Output”, figura 4.3.2.

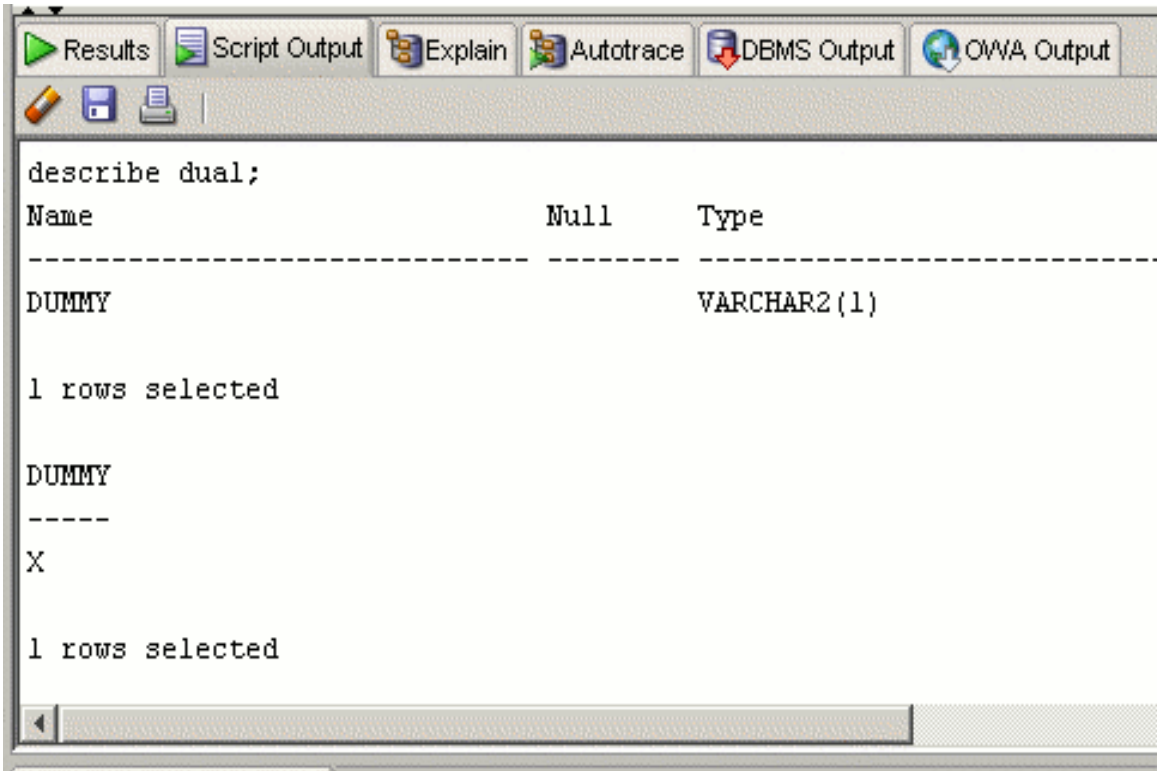

Figura 4.3.2 Pestañas “Results” y “Script Output”.

El icono permite acceder a un histórico de sentencias SQL ejecutadas. Para cargar una sentencia del histórico sobre el editor SQL se hace doble-clik sobre la sentencia.

El icono borra el contenido del editor SQL.

Para ver el número de línea en el editor SQL hay que activar Tools □□Preferentes □□Code Editor □□Line Gutter □□Show Line Numbers.

Para grabar a un fichero .SQL el contenido del editor SQL se utiliza la opción File □□Save o el icono.

Para abrir un fichero .SQL en el editor SQL se utiliza la opción File □□Open o el icono.

Para abrir un nuevo editor SQL se utiliza la opción Tools □□SQL Worksheet o el icono.

Para crear y editar un nuevo fichero SQL se utiliza la opción File □□New □□SQL File

IMPORTANTE: Las sentencias SQL que modifican la base de datos (INSERT INTO, UPDATE, DELETE, ...) no se realizan (ejecutan) en la base de datos hasta que se pulsa el icono .

Si se quiere que las sentencias SQL se realicen automáticamente después de ejecutarlas hay que activar la opción Tools □□Preferences □□Database □□Worksheet Parameters □□Autocommit in SQL Worksheet

Para que los cambios realizados por sentencias SQL de creación de objetos (DDL) se reflejen en el navegador de objetos, es necesario pulsar el icono "Refresh".

Creación de tablas.

Pulsar el botón derecho sobre el icono “Tables” de la conexión, figura 4.3.3.

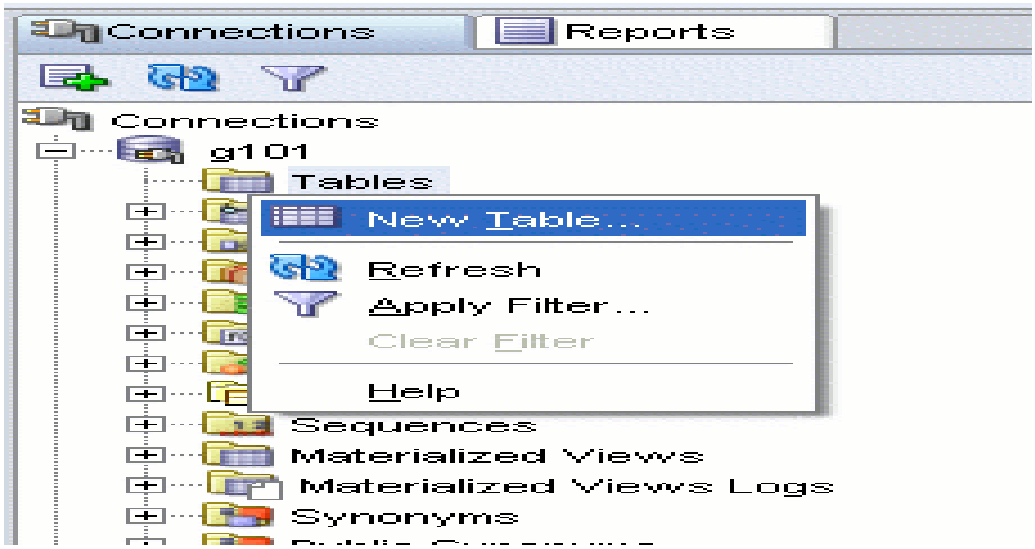

Figura 4.3.3 Pulsar “New Table...”.

Nos despliega un menú, en el cual, se selecciona la opción “New Table”, devolviéndonos la siguiente pantalla, figura 4.3.4.

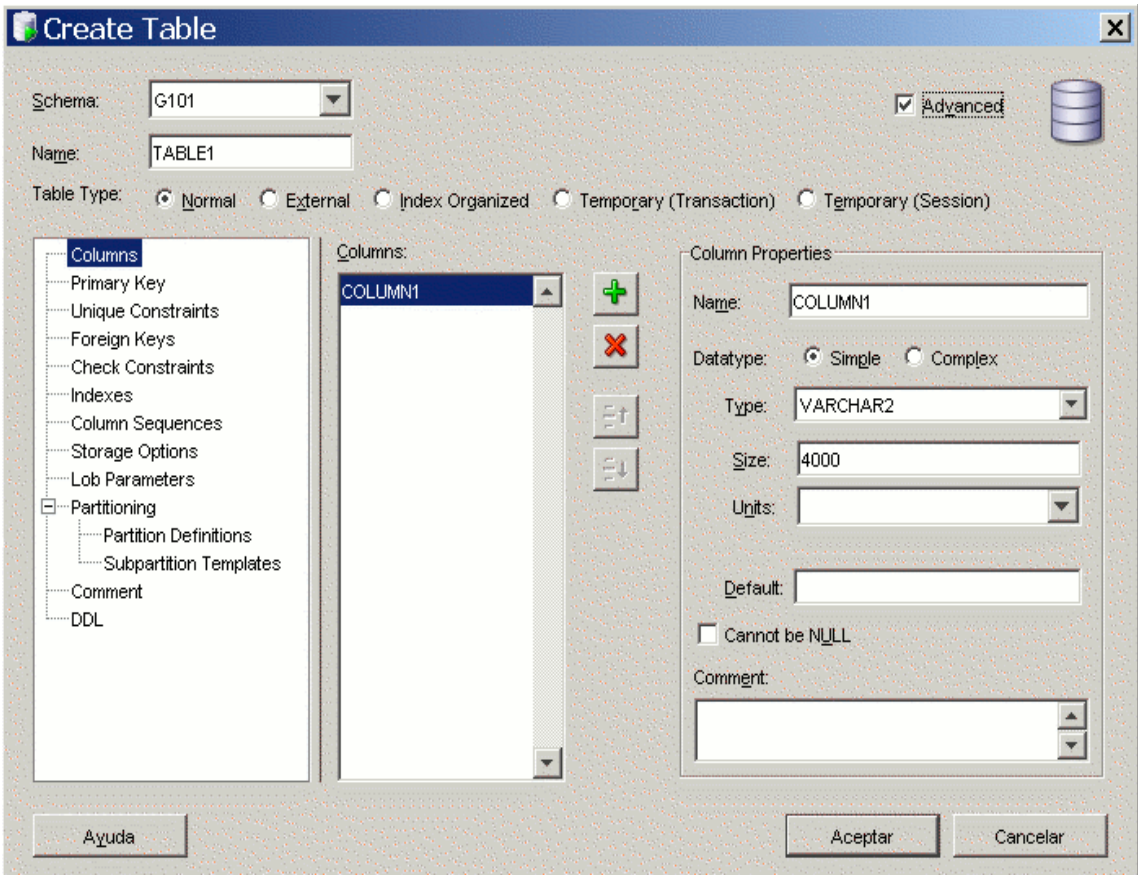

Figura 4.3.4 Creación de tablas.

Modificación de la definición de una tabla.

1. Seleccionar la tabla (doble-click sobre su icono)
2. Elegir la pestaña “Columns”
3. Pulsar sobre el icono “Edit”

También se puede hacer pulsando con el botón derecho sobre el icono de la tabla que se quiere modificar y eligiendo la opción “Edit...”

Inserción de tuplas.

Para insertar tuplas en una tabla, se selecciona la tabla, y se pulsa la pestaña “Data”, figura 4.3.5.

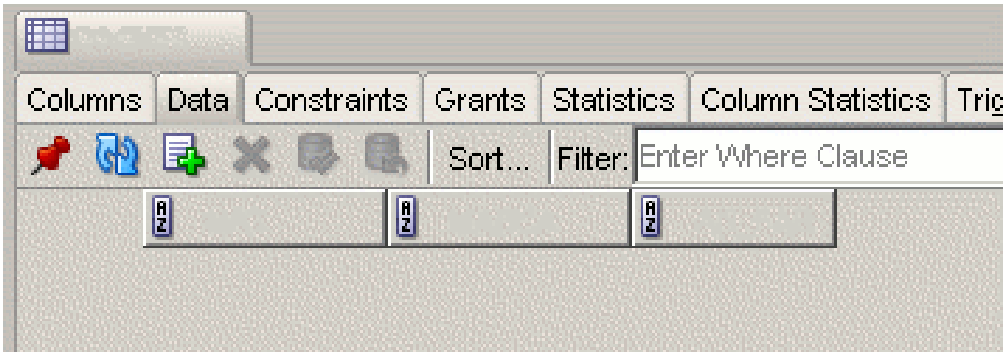

Figura 4.3.5 Inserción de tuplas “data”.

Para introducir una nueva tupla se pulsa sobre el icono, y se escriben los valores de cada atributo, figura 4.3.6.

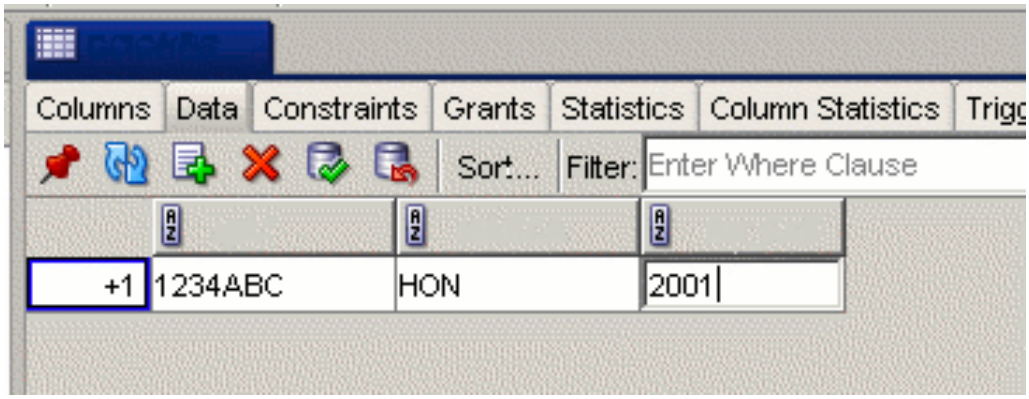

Figura 4.3.6 Inserción de tuplas “Icono”.

Para grabar la(s) tupla(s) en la tabla se pulsa el icono (commit).

El icono permite borrar una tupla.

Para grabar la(s) tupla(s) en la tabla se pulsa el icono (commit).

El icono permite borrar una tupla.

El icono permite fijar la pestaña de la tabla actual de manera que si se selecciona otra tabla en el navegador de objetos se abrirá otra pestaña nueva y no se reutilizará la pestaña fijada.

Generación de código SQL.

Generación del código SQL de un único objeto (tabla, secuencia, procedimiento, disparador), figura 4.3.7.

Pulsar botón derecho sobre el objeto y seleccionar “Export DDL”

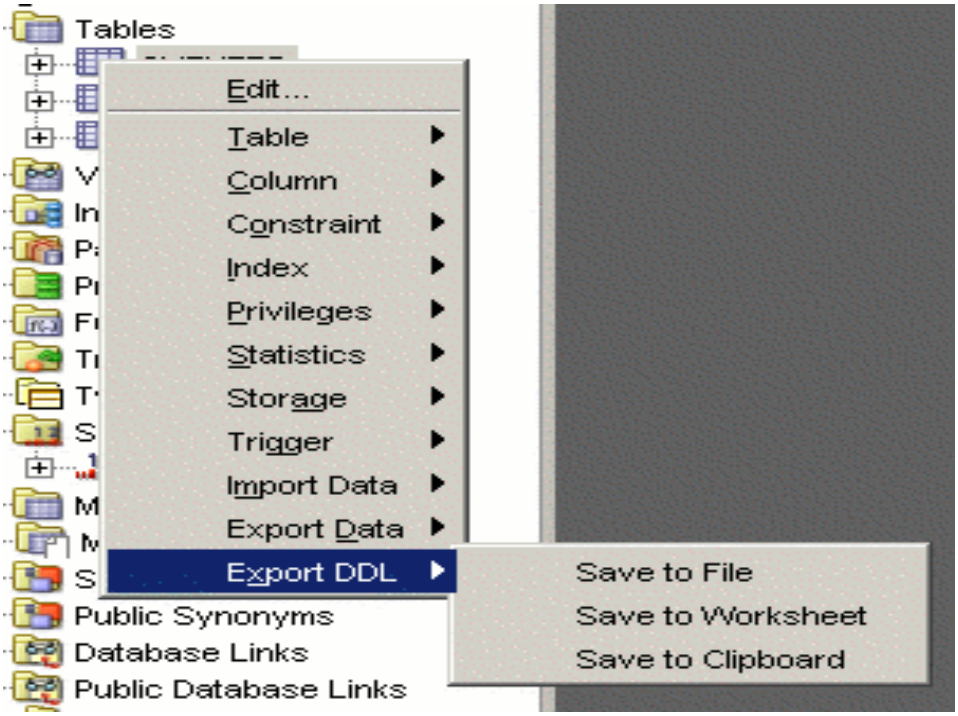

Figura 4.3.7 Generación del Código de Toda una Conexión.

Ir a “Tools” --> “Export DDL (and data)”, figura 4.3.8.

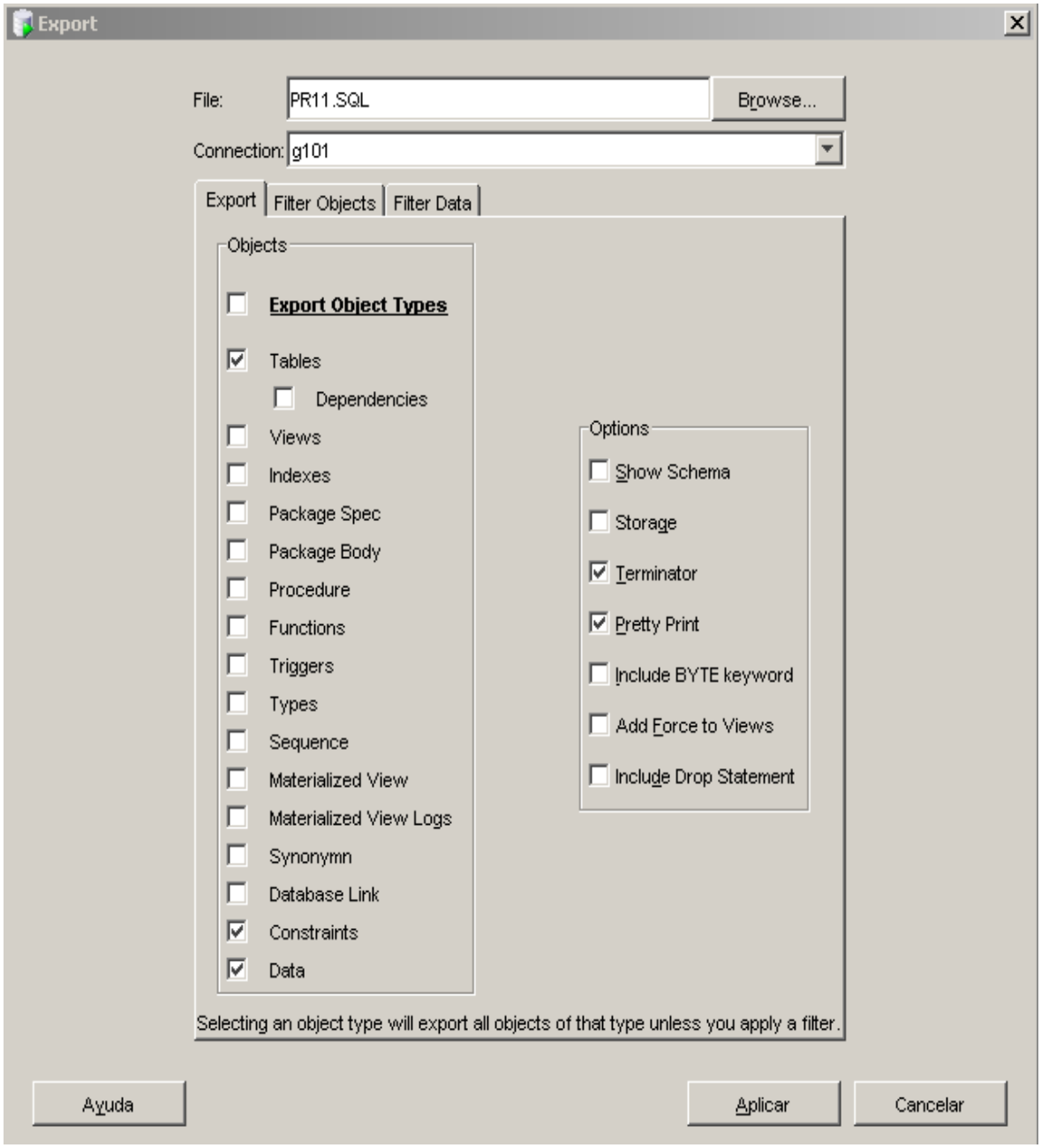

Figura 4.3.8 “Export DDL (and data)”.

Creación y edición de secuencias.

Para crear una nueva secuencia se pulsa el botón derecho sobre icono “Sequences” de la conexión, figura 4.3.9.

Figura 4.3.9 "Creación y edición de secuencias".

Elegir "New sequence...", figura 4.3.10.

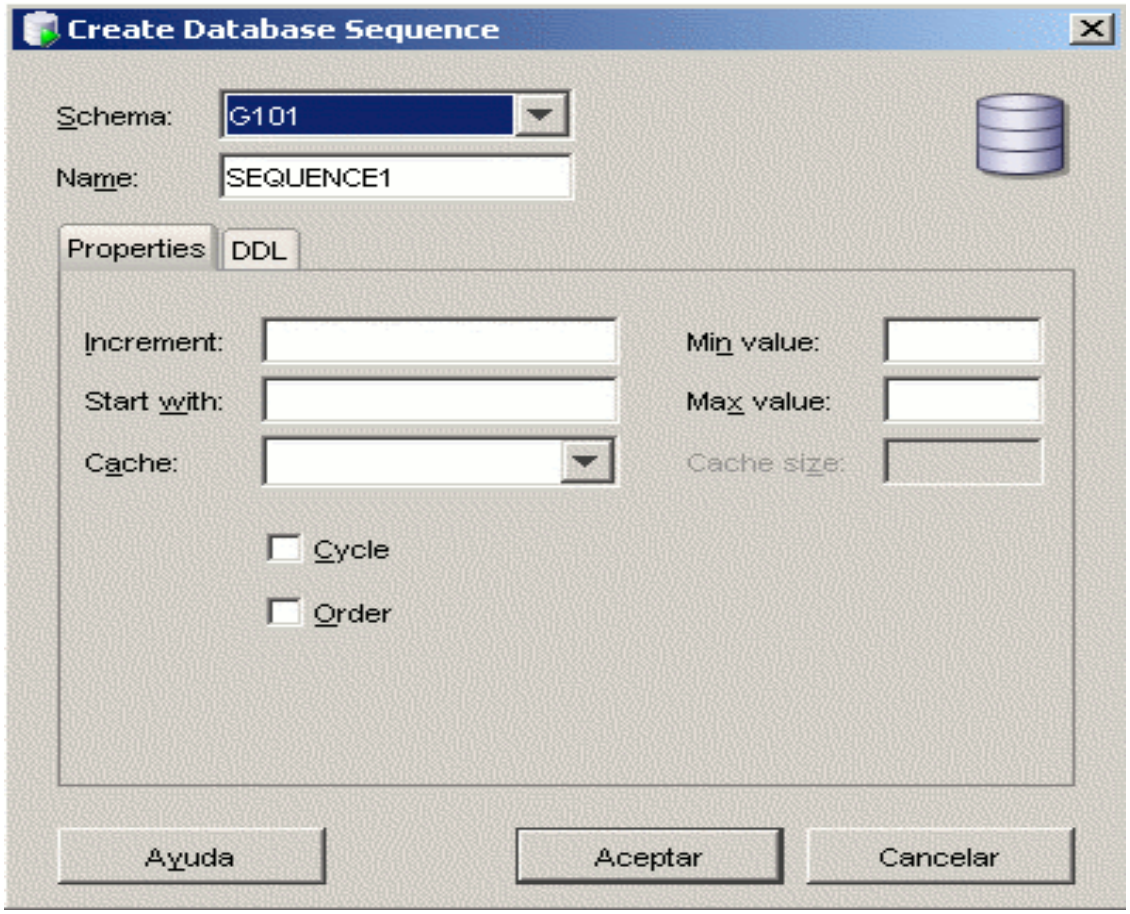

Figura 4.3.10 New Sequence.

Una vez creada la secuencia, se puede acceder a su definición y valor actual haciendo doble-click sobre su icono, figura 4.3.11.

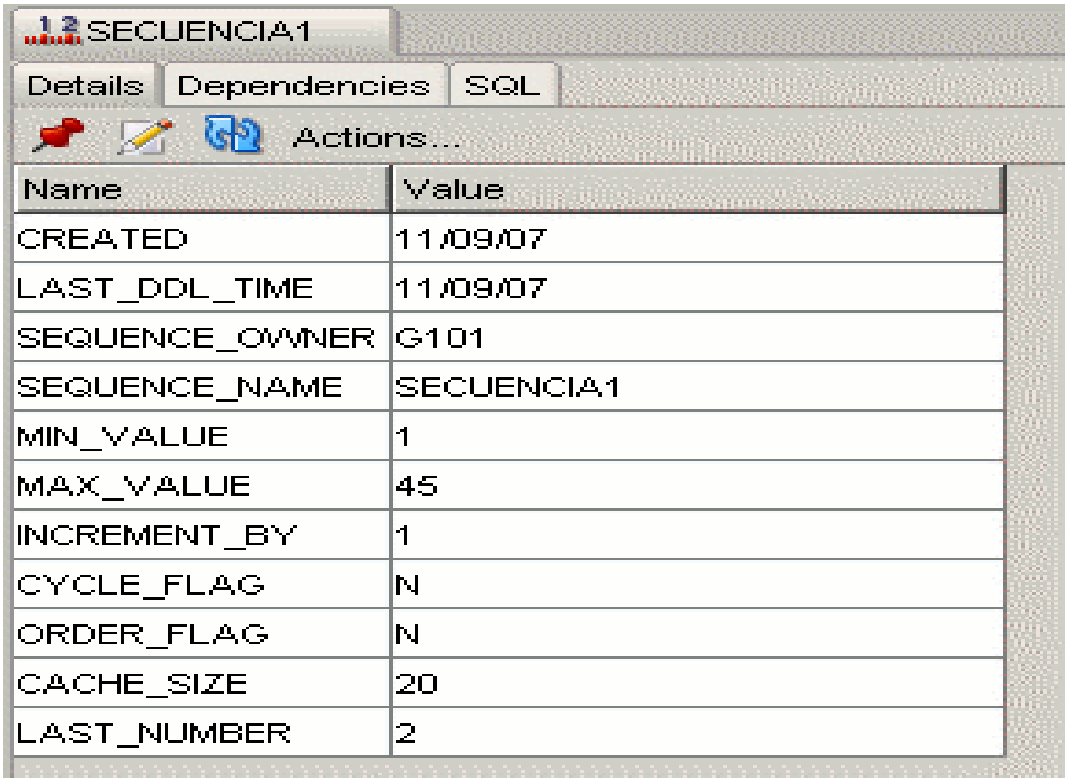

Figura 4.3.11 Definición y valor actual.

Como cualquier otro objeto, para modificar la definición de una secuencia se puede optar por:

1. Seleccionar la secuencia (doble-click sobre su icono) y elegir la pestaña “Details” y pulsar sobre el icono “Edit”
2. Pulsar con el botón derecho sobre el icono de la secuencia que se quiere modificar y elegir la opción “Edit...”, figura 4.3.12.

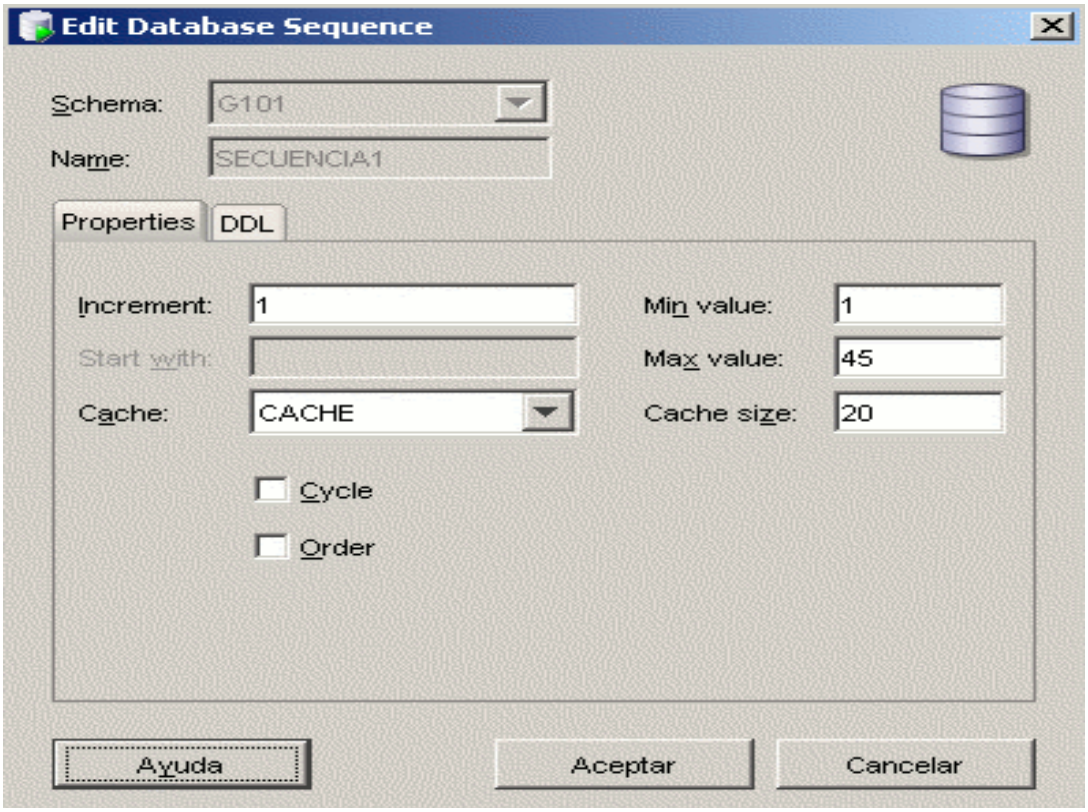

Figura 4.3.12 Definición y valor actual.

Creación, compilación y ejecución de funciones y procedimientos.

Para crear una función o procedimiento se pulsa con botón derecho sobre el icono "Functions" o "Procedures" del navegador de objetos y se elige la opción "New function" o "New Procedure" respectivamente, figura 4.3.13.

Figura 4.3.13 New Function, Procedure.

Se introduce el nombre del procedimiento o función, los nombres de los parámetros, sus tipos de datos, el modo del parámetro y los valores por defecto. Para el caso de las funciones también hay que especificar el tipo del resultado de la función (parámetro <Return>), figura 4.3.14 y figura 4.3.15.

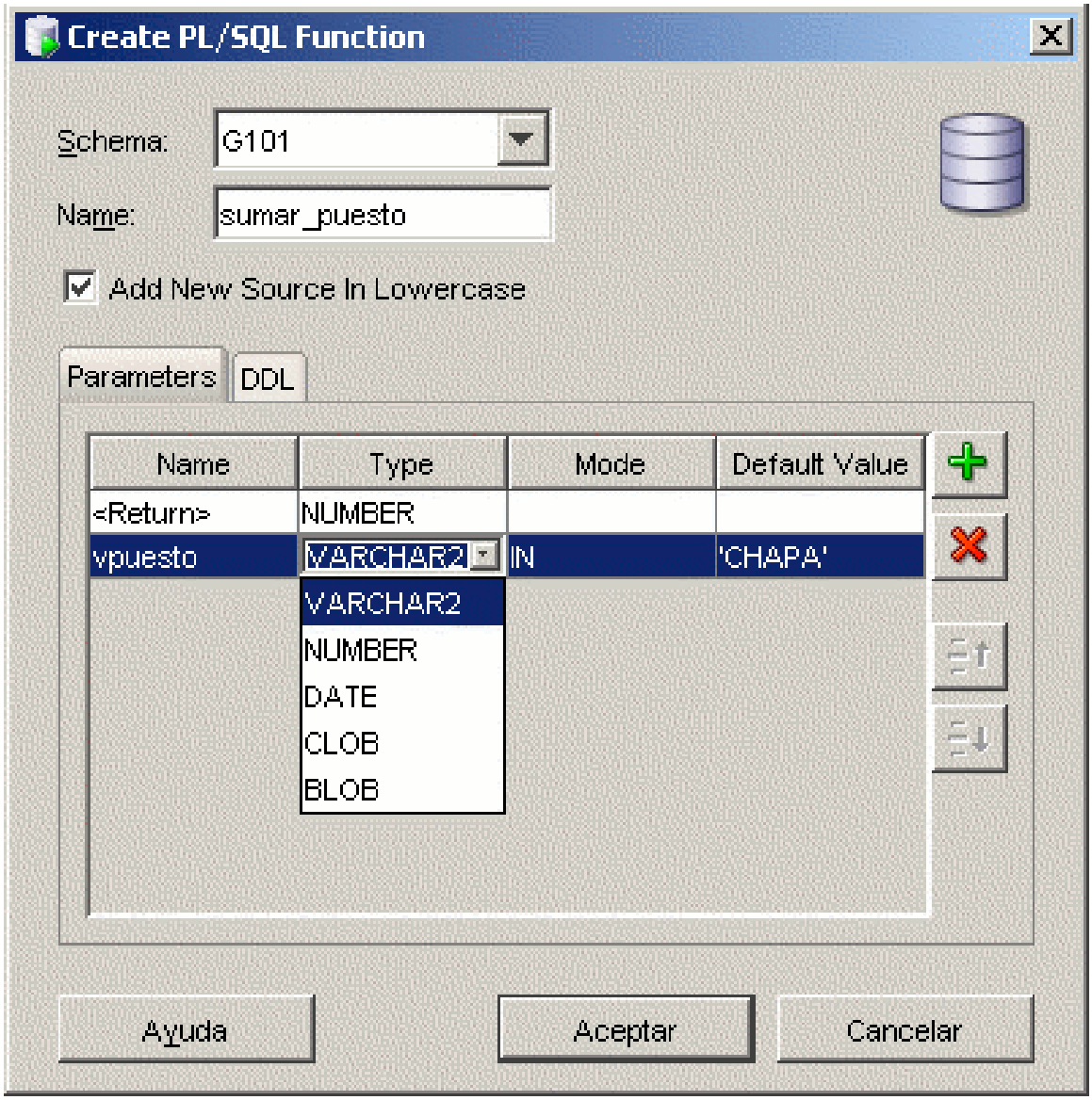

Figura 4.3.14 Parametros New Function Procedure.

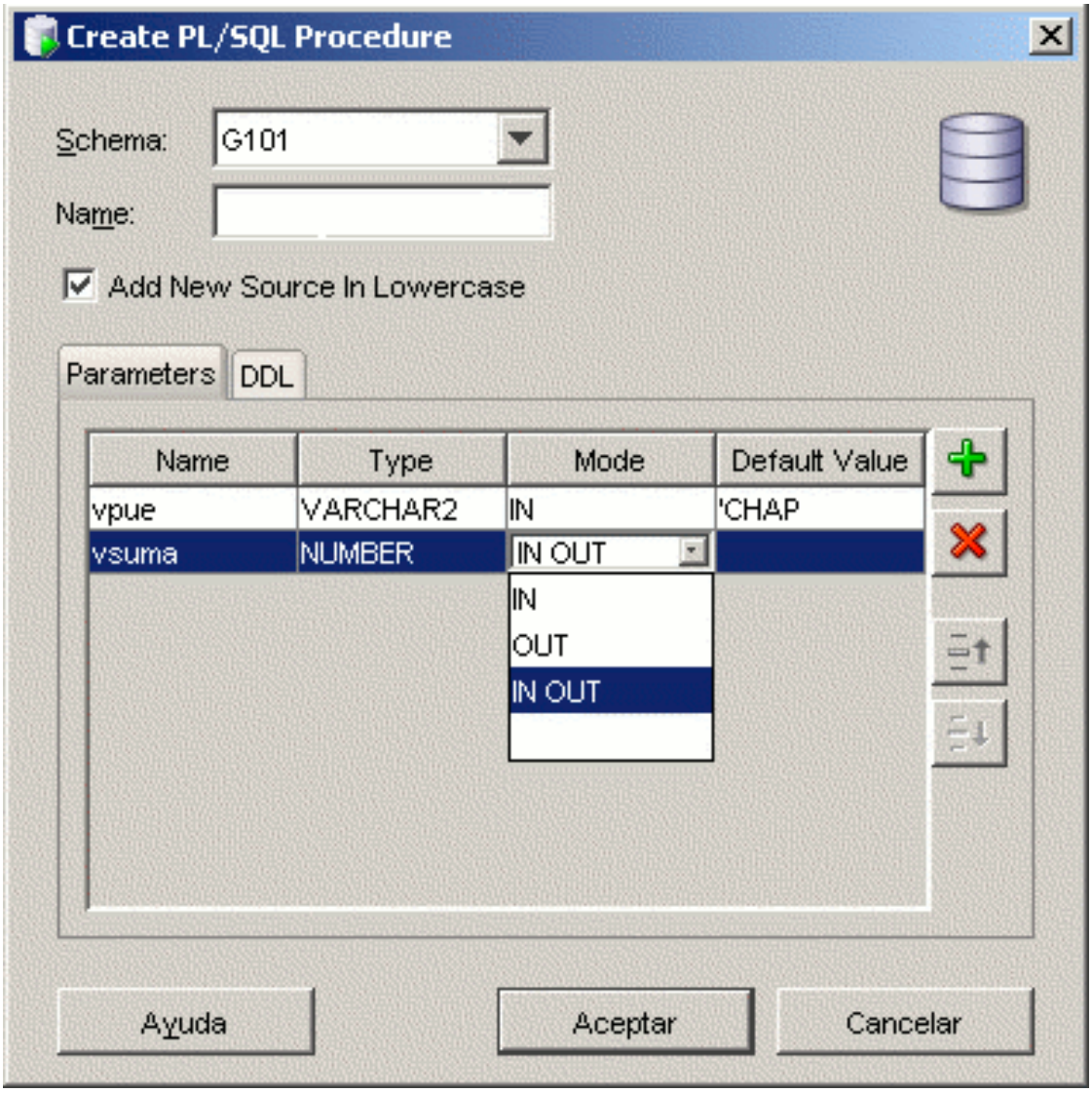

Figura 4.3.15 Parametros New Function , Procedure.

El asistente abre una pestaña de edición con el código generado para la función o el procedimiento con la cabecera especificada y el cuerpo vacío, figura 4.3.16 y figura 4.3.17.

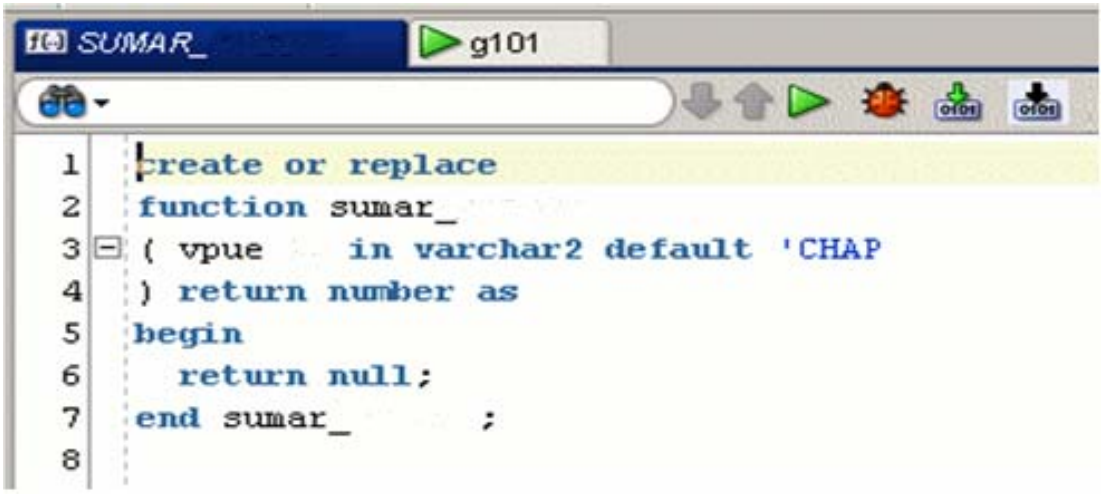

Figura 4.3.16 Pestaña de edición.

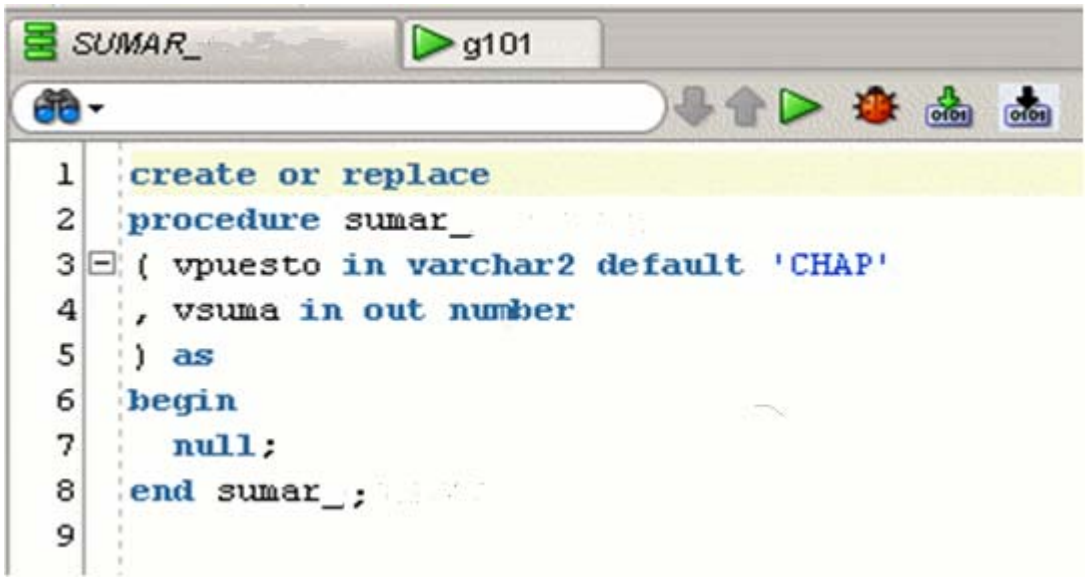

Figura 4.3.17 Pestaña de edición.

Para compilar se pulsa el icono. También se compila automáticamente cuando se almacena el procedimiento o función en la base de datos (icono). Los errores y warnings aparecen en el panel “Log” en la pestaña “Compiler”. Junto a la palabra error o warning se indica entre paréntesis la línea y la columna en la que se ha producido el error. Las sentencias erróneas aparecen subrayadas en rojo y los warnings subrayados en amarillo en la ventana de edición, figura 4.3.18.

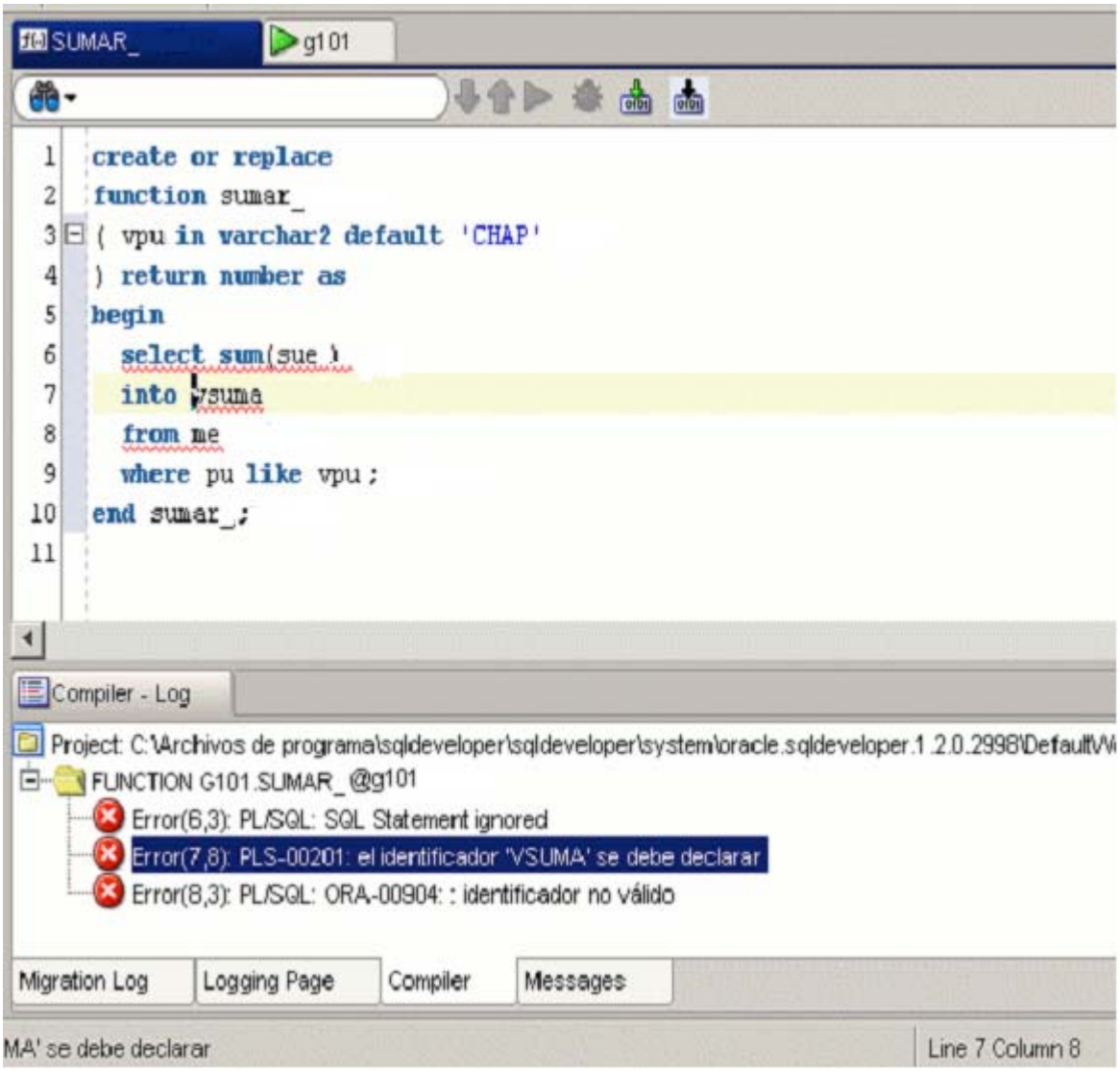

Figura 4.3.18 Errores y Warnings

Para ejecutar un procedimiento o función se pulsa el icono de la ventana de edición o se elige la opción “Run...” que aparece tras pulsar con el botón derecho sobre el icono de la función o procedimiento en el navegador de objetos, figura 4.3.19.

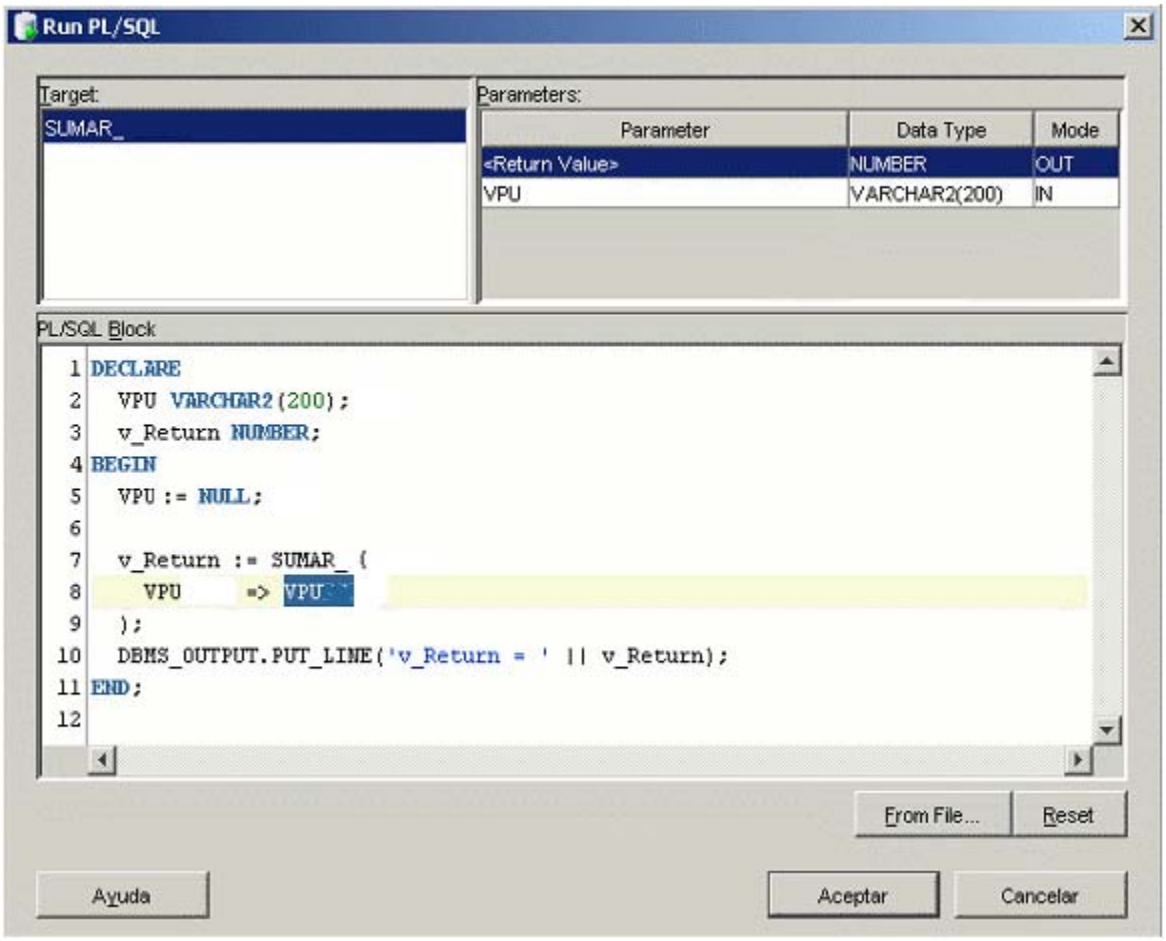

Figura 4.3.19 Ejecución de procedimiento o función.

Para poder ejecutar una función o procedimiento, SQL Developer crea un bloque con las variables necesarias para pasar los parámetros en la llamada a la función o procedimiento, debiéndose sustituir los valores por defecto predefinidos, por el valor actual que se le quiere dar al parámetro para la ejecución:

Inicialmente: VARIABLE => VARIABLE

Se sustituye por: VARIABLE => valor_actual, figura 4.3.20.

```
4  BEGIN
5    VPU:= NULL;
6
7    v_Return := SUMAR_(
8      VPU => 'CHAP'
9    );
10   DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
```

Figura 4.3.20 Bloque con variables.

Tras pulsar el botón “Aceptar”, el bloque que contiene la llamada a la función o el procedimiento se ejecuta y se muestran los resultados en el panel “Log” en la pestaña “Running”, figura 4.3.21.

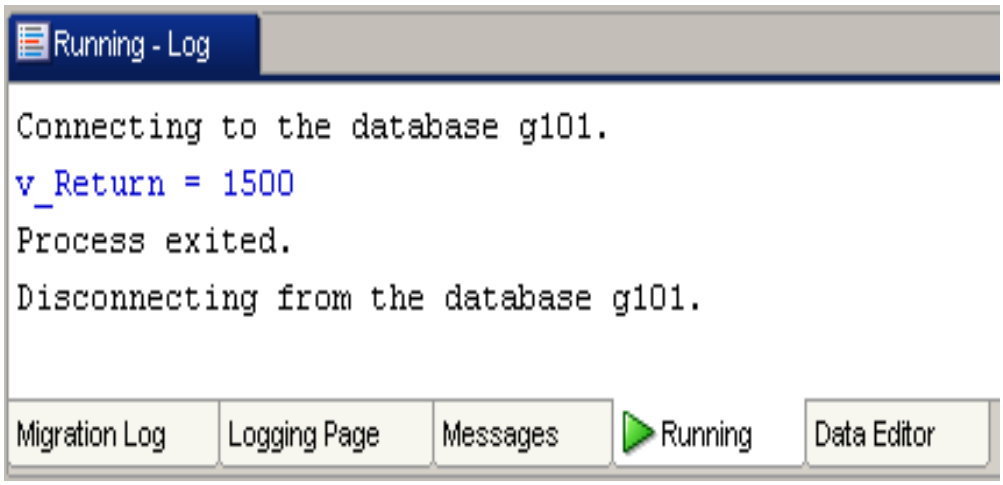

Figura 4.3.21 Resultados en panel log.

Depuración de funciones y procedimientos.

Para poder depurar es necesario que el usuario tenga los privilegios:

- DEBUG ANY PROCEDURE
- DEBUG CONNECT SESSION

El Administrador ya tiene estos privilegios asignados. Para asignarlos en una instalación local de Oracle Express debe consultar el apartado “Asignación de Privilegios y Roles” más abajo.

Para comenzar la depuración, en la ventana de edición del procedimiento o función (botón derecho sobre el icono del objeto y elegir “Edit...”) introducir los puntos de ruptura deseados dentro del cuerpo del procedimiento o función, como mínimo uno para que la ejecución del depurador se interrumpa y se pueda avanzar paso a paso viendo los valores de las distintas variables. Los puntos de ruptura se especifican pulsando con el ratón sobre el número de la línea donde se quiere introducir (el número de línea se sustituye por un círculo rojo), figura 4.3.22.

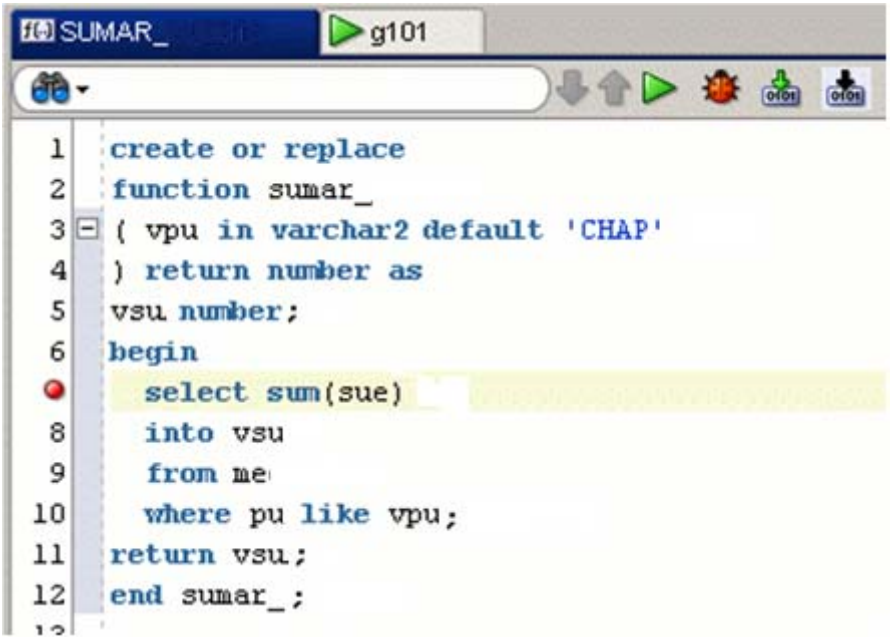

Figura 4.3.22 Puntos de ruptura.

Antes de iniciar la depuración es necesario compilar el procedimiento o función de manera especial para que pueda ser depurado. Esto se realiza pulsando el icono .

Para iniciar el depurador, se pulsa sobre el icono de la ventana de edición del procedimiento o función. A continuación se mostrará una ventana similar a la que aparece cuando se ejecuta un procedimiento o función, en la que hay que establecer los valores actuales de los parámetros como se describió

anteriormente. A continuación el flujo de control (indicado por una flecha roja) se detiene en el primer punto de ruptura establecido, pudiéndose ver los valores de las distintas variables en las pestañas “Data” y “Watches” del depurador, figura 4.3.23.

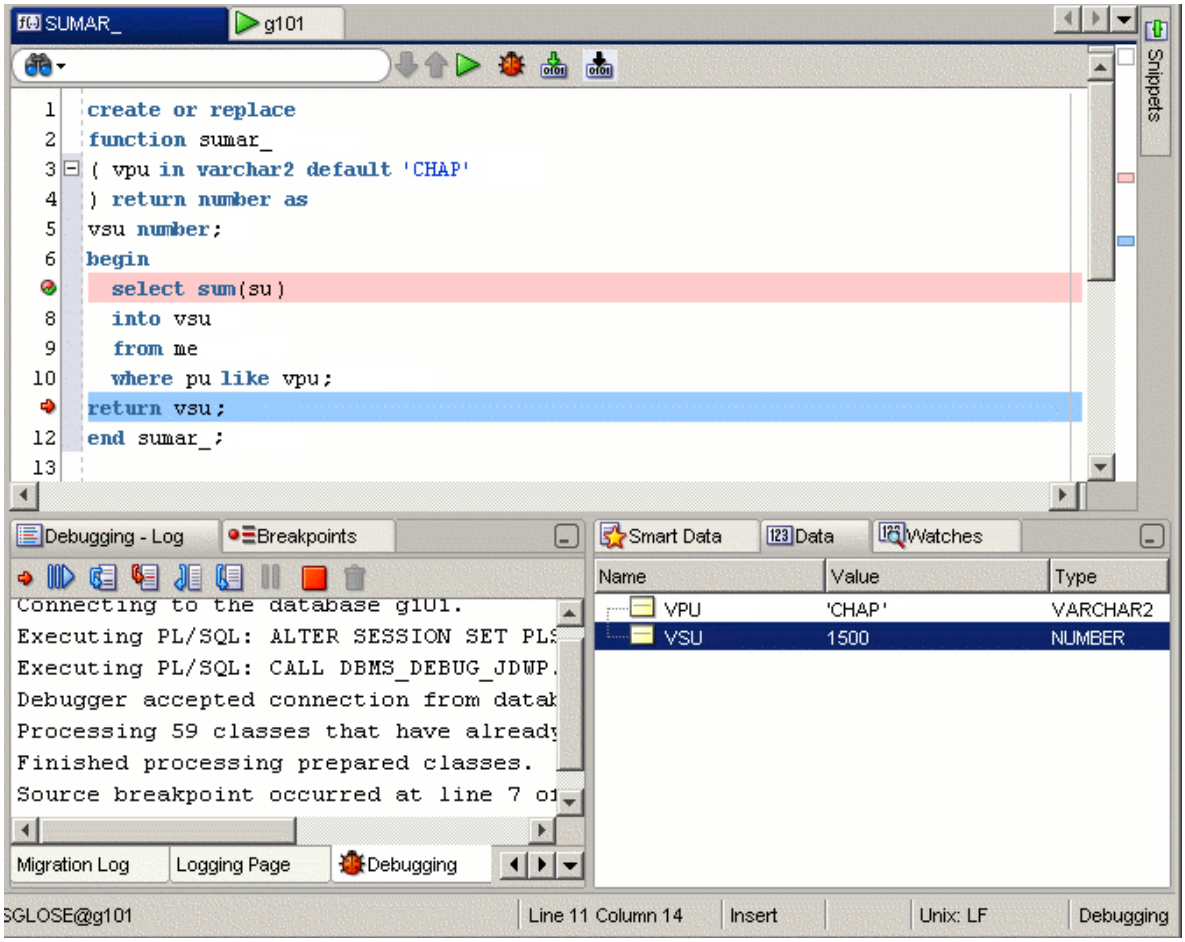

Figura 4.3.23 Inicio del depurador.

Las posibles acciones del depurador se encuentran en el menú “Debug”, las más típicas son:

- Avanzar sin entrar: F8
- Avanzar entrando: F7
- Avanzar hasta el cursor: F4
- Avanzar hasta el próximo punto de ruptura: F9

Creación y compilación de disparadores.

Para crear un disparador se pulsa con botón derecho sobre el icono “Triggers” del navegador de objetos y se elige la opción “New Trigger...”, figura 4.3.24.

Figura 4.3.24 New Trigger.

Se introduce:

- El nombre del disparador
- El tipo de disparador
- La tabla asociada al disparador
- Si el disparador es de sentencia (“Statement Level”) o de tupla (“Row Level”)
- El momento del disparo (“Before” o “After”)
- Los eventos de disparo (“Insert”, “Delete” o “Update”)
- Para el caso del evento “Update” se pueden especificar sobre qué columnas debe ser la actualización
- Para el caso de disparadores de tupla, se puede especificar una condición para la cláusula “When” y cambiar en “Referencing” el nombre de las variables de referencia de tupla por defecto (old y new), figura 4.3.25.

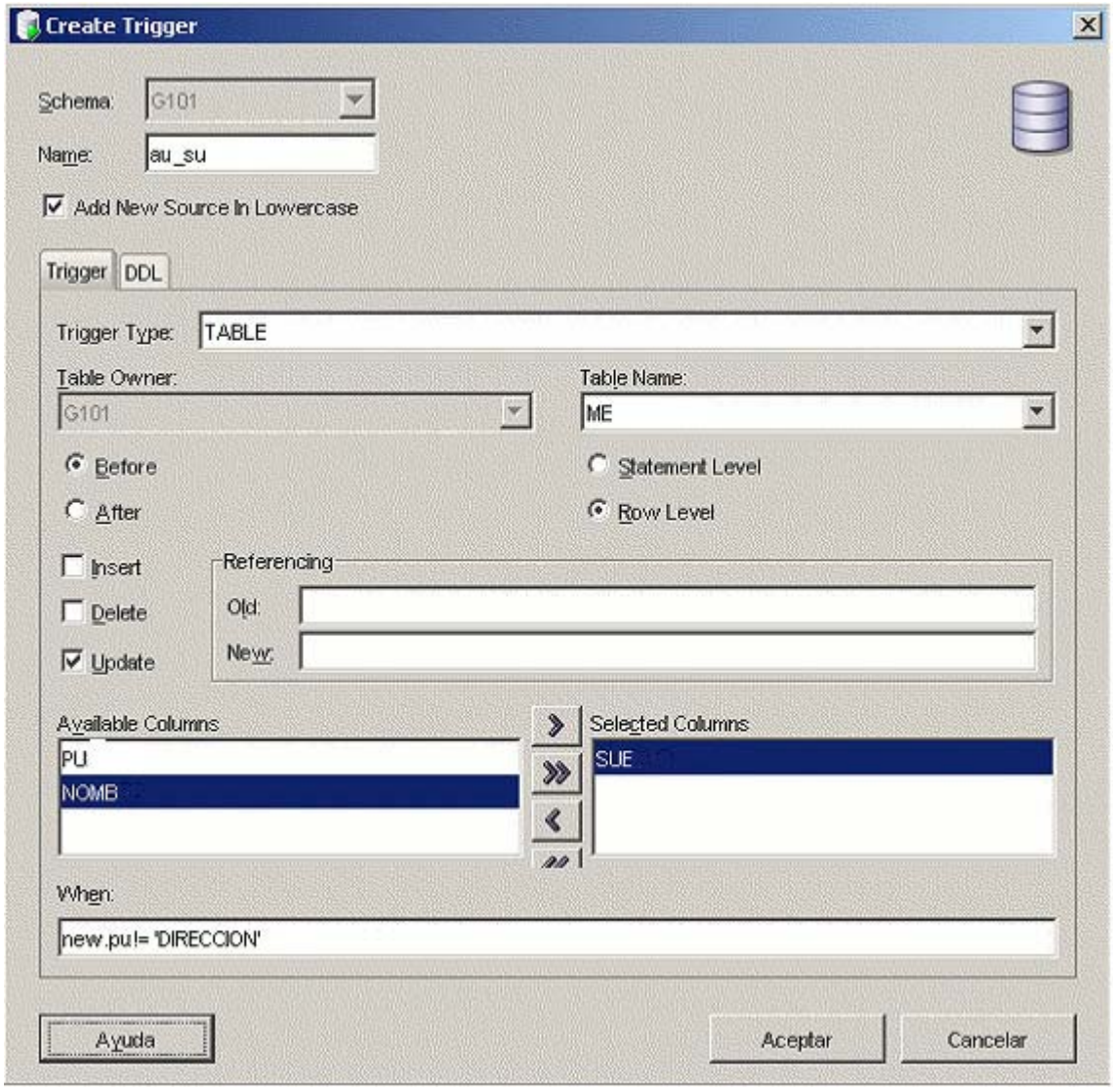

Figura 4.3.25 Create Trigger.

El asistente abre una pestaña de edición con el código generado para el disparador con la cabecera especificada y el cuerpo vacío, figura 4.3.26.

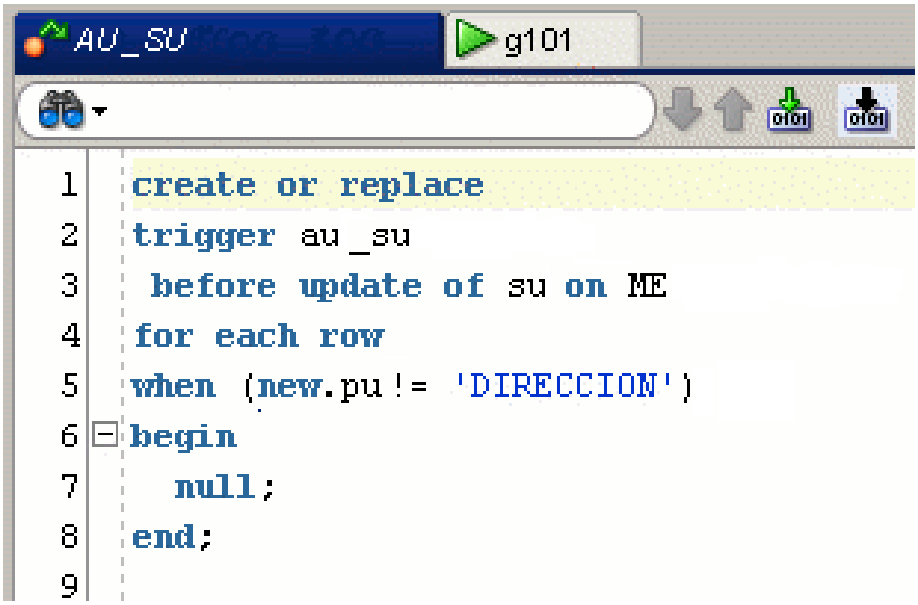

Figura 4.3.26 Código de Trigger.

Para compilar se pulsa el icono. También se compila automáticamente cuando se almacena el disparador en la base de datos (icono). Al igual que para el caso de los procedimientos y las funciones, los errores y warnings aparecen en el panel "Log" en la pestaña "Compiler". Junto a la palabra error o warning se indica entre paréntesis la línea y la columna en la que se ha producido el error. Las sentencias erróneas aparecen subrayadas en rojo y los warnings subrayados en amarillo en la ventana de edición del disparador.

Depuración de disparadores.

La versión actual de SQL Developer no permite la depuración mediante traza del código de los disparadores. La manera tradicional de trazar los disparadores consiste en mostrar mensajes en pantalla.

Para mostrar un mensaje desde un bloque PL/SQL, por ejemplo desde el cuerpo de un disparador, se utiliza la función:

DBMS_OUTPUT.PUT_LINE (cadena);

Por ejemplo: DBMS_OUTPUT.PUT_LINE ('El valor de var es: ' || var);

Para que los mensajes aparezcan por pantalla es necesario activar la salida del servidor (SERVEROUTPUT). Desde SQL*PLUS se realiza mediante la sentencia:
SQL> SET SERVEROUTPUT ON
En SQL Developer, la salida del servidor se establece activando el icono que se encuentra en la pestaña "DBMS Output" de las ventanas de edición de SQL (SQL Worksheet).

Asignación de privilegios y roles.

Para asignar privilegios hay crear una conexión en SQL Developer para el administrador “SYSTEM”, abrir la categoría “Other Users” en el navegador de objetos y elegir la opción “Edit User” al pulsar con el botón derecho sobre el usuario al que se le quieren dar los privilegios, figura 4.3.27.

Figura 4.3.27 Asignación de Privilegios.

En la pestaña “System Privileges” activar la casilla “Granted” para los privilegios que se quieren conceder al usuario y pulsar aplicar, figura 4.3.28.

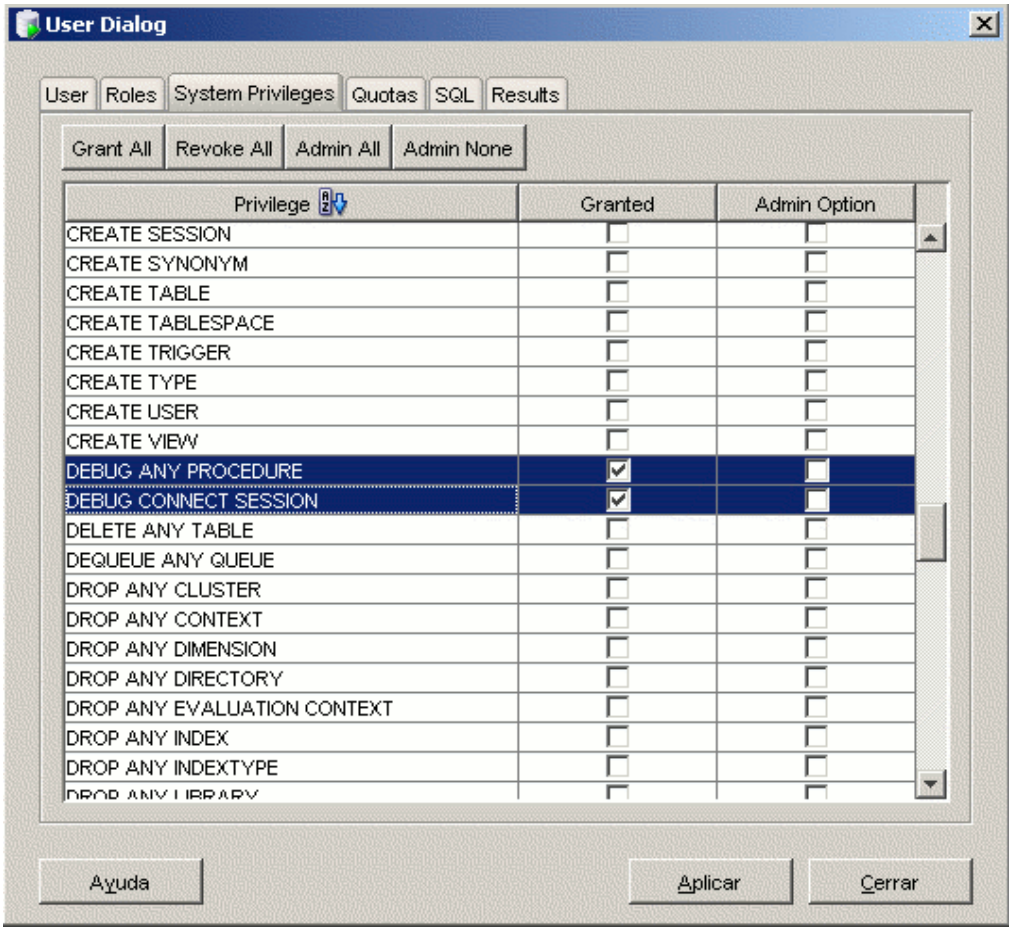

Figura 4.3.28 Concesión de privilegios.

Para poder conceder roles, el usuario debe tener previamente concedido por el administrador el privilegio “CREATE ROLE”.

4.4 Integración, pruebas y mantenimiento.

Integración.

Es el proceso de construir un sistema de software combinando componentes a una unidad de trabajo. La integración de componentes debe proceder de una manera ordenada, siguiendo una secuencia de función en función.

Esto permite que las capacidades operacionales del software sean mostradas tempranamente, dando la confianza de que el proyecto está progresando de manera satisfactoria.

El sistema de mediación se integró a partir del módulo de recolección, dado que este tiene la tarea de establecer la conectividad con las plataformas de conmutación celular (transferencia AFT), para después enviar bloques de CDRs hacia el módulo de parseo y su posterior extracción de archivos entre equipos FTP (File Transfer Protocol).

El módulo de parseo recibe los bloques de CDRs para procesarlos y por medio del método remoto de invocación RMI (Java Remote Method Invocation), consulta la información que requiere del módulo de catálogos, para almacenar los CDRs en la base de datos y de esta forma puedan estar disponibles para otras aplicaciones.

La conexión con la base de datos hacia el módulo de catálogos se realiza mediante un lenguaje de programación java de esta forma se manipulan los datos de un módulo a otro y se filtra a su vez la información necesaria a las diferentes aplicaciones clientes.

Pruebas.

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Las pruebas de software se integran dentro de las diferentes fases del ciclo del software dentro de la ingeniería de software. Para determinar el nivel de calidad se deben efectuar las pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema.

Para el caso en particular al Sistema de Mediación, las pruebas que se aplicaron para verificar el correcto funcionamiento fueron:

- La prueba de la caja blanca. Se introdujeron varios CDRs de prueba para determinar el comportamiento del código del programa en cada una de las etapas o módulos del sistema de mediación.
- La prueba de caja negra. Se verificó que no existiera errores en establecer la conexión con las plataformas de conmutación celular, para poder asegurar el correcto ingreso de bloques de CDRs de entrada, así como verificar que los CDRs de salida tuvieran la estructura adecuada y se almacenaran de forma correcta en la base de datos.
- La prueba de integración. Se probó cada módulo por separado y también al momento de unir dos módulos para verificar su perfecto ensamblaje.
- La prueba de validación. Se probó el sistema completamente ensamblado, verificando que los resultados arrojados en fechas, datos, tamaños, etc., fueran los correctos y cumplieran con los requerimientos y objetivos que se plantearon al inicio del desarrollo.
- La prueba de stress. Como el sistema de mediación va a procesar grandes volúmenes de CDRs diarios, es necesaria esta prueba, para verificar el comportamiento del sistema al procesar 4 millones de CDRs en 2 horas.

Tipos de pruebas.

Prueba de la caja blanca.

Su objetivo principal es asegurar el correcto funcionamiento de las interfaces, o flujo de datos entre componentes.

Su diseño está fuertemente ligado al código fuente. El verificador escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa para cerciorarse de que se devuelven los valores de salida adecuados.

Aunque las pruebas de caja blanca son aplicables a varios niveles (unidad, integración y sistema), habitualmente se aplican a las unidades del software. Se

comprueban los flujos de ejecución dentro de cada unidad (función, clase, modulo, etc.), pero también pueden verificar los flujos entre unidades durante la integración, e incluso entre subsistemas, durante las pruebas del sistema.

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se intenta usar todas las estructuras de datos internas.

El principal factor que se debe considerar al inicio de las pruebas es el tamaño del módulo a probar, se debe considerar si el tamaño del módulo permitirá probar adecuadamente toda su funcionalidad de manera sencilla y rápida. También es importante separar los módulos de acuerdo a su funcionalidad, si los módulos son muy grandes y contienen muchas funcionalidades, estos se volverán más complejos de probar y al encontrar algún error será más difícil ubicar la funcionalidad defectuosa y corregirla.

En el sistema, cada componente debe ser probado en su totalidad (óvalos) y también sus interfaces o comunicaciones con los demás componentes (flechas), figura 4.4.1.

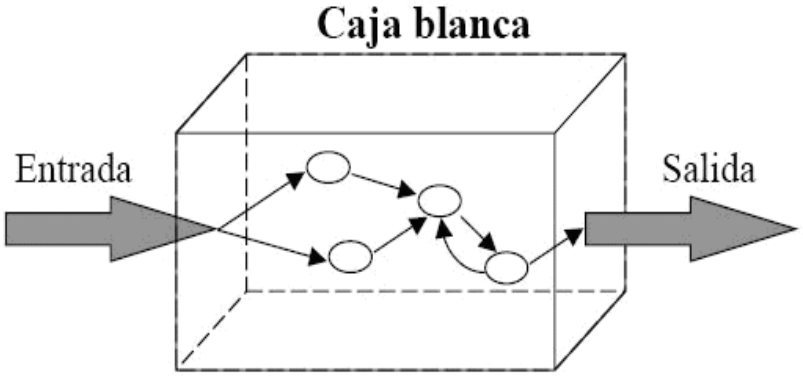

Figura 4.4.1 Diagrama de la caja blanca.

Prueba de caja negra.

Es la prueba que estudia al sistema desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Solo interesa su forma de interactuar con el medio que le rodea, entendiendo ¿Qué es lo que hace?, pero sin dar importancia en ¿Cómo lo hace?. Por lo que debe de estar bien definidas sus entradas y salidas, es decir, su interfaz, pero no es importante conocer los detalles internos de su funcionamiento.
Esta prueba se centra principalmente en los requisitos funcionales del software, permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos, figura 4.4.2.

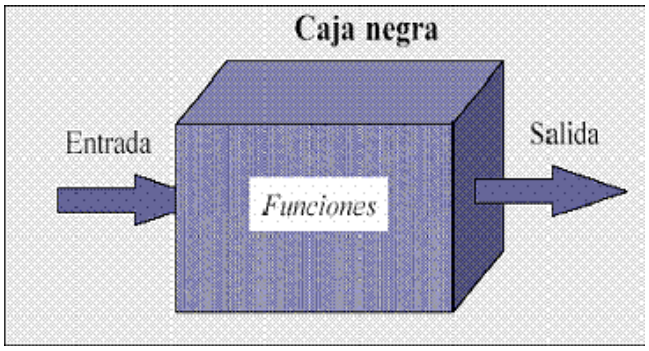

Figura 4.4.2 Diagrama de la caja negra.

Prueba de Integración.

Es una técnica sistemática para construir la estructura del programa mientras al mismo tiempo, se lleva a cabo pruebas para detectar errores asociados con la interacción. Consiste en verificar que un gran conjunto de partes de software funcionan juntos.
Se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como un conjunto.

Estas pruebas cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. Estas pruebas se pueden plantear desde un punto de vista estructural o funcional.

Los tipos fundamentales de integración son los siguientes:

- Integración incremental: Se combina el siguiente módulo que se debe probar con el conjunto de módulos que ya están probados y se va incrementando progresivamente el número de componentes a probar.
- Integración no incremental: Se prueba cada módulo por separado y posteriormente se integran todos de una vez realizando las pruebas pertinentes.

Pruebas de regresión.

El objetivo es comprobar que los cambios sobre un componente del sistema, no introducen un comportamiento no deseado o errores adicionales en otros componentes no modificados.

Son cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, incluidos por cambios recientemente realizados en partes de la aplicación que anteriormente no eran propensas a errores. Esto implica que el error se presenta como consecuencia inesperada del cambio en el sistema.

Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falto de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido, o simplemente una consecuencia del rediseño de la aplicación.

Estas pruebas se deben llevar a cabo cada vez que se hace un cambio en el sistema, tanto para corregir un error como para realizar una mejora. No es suficiente probar solo los componentes modificados o añadidos, o las funciones que en ellos se realizan, sino que también es necesario controlar que las

modificaciones no produzcan efectos negativos sobre el mismo u otros componentes.

Las pruebas de regresión pueden incluir:

- La repetición de los casos de pruebas que se han realizado anteriormente y están directamente relacionados con la parte del sistema modificado.
- La revisión de los procedimientos manuales preparados antes del cambio, para asegurar que permanecen correctamente.
- La obtención impresa del diccionario de datos de forma que se compruebe que los elementos de datos que han sufrido algún cambio son correctos.

Pruebas de validación.

Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento de los requisitos especificados. Estas empiezan tras la culminación de las pruebas de integración, cuando se han ejercitado los componentes individuales. Se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz.

La prueba se concentra en las acciones visibles para el usuario y en la salida del sistema que este puede reconocer.

La validación se define de una forma simple, cuando el software funciona de tal manera que satisface las expectativas razonables del cliente. La validación del software se logra mediante una serie de pruebas que demuestren que se cumplen los requisitos. Un plan de prueba delinea la clase de pruebas que se aplicarán y un procedimiento de prueba define los casos de prueba específicos.

La revisión de la configuración, es un elemento importante del proceso de validación. Su objetivo, es asegurar que todos los elementos de la configuración del software se hayan desarrollado apropiadamente, estén catalogados y tengan el detalle suficiente para reforzar la fase de soporte del ciclo de vida del software.

Prueba de stress.

Consiste en la simulación de grandes cargas de trabajo para observar de qué forma se comporta la aplicación ante situaciones de uso intenso.

- Prueba de stress de componentes: Se aíslan los servicios y componentes que conforman el sistema, se infieren los métodos de navegación, de funcionamiento y de interfaz de estos servicios y componentes y se crea un cliente de prueba que llame a dichos métodos.

  Para aquellos métodos que tienen acceso a un servidor de base de datos o a cualquier otro componente, se puede crear un cliente que proporcione datos simulados en el formato previsto.
  La idea es forzar cada componente de forma aislada, más de lo que la aplicación podría experimentar en condiciones normales.
- Prueba de stress de integración: Después de forzar cada componente individual, se deberá someter a una situación de stress a toda la aplicación con todos sus componentes y servicios.
  Estas pruebas están íntimamente relacionadas con las interacciones con otras estructuras de datos, procesos y servicios tanto de los componentes internos como de otros servicios externos de la aplicación.

Prueba alfa.

Se lleva a cabo por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tenga validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

Prueba beta.

Se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.
Como resultado de los problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda clase de clientes.

Mantenimiento.

El mantenimiento de software es el proceso de mejorar y optimizar el software, así como de remediar los defectos, con el fin de asegurar que se continúen desempeñando las funciones deseadas. Esta fase involucra cambios al software en orden de corregir defectos y dependencias encontradas durante su uso, tanto en la adición de nueva funcionalidad para mejorar la usabilidad y en la aplicabilidad del software.
A la entrega del software al usuario final, dentro del contrato o el acuerdo, también se puede establecer como realizar el mantenimiento. El mantenimiento puede ser variado ya que depende de las necesidades de cada usuario, por esta razón se pueden tener varios tipos de mantenimiento.

Tipos de mantenimiento.

Mantenimiento preventivo.

Consiste en usar el producto de software antes de su entrega, para detectar y corregir fallos latentes antes de que se conviertan en fallos efectivos. El mantenimiento preventivo proporciona mejoras para los usuarios, mejora de la documentación del sistema y la recodificación para mejorar el rendimiento del software, su mantenibilidad u otros atributos. Se incluyen sentencias para comprobar la validez de los datos de entrada, reestructuración de los programas para aumentar su legibilidad o incluir nuevos comentarios.

Este tipo de mantenimiento se diferencia del resto de los tipos de mantenimiento, en que, mientras que el resto se produce generalmente tras una petición por parte del cliente o del usuario final, el preventivo se produce tras un estudio de posibilidades de mejora en los diferentes módulos del sistema.

Uno de sus principales objetivos en este tipo de mantenimiento es evitar los fallos antes de que estos ocurran.

Mantenimiento correctivo.

Tiene por objeto corregir defectos y fallas detectados en la operación del producto de software después de su entrega. Un defecto en un sistema es una característica del sistema con el potencial de provocar un fallo. Un fallo se produce cuando el comportamiento del sistema difiere con respecto al comportamiento definido en la especificación. Los fallos en un sistema de software pueden ser de:

- Procesamiento (salidas incorrectas de un programa).
- Rendimiento (tiempo de respuesta demasiado alto).
- Programación (inconsistencias en el diseño).

- Documentación (inconsistencias entre la funcionalidad de un programa y el manual del usuario).

Se puede dividir el mantenimiento correctivo en:

- Programado: Es el que se efectúa cuando la falla no es urgente, difiriendo de la ejecución para el momento más oportuno y con la reparación más adecuada.
- Crítico: Es cuando la falla es urgente, de la manera más directa en el menor tiempo posible y con la mejor preparación que permitan las circunstancias.
- Normal: Se aplica a los sistemas que al fallar no afectan la seguridad ni la producción, por lo que su reparación puede ser programada y resuelta con los recursos normales.
- Emergente: Se produce cuando las fallas ponen en peligro la seguridad o integridad de la producción.

Mantenimiento perfectivo.

Consiste en modificar el producto de software, después de su entrega, para mejorar su rendimiento o su mantenimiento. Este mantenimiento proporciona mejoras para los usuarios, mejora de la documentación del sistema y la recodificación para mejorar el rendimiento del software u otros atributos.
Los requisitos pueden cambiar con el tiempo para ajustarse a nuevas necesidades o para mejorar las prestaciones actuales, por lo que pueden ir desde una pequeña modificación en un módulo, hasta la adición de nuevos módulos. Este mantenimiento no está únicamente enfocado a mejorar técnicamente una solución, sino que también incluye un proceso continuo de optimización a nivel funcional y de procesos. Este mantenimiento se enfoca en:

- La optimización constante del rendimiento de las aplicaciones mediante análisis técnicos.

- La adaptación de las aplicaciones a las nuevas necesidades del cliente en función de los análisis funcionales.
- La detección de posibles puntos a mejorar en el diseño y uso de las bases de datos mediante el análisis de la misma.

Por lo tanto el principal objetivo del mantenimiento perfectivo es llevar a cabo las tareas y procesos necesarios para identificar aquellos puntos susceptibles de mejora. Este tipo de mantenimiento se divide en:

- Mantenimiento de ampliación (incorpora nuevas funcionalidades).
- Mantenimiento de eficiencia (mejora la eficiencia de ejecución).

Este mantenimiento es derivado de nuevos requisitos en cuanto a funcionalidad y como consecuencia de una posible optimización de rendimiento. Estas actividades están motivadas por cambios introducidos por el usuario, más allá del alcance y objetivos iniciales del sistema.

Mantenimiento tecnológico.

Este tipo de mantenimiento está pensado para aquellas acciones que están relacionadas con el software instalado. Está pensado para dar una cobertura tecnológica a las aplicaciones instaladas en el cliente.

Cuando se instala una aplicación, se instala para un sistema operativo y una base de datos determinada. A lo largo del tiempo estás tecnologías cambian por parte del fabricante o por parte del cliente. Este cambio tecnológico no entra en el mantenimiento estándar.

Este mantenimiento consiste en la disponibilidad de descarga de nuevas actualizaciones específicas del software suministrado, con objeto de garantizar la compatibilidad ante el cliente.

Este mantenimiento permite que la aplicación siga funcionando correctamente cuando surjan en el mercado nuevas versiones de los productos que conforman el entorno tecnológico requerido por la aplicación.

Las nuevas versiones específicas de la aplicación, incorporan nuevas funcionalidades que optimizan y mejoran el uso de la aplicación con el nuevo entorno tecnológico del cliente, con objeto de aprovechar, en la medida de lo posible, las prestaciones que ofrecen las nuevas versiones del entorno tecnológico.

Este mantenimiento se suministra a:

- Sistemas operativos.
- Herramientas.
- Bases de datos.
- Servidores.

Mantenimiento adaptativo.

Consiste en la modificación de un programa debido a cambios en el entorno (hardware o software), en el cual se ejecuta. Estos cambios pueden afectar al sistema operativo, a la arquitectura física del sistema informático o al entorno de desarrollo del software (incorporación de nuevos elementos o herramientas). La envergadura del cambio necesario puede ser muy diferente, desde un pequeño retoque en la estructura de un módulo hasta tener que reescribir prácticamente todo el programa para su ejecución en un ambiente distribuido en una red.

Los cambios en el entorno del software pueden ser de dos clases:

- En el entorno de los datos.
- En el entorno de los procesos.

4.5 Generación de reportes

La información que se obtiene de los CDRs se puede revisar realizando consultas en la base de datos ya que el sistema no tiene un front-end y los archivos se

envían al área de facturación para su procesamiento, en estos reportes se puede observar muchos campos que nos sirven para tener un control sobre el tipo de llamada, la forma de pago, duración de la llamada etc. A continuación se describen los campos más comunes al realizar una consulta.

- Tipo: Muestra si la llamada corresponde a la propia red o si es de interconexión
- Dato: Informa si se enviaron voz, datos o multimedia
- Número que llama: Número que realiza la llamada
- Número al que se llama: Número que recibe la llamada
- Zona: Lugar del país donde se realizó la llamada(Zona 1 norte, Zona 2 centro y Zona 3 sur)
- Forma de pago: Indica si fue en renta mensual o pospago

Se presenta los datos de una consulta.

TIPO	DATO	NUM. QUE LLAMA	NUM. AL QUE SE LLAMA	ZONA	FORMA DE PAGO
Outgoing	Voice	5585089671	7444851920	1	POSTPAID
Outgoing	Voice	5550300001	4777914431	2	POSTPAID
Outgoing	Voice	5550300001	4499749524	2	POSTPAID
Outgoing	Voice	5550300001	4491193288	2	POSTPAID
Outgoing	Voice	5550300001	4499155103	2	POSTPAID
Outgoing	Voice	5550300001	4499121578	2	POSTPAID
Outgoing	Voice	5550300001	4499145757	2	MONTHPAID
Incoming	Voice	5550300001	4777914431	2	POSTPAID
Outgoing	Voice	5550300001	4771006921	2	POSTPAID
Outgoing	MMS	5550300001	4771006921	2	POSTPAID
Outgoing	Voice	5550300001	4771006921	2	POSTPAID
Outgoing	Voice	5585077140	3334901761	3	POSTPAID
Outgoing	Voice	5519750121	3331900576	3	POSTPAID
Incoming	Voice	3331900576	5519750121	3	POSTPAID

TIPO	DATO	NUM. QUE LLAMA	NUM. AL QUE SE LLAMA	ZONA	FORMA DE PAGO
Outgoing	Voice	5519750121	3331900576	3	POSTPAID
Outgoing	Voice	5585077140	3338462145	3	POSTPAID
Outgoing	Voice	5585080288	3311724845	3	POSTPAID
Outgoing	Voice	5585077140	3336721642	3	POSTPAID
Outgoing	Voice	5585080288	3336788888	3	POSTPAID
Outgoing	Voice	5519750121	3331441008	3	POSTPAID
Outgoing	Voice	5514584531	3336027016	3	POSTPAID
Outgoing	Voice	5585077134	3333942593	3	POSTPAID
Outgoing	Voice	5591429718	3573840998	3	POSTPAID
Outgoing	Voice	5550300001	3336753319	3	POSTPAID
Outgoing	Voice	5585077134	3338121867	3	MONTHPAID
Outgoing	Voice	5585077124	3338256401	3	MONTHPAID
Incoming	Voice	5514584531	3334815141	3	POSTPAID
Outgoing	Voice	5585077134	3338121867	3	POSTPAID
Outgoing	Voice	5585080288	3919161846	3	POSTPAID
Outgoing	Voice	5550300001	3337321956	3	POSTPAID
Outgoing	Voice	5519750121	3757584304	3	POSTPAID
Outgoing	Voice	5514584531	3477884834	3	POSTPAID
Outgoing	Voice	5585077124	3336984114	3	POSTPAID
Outgoing	Voice	5519750121	3757588009	3	POSTPAID
Incoming	Voice	5585077134	3336458439	3	POSTPAID
Outgoing	Voice	5519750121	3757588009	3	POSTPAID
Outgoing	Voice	5550300001	3336044533	3	POSTPAID
Outgoing	Voice	5519750121	3757584304	3	POSTPAID
Outgoing	Voice	5585080288	9535410599	3	POSTPAID
Outgoing	Voice	5585077134	3335863589	3	MONTHPAID
Incoming	Voice	5585080288	3333427811	3	POSTPAID
Outgoing	Voice	5519750121	3757584304	3	POSTPAID

TIPO	DATO	NUM. QUE LLAMA	NUM. AL QUE SE LLAMA	ZONA	FORMA DE PAGO
Outgoing	Voice	5591429718	3333662767	3	POSTPAID
Outgoing	Voice	5519750121	3336007707	3	POSTPAID
Outgoing	Voice	5550300001	3310574657	3	POSTPAID
Outgoing	Voice	5585077134	3336075446	3	POSTPAID
Outgoing	Voice	3338394107	3311442184	3	MONTHPAID
Outgoing	Voice	5591429718	3331829030	3	MONTHPAID
Incoming	Voice	5519750121	3336325139	3	MONTHPAID
Outgoing	MMS	5591429718	3335700198	3	POSTPAID

Ahora se muestra el reporte que genera el área de facturación sobre las llamadas realizadas en el mes de enero de telefonía celular.

Facturación enero:

- No. de abonados: 17728

Llamadas nacionales e internacionales:	802296
Llamadas locales:	2723325
Llamadas prepago:	2053
Llamadas entrantes:	1215536
Llamadas salientes de otros operadores:	159464
Total de llamadas procesadas:	4902674

- Tamaño de la base de datos generada: 415 MB
- Tiempo aproximado del proceso de facturación: 20 minutos

CONCLUSIONES

- Con un Sistema de Mediación que homologue y complemente la información obtenida de los CDRs se reduce el tiempo de procesamiento de la facturación lo que posibilita a la empresa cumplir con sus acuerdos con otros operadores en tiempo y forma.

- Al tener información de CDRs en una sola base de datos se pueden realizar reportes y/o consultas en un tiempo bastante breve, pudiendo inclusive cruzar información con otras bases de datos y obtener información valiosa y estratégica para la compañía, en un tiempo record inimaginable anteriormente para la compañía.

- Al contar con una plataforma que nos permite obtener la información de manera oportuna, el periodo medio de pago es óptimo porque en lugar se contabilizan las llamadas el mismo día que se realizan y cuando el cliente recibe su facturación se le cobran todas las llamadas del periodo evitando quejas y reclamaciones.

- Con el nuevo Sistema de Mediación se alimenta a los diversos sistemas de estadísticas de manera inmediata, lo cual proporcionara herramientas seguras y oportunas para la toma de decisiones así como para las estrategias de marketing y estudios de mercado, los cuales ayudaran al crecimiento de la compañía.

- Con el nuevo sistema de Mediación se tiene un mejor control para la auditoria de la propia central, para verificar su funcionamiento, rendimiento, como opera, así como la eficiencia de su trabajo, y con ello poder prever alguna falla en su operación o poder detectarla a tiempo. Ya que si llega a tener una falla, podría costar varios millones de pesos y esto conlleva a

diferentes consecuencias, como sería el retraso del monitoreo de las llamadas, perdida de información, etc.

- Como ahora el sistema de Mediación es totalmente automático en todos sus procesos, el transporte y almacenamiento de los datos ya no se ve comprometido por la posibilidad de que existan errores u omisiones de parte del personal operativo; además de que dichas tareas ya son fácilmente rastreables y supervisables.

- Con el nuevo sistema de mediación se ha logrado tener un perfecto control en todas las etapas de procesamiento de CDRs, desde la recolección en las centrales telefónicas, logrando el correcto transporte hasta el almacenamiento en la base de datos, facilitando el manejo de los CDRs, para los diferentes fines que se requieran.

- Como ahora se cuenta con un formato estándar de CDRs, los registros de la base de datos son estándares y cuentan con un bajo dinamismo fortaleciendo el proceso y simplificando las tareas de seguimiento y documentación de las operaciones.

- Para la construcción del sistema de mediación se utilizó una metodología orientada a objetos, esto proporciona la ventaja de poder modificar el sistema cuando alguna de sus partes se vuelva obsoleta o agregarle una nueva aplicación cuando se generen nuevas necesidades.

- Gracias a los conocimientos y habilidades adquiridas en la Facultad de Ingeniería así como durante la experiencia en la vida laboral, pudimos satisfacer los requerimientos y necesidades de la compañía con la correcta elección de las herramientas.

REFERENCIAS

1. Bases de datos relacionales.,Celma Gimenez, M.Casa Mayor Rodenas, J.C.Mota Herranz, L.,Pearson Educacion, S.A..Madrid, 2003.
2. Beck, K.; Cunningham, W. A laboratory for teaching object-oriented thinking. Proc. of Object-Oriented Programming Systems, Languages and Applications 1989 (OOPSLA '89). SIGPLAN Notices, Vol. 24, No. 10, October 89, pp 1-6.
3. Booch, Grady. Object-Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994. Addison-Wesley, 1996.
4. Booch, G.; Rumbaugh, J.; Jacobson, I.: Unified Modeling Language User Guide, Addison-Wesley, 1998
5. Coad, Peter; Yourdon, E. Object-Oriented Analysis. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.
6. Coad, Peter; Yourdon, E. Object-Oriented Design. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.
7. Eriksson, H.E.; Penker, M.: UML Toolkit, IEEE, John Wiley and sons, 1998.
8. Jacobson, Ivar. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley Publishing Co., Reading, Mass, 1992.
9. Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process, Addison-Wesley, Reading, Mass., 1999.
10. Lee, R.M.; Tepfenhart, W.M.: UML and C++, Prentice Hall, 1997.
11. Martin, James; Odell, James. Object-Oriented Analysis and Design. Prentice Hall, Englewood Cliffs, NJ, 1992.
12. Martin, James; Odell, James. Object-Oriented Methods: A Foundation. Prentice Hall, Englewood Cliffs, NJ, 1995.
13. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorenson, W. Object-Oriented Modelling and Design. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991
14. Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual, Addison-Wesley, Reading, Mass., 1999.
15. Shlaer, S.; Mellor, S.J. Object-Oriented System Analysis: Modelling the World in Data. Prentice Hall, Englewood Cliffs, NJ, 1988.
16. Stevens, P.; Pooley, R. Utilizaci¢n de UML. Addison Wesley, Madrid, 2002.
17. Wirfs-Brock, R.; Wilkerson, B.; Wiener, L. Designing Object-Oriented

18. Software. Prentice Hall, Englewood Cliffs, NJ, 1990.
19. http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos
20. http://www.uaem.mx/posgrado/mcruz/cursos/miic/informix3.ppt
21. http://es.wikipedia.org/wiki/Oracle
22. http://gplsi.dlsi.ua.es/~slujan/asp/Introduccion_a_ASP.htm
23. http://en.wikipedia.org/wiki/Sybase
24. http://algebrarelacional.awardspace.com/Algebra%20Relacional.htm
25. http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_bases_de_datos
26. http://alvherre.cl/pgsql/modBasico/node6.html
27. http://www.mitecnologico.com/Main/CardinalidadEntidadesRelacion
28. http://es.scribd.com/doc/4527025/CARDINALIDAD
29. http://es.wikipedia.org/wiki/PHP
30. http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29
31. http://es.scribd.com/doc/30772869/Sistema-de-Mediacion
32. http://www.slideshare.net/e1da4/mtodos-poo
33. http://www.rodolfoquispe.org/blog/que-es-la-programacion-orientada-a-objetos.php
34. http://www.google.com.mx/images?sourceid=navclient&hl=es&ie=UTF8&rlz=1T4GFRC_esMX202MX203&q=bases+de+datos+relacionales&oi=image_result_group&sa=X
35. http://usuarios.multimania.es/cursosgbd/UD4.htm
36. http://www.trabajoline.com.ar/SAP/CursoABAP/index.asp
37. http://www.programacion.com/articulo/estructuras_de_oracle_89
38. http://www.programacionweb.net/foros/mensaje/?num=16941
39. http://132.248.67.65:8991/F/-/?func=find-b-0&local_base=TES01
40. http://132.248.67.65:8991/F/-/?func=find-b-0&local_base=TES01

41. http://www.scribd.com/doc/210452/Arquitectura-de-Software-Adrian-Lasso
42. http://es.wikipedia.org/wiki/Anotaci%C3%B3n_Java
43. http://upcommons.upc.edu/handle/123456789/152332