DESARROLLO DE INTERFAZ OPENSMARTFLEX – ENERGIS: MODULOS DE PETICIONES Y CONSUMOS

DAVID ARIAS VILLALOBOS
JULIO CESAR LEDESMA ROMÁN
ANDRÉS FABIÁN SÁNCHEZ CIFUENTES

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE INFORMACIÓN
PROGRAMA DE INGENIERÍA INFORMÁTICA
SANTIAGO DE CALI
2007

DESARROLLO DE INTERFAZ OPENSMARTFLEX – ENERGIS: MODULOS DE PETICIONES Y CONSUMOS

DAVID ARIAS VILLALOBOS
JULIO CESAR LEDESMA ROMÁN
ANDRÉS FABIÁN SÁNCHEZ CIFUENTES

Proyecto de pasantía para optar por el título de Ingeniero en Informática

Director académico
RICARDO LLANO VALENCIA
Ingeniero de Sistemas.

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE INFORMACIÓN
PROGRAMA DE INGENIERÍA INFORMÁTICA
SANTIAGO DE CALI
2007

Nota de aceptación:

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al titulo de Ingeniero en Informática.

RICARDO LLANO VALENCIA

Firma del jurado

Santiago de Cali, 31 de Enero de 2007.

AGRADECIMIENTOS

Agradecemos a todo el personal de Open Systems Ltda. En especial al gerente de gestión humana Hugo Ocampo por habernos dado la oportunidad de formar parte de la organización y a los ingenieros Ricardo Calixto y Rodolfo Bautista por facilitarnos toda la información y ayuda necesaria con una gran disponibilidad.

Al Ingeniero Ricardo Llano quien en su función de coordinador de la pasantía nos brindo una gran orientación, conocimientos en el tema y consejos, elementos sin los cuales no nos hubiera resultado posible el desarrollo de este trabajo.

Un especial reconocimiento a nuestros padres que nos apoyaron en los momentos difíciles de esta etapa tan importante de nuestra vida, nos dieron las fuerzas y el amor necesario para afrontar todos los obstáculos que se nos presentaron.

CONTENIDO

pág.

RESUMEN 13
INTRODUCCIÓN 14
1. DEFINICIÓN DEL PROBLEMA 16
1.1 ANTECEDENTES 16
1.2 FORMULACIÓN DEL PROBLEMA 16
2. JUSTIFICACIÓN 17
3. OBJETIVOS 18
3.1 OBJETIVO GENERAL 18
3.2 OBJETIVOS ESPECÍFICOS 18
4. MARCO DE REFERENCIA 19
4.1 MARCO TEÓRICO 19
4.1.1 Suite de desarrollo Oracle 19
4.1.1.1 Oracle Database Server 19
4.1.1.2 Oracle Forms 20
4.1.1.3 Oracle Reports 21
4.1.1.4 Oracle PL/SQL 21
4.1.2 Técnicas de rendimiento y ajuste para procesos SQL 22
4.2 MARCO CONCEPTUAL 24
4.2.1 Historia de Open Systems Internacional 24
4.2.2 Misión de Open Systems Internacional 25
4.2.3 Modelo de negocio de Open Systems Internacional 25
4.3 ESTADO ACTUAL 26
4.3.1 Modelo de integración con sistemas externos 26
5. METODOLOGÍA 29
5.1 PROPÓSITO DEL PROYECTO 30
5.2 ALCANCE DEL PROYECTO 30
6. DESCRIPCION GENERAL 31
6.1 DESCRIPCIÓN DE LA SITUACIÓN 31
6.2 MODULO DE CONSUMOS 32
6.2.1 Descripción General 32
6.2.2 Flujo de Negocio 33
6.2.3 Datos a Publicar 35
6.3 MODULO DE PETICIONES 36
6.3.1 Descripción General 36
6.3.2 Flujo de Negocio 37
6.3.3 Datos a Publicar 38
6.4 MÉTODOS DE INTEGRACIÓN 41

6.4.1 Publicación de información mediante entidades customizadas 42
6.4.2 Publicación de información mediante archivos planos 43
6.4.2.1 Definición de la estructura para el archivo de consumos 43
6.4.2.2 Definición de formato de archivo para peticiones 44
7. ALCANCE FUNCIONAL 47
7.1 GENERACIÓN DE CONSUMOS MEDIDOS 47
7.1.1 Definición de la estructura de datos 47
7.1.2 Generación asistida de consumos medidos 48
7.1.3 Generación programada de consumos medidos 49
7.1.4 Implementación del API 50
7.1.5 Generación del Reporte Cantidad de unidades de Consumo 50
7.2 GENERACIÓN DE NOVEDADES 50
7.2.1 Definición de la estructura de datos 51
7.2.2 Configuración de las novedades a publicar 52
7.2.3 Generación asistida de Novedades 53
7.2.4 Generación programada de Novedades 54
7.2.5 Implementación del API 55
7.2.6 Generación del Reporte de Novedades 55
7.3 GENERACIÓN DE ARCHIVOS PLANOS 55
7.3.1 Generación asistida de archivos planos de consumos medidos 57
7.3.1.1 Mapeo de datos para la extracción de consumos medidos 57
7.3.1.2 Estructura del archivo plano de consumos medidos 57
7.3.2 Generación programada de archivos planos de consumos medidos 58
7.3.3 Generación asistida de archivos planos de Novedades 59
7.3.3.1 Mapeo de datos para la extracción de Novedades 59
7.3.3.2 Estructura del archivo plano de novedades 60
7.3.4 Generación programada de archivos planos de novedades 61
8. ARQUITECTURA DE LA SOLUCIÓN 63
8.1 ARQUITECTURA DE LA SOLUCIÓN 63
8.2 TRANSACCIONALIDAD 63
8.3 RESTRICCIONES 64
9. INTERFACES DE APLICACIÓN – API 65
9.1 API CONSULTA DE CONSUMOS MEDIDOS 65
9.2 API CONSULTA DE NOVEDADES 65
10. MODELO ENTIDAD RELACION 67
10.1 ENTIDADES DE BASES DE DATOS 67
10.1.1 Entidad nueva para consumos medidos 67
10.1.2 Entidades nuevas para novedades 68
10.1.3 Entidades nuevas para generación de archivos planos 69
10.2 DIAGRAMA ENTIDAD RELACION 70
10.2.1 Diagrama entidad relación de consumos medidos 70
10.2.2 Diagrama entidad relación de novedades 71
11. INTERFACES DE USUARIO 72
11.1 DEFINICION Y ALCANCE 72
11.1.1 Formas nuevas 72

11.1.1.1 Forma de generación de consumos medidos 72
11.1.1.2 Formas de generación de novedades 72
11.1.1.2.1 Forma de novedades a publicar 72
11.1.1.2.2 Forma de generación de novedades 73
11.1.1.3 Formas de generación de archivos planos 73
11.1.1.3.1 Formas de generación de archivos planos de consumos medidos 73
11.1.1.3.2 Forma de generación de archivos planos de novedades 73
11.1.1.3.3 Forma de configuración de reglas de mapeo 74
11.1.2 Reportes nuevos 74
11.1.2.1 Reporte de cantidad de unidades de consumo 74
11.1.2.2 Reporte de novedades 76
11.2 PROTOTIPO NAVEGABLE 78
11.2.1 Forma de generación de Consumos Medidos 78
11.2.2 Formas de generación de Novedades 80
11.2.2.1 Forma de Novedades a publicar 81
11.2.2.2 Forma de generación de Novedades 81
11.2.3 Formas de generación de archivos planos 82
11.2.3.1 Forma de generación de archivos planos de Consumos Medidos 82
11.2.3.2 Forma de generación de archivos planos de Novedades 83
11.2.3.3 Forma de mapeo 83
11.2.4 Reportes 84
11.2.4.1 Reporte de cantidad de Unidades de Consumo 84
11.2.4.2 Reporte de Novedades 84
12. CONFIGURACION FUNCIONAL 85
12.1 APLICATIVO 85
12.1.1 Configuración de objetos 85
12.1.2 Parámetros nuevos 85
12.1.3 Mensajes nuevos 86
12.2 BASE DE DATOS 86
12.2.1 Creación de sinónimo 86
12.2.2 Creación de usuario 86
13. PROCESOS 87
13.1 GENERACIÓN DE CONSUMOS MEDIDOS 87
13.2 EXTRACCIÓN DE INFORMACIÓN DE CONSUMOS MEDIDOS A TRAVÉS DEL API 89
13.3 GENERACIÓN DE ARCHIVO PLANO DE CONSUMOS MEDIDOS 90
13.4 GENERACIÓN DEL REPORTE CANTIDAD DE UNIDADES DE CONSUMO 91
13.5 GENERACIÓN DE NOVEDADES 91
13.6 EXTRACCIÓN DE INFORMACIÓN DE NOVEDADES A TRAVÉS DEL API 94
13.7 GENERACIÓN DEL ARCHIVO PLANO DE NOVEDADES 94
13.8 GENERACIÓN DEL REPORTE DE NOVEDADES 96
14. DIAGRAMAS DE SECUENCIA 97
14.1 GENERACIÓN DE CONSUMOS MEDIDOS 97

14.2 EXTRACCIÓN DE INFORMACIÓN DE CONSUMOS MEDIDOS A TRAVÉS DEL API 98
14.3 GENERACIÓN DE ARCHIVO PLANO DE CONSUMOS MEDIDOS 99
14.4 GENERACIÓN DE NOVEDADES 100
14.5 EXTRACCIÓN DE INFORMACIÓN DE NOVEDADES A TRAVÉS DEL API 101
14.6 GENERACIÓN DEL ARCHIVO PLANO DE NOVEDADES 102
15. ANÁLISIS COMPARATIVO DE SENTENCIAS 103
15.1 SOLUCIÓN A 103
15.1.1 Plan de ejecución para el acceso a productos 104
15.2 SOLUCIÓN B 113
15.2.1 Plan de ejecución para el acceso a productos 114
15.3 SOLUCIÓN C 116
15.3.1 Plan de ejecución para el acceso a productos 118
15.4 PLAN DE EJECUCION PARA LAS SENTENCIAS COMUNES A LAS SOLUCIONES A, B Y C 123
16. COMPARACIÓN DE TIEMPOS EN LAS SOLUCIONES 133
17. PARTICIPANTES 136
18. CRONOGRAMA 137
19. CONCLUSIONES DE DESARROLLO 138
20. CONCLUSIONES PERSONALES 142
21. CONSIDERACIONES ESPECIALES 144
BIBLIOGRAFÍA 145
ANEXOS 147

LISTA DE TABLAS

Pag.

Tabla 1. Estructura de consumos medidos 35
Tabla 2. Tipos de novedades a publicar 38
Tabla 3. Estructura de novedades 39
Tabla 4. Estructura del archivo plano de consumos medidos. 44
Tabla 5. Regla de mapeo y sus diferentes atributos 45
Tabla 6. Estructura del archivo plano de peticiones. 45
Tabla 7. Estructura de consumos medidos a publicar 47
Tabla 8. Estructura de novedades a publicar 51
Tabla 9. Estructura configuración de novedades a publicar 52
Tabla 10. Novedades a publicar 53
Tabla 11. Prototipo de archivo plano de consumos medidos 58
Tabla 12. Campos a mapear 59
Tabla 13. Valores de mapeo 60
Tabla 14. Valores a mapear 60
Tabla 15. API de consulta de consumos medidos 65
Tabla 16. API de consulta de novedades 66
Tabla 17. Entidad EMC_CONSMEDI 67
Tabla 18. Entidad EMC_NOVEDADES 68
Tabla 19. Entidad EMC_NOVEPUBL 69
Tabla 20. Entidad EMC_VALOPAME 69
Tabla 21. Entidad EMC_CAMPMAPE 70
Tabla 22. Generación de consumos medidos 87
Tabla 23. Extracción de información de consumos medidos a través del API 89
Tabla 24. Generación del archivo plano de consumos medidos 90
Tabla 25. Generación del reporte de cantidad de unidades de consumo 91
Tabla 26. Generación de novedades 91
Tabla 27. Continuación de generación de novedades 93
Tabla 28. Extracción de información de novedades a través del API 94
Tabla 29. Generación del archivo plano de novedades 94
Tabla 30. Generación del reporte de novedades 96
Tabla 31. Solución A 134
Tabla 32. Solución B 134
Tabla 33. Solución C 134
Tabla 34. Cronograma 137

LISTA DE FIGURAS

Pag.

Figura 1. Arquitectura de desarrollo para el producto OpenSmartFlex 27
Figura 2. Proceso general 33
Figura 3. Proceso de Telemedidos 34
Figura 4. Ciclo de Peticiones 38
Figura 5. Arquitectura de la solución 63
Figura 6. Diagrama entidad relación de consumos medidos 70
Figura 7. Diagrama entidad de relación de novedades 71
Figura 8. Reporte de consumos medidos resumido 75
Figura 9. Reporte de consumos medidos detallado 76
Figura 10. Reporte de novedades resumido 77
Figura 11. Reporte de novedades detallado 78
Figura 12. Generación de consumos medidos 79
Figura 13. Programación de consumos medidos – Tipo de producto 79
Figura 14. Programación de consumos medidos – Método de cálculo 80
Figura 15. Programación de consumos medidos – Tipo de consumo 80
Figura 16. Publicación de novedades 81
Figura 17. Generación de novedades 81
Figura 18. Programación y ejecución de novedades 82
Figura 19. Generación de archivos de consumos medidos 82
Figura 20. Generación de archivos de novedades 83
Figura 21. Configuración de reglas de mapeo 83
Figura 22. Cantidad de unidades de consumo 84
Figura 23. Reporte de novedades 84
Figura 24. Generación de Consumos Medidos 97
Figura 25. Extracción de información de consumos medidos a través del API 98
Figura 26. Generación del archivo plano de consumos medidos 99
Figura 27. Generación de novedades 100
Figura 28. Extracción de información de novedades a través del API 101
Figura 29. Generación del archivo plano de novedades 102
Figura 30. Modelo de operación de la Solución A 104
Figura 31. Acceso a productos en la solución A 106
Figura 32. Operación de la solución A – Caso 1 (primera iteración) 107
Figura 33. Operación de la solución A – Caso 1 (segunda iteración) 108
Figura 34. Operación de la solución A – Caso 1 (última iteración) 109
Figura 35. Operación de la solución A – Caso 2 (primera iteración) 110
Figura 36. Operación de la solución A – Caso 2 (segunda iteración) 111
Figura 37. Operación de la solución A – Caso 2 (última iteración) 112

Figura 38. Modelo de operación de la solución B 114
Figura 39. Acceso a productos en la solución B 116
Figura 40. Representación de un índice B-Tree 117
Figura 41. Modelo de operación de la solución C 118
Figura 42. Acceso a productos en la solución C 120
Figura 43. Operación de la solución C (primera iteración) 121
Figura 44. Operación de la solución C (segunda iteración) 122
Figura 45. Operación de la solución C (última iteración) 123
Figura 46. Sentencias comunes – Acceso a periodos de consumo 125
Figura 47. Sentencias comunes – Acceso a consumos por producto 126
Figura 48. Sentencias comunes – Acceso a consumos medidos por fecha de generación 128
Figura 49. Sentencias comunes – Acceso a consumos medidos por fecha de proceso 131

LISTA DE ANEXOS

	Pag.
Anexo a. Paper de pasantía	147

RESUMEN

Un proyecto de desarrollo de software de misión crítica para sistemas robustos donde su operación nunca se detiene, requiere de un alto conocimiento de las herramientas de desarrollo de software que se manejen y los sistemas operativos donde éste funcione.

Para llevar a cabo un desarrollo de esta envergadura, es necesario una excelente planeación de cada una de las etapas del proyecto: análisis, diseño y construcción, las cuales son cruciales para este tipo de proyectos.

Adicionalmente a estos pasos se debe poseer un know how propio desarrollado a través del tiempo, la investigación en sistemas críticos y retroalimentación recibida por los clientes. Tal es el caso de la empresa en la que se desarrollo esta pasantía, Open Systems, que es una casa de software dedicada al desarrollo de software para empresas de servicios públicos. Esta empresa con su producto OpenSmartFlex ha alcanzando una solución de software de misión crítica con calidad, que cubre los aspectos necesarios para el funcionamiento de una empresa dedicada al sector de servicios públicos. Sin embargo, no todas las áreas de una empresa pueden cubrirse mediante un solo sistema, ya sea por la existencia de sistemas que han perdurado en el tiempo o por asuntos externos a la implantación del nuevo software; tal es el caso del requerimiento cubierto por esta pasantía, en donde se busca comunicar la aplicación OpenSmartFlex con el sistema de gestión existente en EMCALI E.I.C.E. Energís. Mediante esta integración se garantiza la unicidad de fuente de datos y la sincronía de datos que alimentan los posteriores procesos de gestión que se vienen realizando mediante el sistema externo.

Este documento plasma importantes experiencias vividas durante este proyecto, con el objetivo que estudiantes de informática y carreras a fines que decidan dedicarse al desarrollo de software de misión crítica, basados en bases de datos Oracle, conozcan algunos tips que les ayuden en su vida profesional.

INTRODUCCIÓN

Como alternativa al titulo de grado, la pasantía es una opción que permite aplicar y profundizar los conocimientos adquiridos durante la carrera en el desarrollo de proyectos reales dentro del ámbito empresarial.

La empresa en la cual se desarrollará esta pasantía es Open Internacional Systems. Open provee soluciones de software especializado para los sectores de las telecomunicaciones y los servicios públicos domiciliarios (Telco & Utilities). Es una empresa líder en tecnología y con operación internacional. El producto que ofrece, OpenSmartFlex™, es un sistema de información BOSS (Business and Operations Support Systems) dirigido a compañías de telecomunicaciones y servicios públicos, el cual por medio de alta tecnología basada en reglas, integra funcionalidades y componentes del núcleo principal de gestión en dichas compañías: atención al cliente, facturación y operaciones de campo relacionadas con la infraestructura para la prestación de los servicios. OpenSmartFlex es una solución que mediante la integración de sus componentes OSS (Operations Support Systems) y BSS (Billing Support Systems) permiten administrar los procesos comerciales y de operación de la compañía, cubriendo de manera integral todo el ciclo del negocio.

Aunque OpenSmartFlex posee todos los componentes necesarios para el perfecto funcionamiento de una empresa de servicios públicos, no se puede ignorar la necesidad que tienen los clientes Open de solicitar objetos de negocio personalizados tales como APIs, reportes, procesos, pantallas de usuario y puntos de contacto con sistemas externos, tal es el caso que se presenta en el proyecto de implantación del producto OpenSmartFlex en Emcali Empresa Industrial y Comercial del Estado (EICE), en el cual se requiere diseñar y desarrollar componentes que permitan obtener la información concerniente a los consumos medidos de los productos y a los trámites de los clientes; además de generar la información anteriormente mencionada, deben proveerse los medios a través de los cuales Emcali ponga esta información a disposición de un sistema externo conocido como Energis.

En respuesta a estas necesidades y queriendo optimizar significativamente la satisfacción del cliente Emcali, nos ponemos en disposición de diseñar y construir este requerimiento cumpliendo con los estándares y procedimientos de desarrollo definidos en la compañía los cuales están avalados por la norma de calidad de

software CMMI nivel 4, certificación que Open Systems Ltda. recibió recientemente.

Se espera que este trabajo sea de gran utilidad y aporte a las estrategias trazadas por Open Systems Ltda. así como a futuras propuestas educativas como alternativa de opción de grado.

1. DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES

La tendencia en el desarrollo de sistemas convergentes e integrados que se vislumbra en las empresas, hace necesario que las aplicaciones soporten los cambios y provean servicios a cualquier sistema externo que haga parte del modelo funcional de la compañía. El modelo de arquitectura sobre el cual se desarrolle cualquier aplicación debe soportar y brindar seguridad para llevar a cabo dicha integración.

Es por esto que en Open Systems International, uno de los principales pilares sobre los cuales se desarrollan sus productos es la facilidad de integración con otras aplicaciones. Por otra parte, cabe destacar que el modelo de publicación de datos a sistemas externos debe ser lo suficientemente seguro y confiable para no comprometer la integridad del sistema y asegurar que la información sea precisa, no obstante sin olvidarse del desempeño y ahorro de recursos físicos, haciendo uso de técnicas de afinamiento y buena programación.

Open Systems a través de sus años de experiencia en el mercado de software ha definido una serie de políticas de confiabilidad que tienen como objetivo proteger su información y código fuente de los productos que ofrece a sus clientes. Dentro de estas políticas se cuenta con estándares de programación cuya premisa es restringir el acceso total a sistemas externos.

Para cumplir con esta restricción pero ofrecer medios de comunicación con otros sistemas, Open Systems Intl. en su producto OpenSmartFlex™ aplica el concepto de interfaces o API's (Applicattion Program Interfaces) en las cuales mediante el uso de arquitectura en capas y sinónimos de objetos permite acceder solo a la información pertinente al sistema externo.

1.2 FORMULACIÓN DEL PROBLEMA

Se requiere un medio de publicación de datos entre el sistema OpenSmartFlex y el sistema Energis:

¿Cómo ofrecer un medio de integración entre los sistemas OpenSmartFlex y Energis, en donde se publique información sobre consumos medidos y tramites de servicios suscritos?

2. JUSTIFICACIÓN

Toda empresa requiere de una plataforma tecnológica que solucione sus necesidades con respecto al manejo de la información y a la operación de la empresa como tal. En el caso de una empresa como Emcali EICE – ESP que maneja un volumen de información considerable – cerca de 1'150.000 de suscriptores - y en la que su sistema debe operar sin interrupciones (sistemas funcionando 8 días las 24 horas, *Sistema 8/24*), dicha plataforma debe estar constituida bajo las premisas de desarrollo de software de misión crítica, ya que un bloqueo o un bajo rendimiento en sus aplicativos se deriva en un incorrecto funcionamiento de los procesos funcionales de la compañía, lo que implica la pérdida de dinero, la mala atención al público, la presencia de errores de facturación en el cliente, entre otros problemas.

Así que, teniendo en cuenta el volumen de información que se publicará al sistema externo y los tiempos de respuesta que se pactaron con el cliente al inicio del proyecto, la solución que cubra el requerimiento solicitado por Emcali debe poseer características de software de misión crítica; dicho tipo de software fundamenta su filosofía en el óptimo rendimiento de los procesos de generación de consultas realizadas sobre la base de datos, esto con el fin de desarrollar unidades de código eficientes que eviten bloqueos al ejecutar procesos masivos y como consecuencia generen inactividad y retardo en procesos 'core de negocio' como son facturación y operaciones.

La solución tecnológica que cubre todas las necesidades de Emcali no se limita a los componentes provistos por OpenSmartFlex sino que se estructura mediante la integración de varios sistemas externos a este, en donde se incluye el sistema Energis; la integración debe contar con un mecanismo que facilite compartir la información de consumos y peticiones (manejada por OpenSmartFlex) con el fin de acoplar OpenSmartFlex a la arquitectura tecnológica de Emcali.

En este orden de ideas podemos afirmar que el presente proyecto cuenta con dos fortalezas: Por un lado cubrir el requerimiento de Emcali, quien puntualmente solicita un mecanismo de integración entre el sistema Energis y OpenSmartFlex. Y por otro lado, la oportunidad que tienen los integrantes del proyecto de aprender y aplicar nuevas técnicas de desarrollo que les permitan construir software de misión crítica.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar un sistema que permita la integración entre la aplicación OpenSmartFlex y Energis, con el fin de transmitir la información sobre consumos y peticiones de los clientes.

3.2 OBJETIVOS ESPECÍFICOS

- Especificar los puntos de contacto entre las dos aplicaciones.
- Definir el alcance funcional del proyecto teniendo en cuenta el impacto que pueda generar la creación de nuevas entidades sobre el modelo de datos existente.
- Identificar diferentes puntos de vista desde donde abordar el problema.
- Realizar el diseño de tres posibles soluciones al problema.
- Describir la estructura de la interfaz, la estructura de datos y el mecanismo de integración.
- Apropiar durante el proyecto la arquitectura del producto OpenSmartFlex así como los estándares de programación establecidos en Open para la construcción de software.
- Construir los componentes de software necesarios para cada proceso, haciendo uso de las herramientas en que está desarrollado OpenSmartFlex, como son Oracle Forms, Oracle Reports, Procedimientos Almacenados en PL/SQL y Pro*C.
- Elaborar un estudio comparativo entre las soluciones propuestas para determinar cual es la mejor solución en términos de rendimiento y flexibilidad para posteriormente implantarla donde el cliente.
- Conocer los procesos y técnicas de producción de software de una manera conceptual y aplicada a casos reales.

4. MARCO DE REFERENCIA

4.1 MARCO TEÓRICO

El desarrollo de este proyecto se encuentra enmarcado en el área de la Ingeniería Informática y de Sistemas, en donde se aplica la teoría del desarrollo de software, teoría de sistemas, procesos de desarrollo unificado y métodos de aseguramiento de la calidad.

De igual manera, para la realización de este proyecto se hace necesario el conocimiento de las practicas de desarrollo estándares en Open Systems, en donde se incluye el modelo de arquitectura para los componentes de software, la documentación de las piezas de código que se desarrollen y la estructura del modelo de datos que almacena la información.

4.1.1 Suite de desarrollo Oracle[1] Oracle es básicamente una herramienta que trabaja bajo la arquitectura cliente/servidor, facilitando un Sistema Manejador de Base de Datos Relacional para el almacenamiento de datos y Sistemas de creación de Formas y Reportes.

4.1.1.1 Oracle Database Server En primera instancia se puede mencionar el Sistema Manejador de Base de Datos Relacional (RDBMS), el cual se encarga de gestionar los datos y registros con los que las aplicaciones trabajan. Adicionalmente Oracle Database Server permite almacenar procedimientos, funciones y paquetes desarrollados en PL/SQL de Oracle, un lenguaje de 5ª generación potente para gestionar procesos de la base de datos y permite implementar la lógica de las aplicaciones.

- Historia

El sistema administrador de bases de datos ORACLE, surgió a final de los años 70 y principio de los años 80. George Koch y su equipo de técnicos fueron los primeros en desembarcar en el terreno de Oracle en 1982, durante un proceso de evaluación de Sistema de Gestión de Base de Datos (SGBD) para una importante aplicación comercial que se estaba diseñando y construyendo. Cuando terminaron, la evaluación fue descrita en Computer World como el estudio más severo de SGBD que se había hecho nunca. El estudio fue tan riguroso con los vendedores cuyos productos había estudiado George, que la prensa hizo eco de

[1] BRITO G., Alexandra Del Valle. Historia de Oracle. [en línea]. México: Monografías, 1997. [Consultado 20 Octubre, 2006]. Disponible en Internet: http://www.monografias.com/trabajos25/oracle/oracle.shtml.

sus palabras en lugares tan distantes como Nueva Zelanda y en publicaciones muy alejadas del campo como el Christian Sciencia Monitor.

Oracle conocida entonces como Relational Software, tenía poco más de 25 empleados en aquel tiempo y solo unos pocos clientes importantes. Sin embargo, cuando se completo el estudio, Oracle fue declarada vencedora. George afirmo que el SGBD Oracle era técnicamente el mejor producto del mercado. Estas declaraciones fueron hecha en una época en la que muy poca gente conocía el significado del término "Relacional", y los que lo conocían (o creían conocerlo) no tenían muchas cosas favorables que decir de él. La compañía de Oracle Corporation estaba trabajando entonces para perfeccionar su joven producto, para comprender los tipos de características y funcionalidad que podría hacerlo útil y productivo en el mundo de los negocios. El esfuerzo contribuyo a su refinamiento. Algunas de las características de Oracle, tales como las salidas de SQL*FORMS fueron el resultado de dicho esfuerzo.

- Evolución

El poderoso modelo relacional ha evolucionado desde herramientas y los modelos de datos de redes. La mayor manera aceptada y usada de un modelo de datos es el modelo relacional, el cual fue publicado en el año de 1969 con la revisión hecha por IBM y el Dr. E. F. Codd.

Dentro de las características del modelo relacional de datos, se pueden mencionar tres importantes aspectos:

- Estructuras: Definición de objetos que contengan datos y que son accesibles a los usuarios.
- Operaciones: Que permitan definir acciones que manipulen datos u objetos.
- Reglas: Las cuales se entienden como leyes para gobernar la información, es decir cómo y a quien manipular.

4.1.1.2 Oracle Forms Es uno de los componentes clientes que provee Oracle para el desarrollo de aplicaciones, el cual posee una herramienta basada en formas y eventos, para el desarrollo de aplicaciones de tres o más capas.

La facilidad de uso radica en la fácil interacción del usuario con las formas, ya que este sistema implementa funcionalidades básicas para la consulta de datos, sin necesidad de generar código por parte del desarrollador. Un ejemplo básico de las funcionalidades implementadas por Forms se presenta al tratar de consultar los registros de una tabla, en donde solo basta construir los componentes de presentación (etiquetas, cuadros de texto, barras de desplazamiento, entre otros componentes gráficos), enlazar la fuente de datos (que puede ser una tabla o un procedimiento almacenado en la base de datos), ejecutar la forma y presionar las teclas F7 y F8 para que se desplieguen los datos.

Es importante resaltar la facilidad de integración de Forms con la base de datos Oracle ya que no se hace necesario el uso de controladores de terceros para la comunicación; además el soporte a procedimientos almacenados en la base de datos facilita la creación de la lógica del negocio en una capa independiente a la del cliente (Capa de Reglas del Negocio).

4.1.1.3 Oracle Reports Es el segundo componente de las herramientas cliente de Oracle, la cual está concebida para el desarrollo de reportes de datos, los cuales pueden ser listados, tabulados o cualquier diseño que se desee implementar.

La concepción de los reportes es muy intuitiva y se basa en el modelo de datos que se tiene en la Base de Datos, por lo que se puede representar fácilmente cualquier reporte solicitado por el proceso.

4.1.1.4 Oracle PL/SQL[2] Es una extensión del lenguaje SQL de ORACLE, el cual cuenta varias de las técnicas de lenguajes modernos como encapsulamiento de datos, sobrecargas, manejo de excepciones y demás complementos del paradigma de programación orientado a objetos. Con programas escritos en PL/SQL se pueden escribir sentencias SQL para manejo y procesamiento de datos, declaración de constantes, procedimientos y funciones.

PL/SQL es un lenguaje estructurado en bloques, donde cada bloque puede representar una unidad de código que resuelve un problema en particular.

Por otra parte, el manejo de cursores en estos bloques es una característica que hace que PL/SQL sea un lenguaje eficiente en el manejo de data obtenida de una base de datos ORACLE, ya que los cursores son una representación del modelo de datos real.

De igual manera es importante mencionar la capacidad de abstraer todas las reglas del negocio y consolidarlas en una capa intermedia entre la capa de datos y la capa de presentación, razón por la cual el lenguaje PL/SQL presenta una excelente opción de implementación para la capa de Reglas del Negocio. Debido a que los procedimientos, triggers o paquetes quedan almacenados en la base de datos, el acceso a los datos se hace muy eficiente, mas aún en procesos donde pueden trabajarse con millones miles de millones de registros.

[2] UNIVERSIDAD DE JAÉN. Programación de Bases de Datos Oracle. [en línea]. España: Universidad de Jaén, 2000. [Consultado 1 Julio 2006]. Disponible en Internet: http://wwwdi.ujaen.es/~molina/bd1/apuntes_prac.pdf

4.1.2 Técnicas de rendimiento y ajuste para procesos SQL es bien sabido que el rendimiento de las aplicaciones depende en gran medida de las buenas técnicas de programación y provecho que se haga de las utilidades de las herramientas de programación que se usan.

El caso de las aplicaciones desarrolladas con Oracle no es una excepción, ya que Oracle provee una serie de técnicas que pueden ser usadas para descargar el procesamiento que pueda hacer la base de datos, conociendo de antemano las condiciones en las cuales se ejecutarán los procesos.

- Ajuste de sentencias SQL: Para ejecutar una orden SQL, la base de datos debe determinar el plan de ejecución, el cual es el método mediante el cual se procesará realmente la instrucción, es decir, a que tablas e índices necesita acceder, si se necesitan hacer o no operaciones de ordenación, entre otras funciones.

El plan de ejecución puede tener una gran influencia en el tiempo que se invierte en ejecutar una orden SQL. Para determinar el plan de ejecución existen diferentes métodos para realizar seguimiento y análisis a las sentencias que se ejecutan, por medio de las utilidades EXPLAIN PLAN y TKPROF[3].

- EXPLAIN PLAN

Es una orden que determinará el plan de ejecución para una sentencia SQL dada. Esta orden tiene una estructura en donde se construye por medio de reglas el camino que debe seguir la sentencia involucrada.

- TKPROF

Por medio de esta utilidad puede obtenerse el plan de ejecución de una sentencia dada, al igual que las estadísticas de cómo se ejecutó la orden SQL. Esta utilidad es propia de la base de datos Oracle y permite obtener información de la sentencia como: Tiempo de consumo en la CPU, tiempo de consumo en disco, consultas que realiza, filas que afecta, filas que retorna, cantidad de parseos que se hacen en la sentencia, fetchs que se hacen con la sentencia.

Por otra parte, en la creación de sentencias SQL se pueden emplear HINTS, en donde se especifica directamente al motor por donde debe ejecutar la sentencia. Esta técnica permite modificar:

- Planes de ejecución, ya que permiten optimizar los modos de ejecución de las sentencias.
- Acceso a datos, porque por medio de hints se puede forzar al optimizador usar métodos de escaneo mientras se ejecuta una sentencia.

[3] ORACLE. En: Oracle Enterprise Manager, Database Tuning with the Oracle Tuning Pack, Release 9.0.1. (Jun. 2001); p. 8-1.

- Pueden afectarse las maneras con que el optimizador de consultas elige filas para hacer joins.
- Ejecución paralela, pueden impulsar la ejecución de operaciones en paralelo, lo que reduce considerablemente el costo de ejecución.

Sin embargo sin desconocer las habilidades que tiene Oracle Database Server en cuanto al procesamiento de cantidades considerables de registros, en ocasiones se hace necesario implementar técnicas adicionales, las cuales son desarrolladas mediante la praxis y el conocimiento de las bondades del motor de base de datos.

Tal es el caso de la técnica de *Granularidad*, desarrollada e implementada en Open para el procesamiento de muchos registros.

- Granularidad

Tiene una concepción básica, la cual se puede explicar mediante una simple analogía simple: *"Use el recurso que necesite, sin necesidad de ocupar los demás registros"*.

El problema del bloqueo entre usuarios surgió en procesos donde se necesitaban generar consumos o cargos para 30.000 suscriptores en una empresa de servicios públicos. Al trabajar con el paradigma tradicional, se notó el bajo desempeño de las aplicaciones y el consumo de memoria y procesamiento de manera descomunal.

En este tipo de procesos, en donde si se seleccionan 1.000 registros para procesar y el motor sólo procesa 1 registro a la vez, se notaba que la ocupación de los 999 registros restantes bloqueaba otros procesos que simplemente consultaban 1 de los 999 registros en no uso y ocupados.

Por la anterior razón, se decidió implementar procesos granulados, en donde de los 1.000 registros totales a procesar se trabaja con bloques de 100 registros y los demás permanecen disponibles a otros procesos. La implementación de esta técnica benefició de gran manera los procesos masivos que se generan en las empresas de servicios públicos (como la generación de facturas), en donde de 36 horas que consumía el proceso inicialmente se logró disminuir a 6 horas.

4.2 MARCO CONCEPTUAL

4.2.1 Historia de Open Systems Internacional[4] Open Systems Internacional nace en la ciudad de Cali, en el año de 1987, producto del empeño y dedicación de Orlando Rincón y William Corredor, quienes enfocaron su creatividad en desarrollar sistemas para el sector de las telecomunicaciones, buscando ofrecer una plataforma que se adecuara a las necesidades del mercado latinoamericano.

En un principio los módulos cubiertos por la aplicación de Open se enfocaron a Sistemas de Soporte a Facturación (*Billing Support Systems B.S.S*) los cuales proveen los mecanismos necesarios para que se puedan llevar a cabo en una empresa el control y seguimiento de procesos como liquidación, consumos, cartera. A medida que los requerimientos de las empresas evolucionaban hacia otros procesos críticos Open investigaba y desarrollaba nuevos módulos para madurar la aplicación inicial. Fue asi como Open dio origen a un Sistema de Soporte Operaciones (*Operations Support Systems O.S.S*) en donde se prestan todos los servicios necesarios para generación de ordenes, mantenimiento de red y atención al cliente.

Como producto de la constante integración y convergencia de las instituciones donde se instalaban los sistemas de Open se dio origen a una nueva solución base en Open llamado *OpenFlexIS*, el cual consolidaba dentro de una misma arquitectura los módulos de B.S.S y O.S.S., proporcionando una misma interfaz y modelo de datos a los usuarios.

A través de los años, la madurez y reconocimiento de Open como empresa desarrolladora e investigadora en sistemas de misión crítica se hizo más fuerte, situación que impulsó a sus desarrolladores a expandir las posibilidades en cuanto a integración del sistema base OpenFlexIS con sistemas externos. Debido a este modelo de integración fue reconocida en el año de 2005 con el premio Mejor Implementación en Sistemas de Facturación.

Como consecuencia de todas las experiencias, investigación, desarrollo y retroalimentación de clientes nacio el último producto en Open, conocido con el nombre de *OpenSmartFlex*, el que como su nombre lo indica busca ser totalmente flexible en cuanto a su configuración y personalización. Este sistema al igual que OpenFlexIS consolida los modulos de B.S.S y O.S.S adicionando el componente de generación de reglas de negocio, las cuales son las que permiten al usuario la configuración de todas sus reglas de negocio, eventos y posibles situaciones dentro de su empresa.

---

[4] OPEN SYSTEMS. Información de la Compañía. [en línea]. Colombia: Open Systems, 2000. [Consultado 20 Octubre 2006]. Disponible en Internet: http://openinternational.com/spanish/company/index.htm

La madurez y capacidad del software desarrollado en Open Systems se encuentra sustentada por su ultimo logro, la acreditación en el modelo CMMI nivel IV, el cual propende las buenas practicas de desarrollo y los modelos de medición necesarios en una industria de software.

4.2.2 Misión de Open Systems Internacional[5]. Proveer soluciones informáticas de categoría mundial con tecnología de punta, calidad e innovación.

4.2.3 Modelo de negocio de Open Systems Internacional. El producto desarrollado en Open busca satisfacer las necesidades de empresas de Servicios Públicos y Telecomunicaciones (denominadas como *e-utilities y telco*), ofreciendo una aplicación modular y fácilmente integrable a aplicaciones externas.

En las empresas de servicios públicos, el modelo de negocio cambió drásticamente a finales de los años 90, en las cuales a nivel mundial se notó el aumento en la competitividad por el mercado, cambiando radicalmente el enfoque del marketing protegido por el monopolio, optando por el crecimiento y aumento de márgenes operativos, sustentado por disminución en tiempos de respuesta en procesos financieros y manteniendo mayor control sobre las inversiones en los activos.

Es por esta razón que se considera que el mayor ingreso a una compañía de este tipo se obtiene por la optimización de sus procesos, disminuyendo tiempos de respuesta en la generación de facturación por ejemplo; y el efectivo control que se tenga sobre su infraestructura y provisionamiento. Por otra parte, otro factor diferenciador entre las empresas se encuentra en el mecanismo que se preste al cliente para una respuesta ágil y sin necesidad de trámites redundantes en sus peticiones.

Haciendo referencia a los componentes principales dentro de una empresa de servicios se pueden mencionar dos sistemas principales:

- **Sistema de Soporte a Facturación (Billing Support Systems – B.S.S):** Esta compuesto por todas las unidades o módulos que se refieren a las finanzas, ingresos y egresos dentro de la empresa, como por ejemplo:
  - Fraudes.
  - Tasación.
  - Descuentos.
  - Cartera.
  - Ventas.
  - Facturación.
  - Medición de consumos.

[5] Ibid., p. 21.

Dentro de estos módulos del sistema se manejan operaciones con grandes cantidades de información y de manera concurrente, lo que conlleva a que las unidades de código y procedimientos internos estén completamente sincronizados y sin posibilidad de generar bloqueos en el sistema.

- **Sistema de Soporte a Operaciones (Operation Support Systems – O.S.S)***:* Dentro de este componente se incluyen los módulos correspondientes al soporte operacional que se debe brindar a una empresa para facilitar su modelo de trabajo, como por ejemplo:
  - Provisionamiento.
  - Ordenes de trabajo.
  - Infraestructura.
  - Diseño de redes.
  - Administración del flujo de trabajo.
  - Mantenimiento de red.

En estos módulos se lleva el control de toda la operación de la empresa, soportando el esquema de la misma para facilitar la automatización de procesos y mejora en el rendimiento de la misma.

4.3 ESTADO ACTUAL

La empresa Open Systems a través sus años de experiencia en el mercado del software ha definido unas políticas de confiabilidad que tienen como objetivo proteger los archivos fuente de los productos que vende a sus clientes.

Dentro de estas políticas se cuenta con un estándar de programación cuya premisa es restringir el acceso total de sistemas externos a OpenSmartFlex. Para cumplir con la restricción mencionada anteriormente pero de igual forma permitir compartir información hacia aplicaciones externas, Open ofrece dentro del producto OpenSmartFlex interfaces (denominadas API's) que mediante el uso de capas y de sinónimos de objetos permite acceder solo a la información pertinente para el sistema externo.

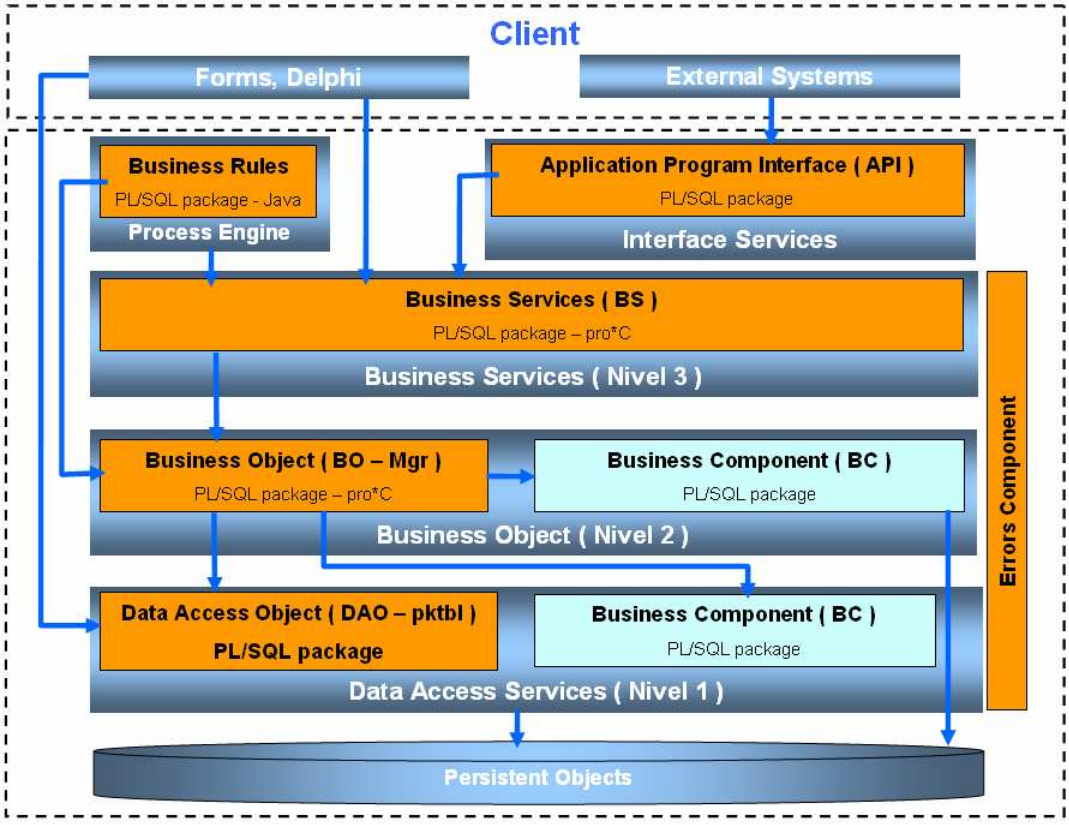
Esta filosofía aplica para todos los clientes Open, entre los que se pueden mencionar EPM-Bogota, EMP-Medellín, Edatel en Antioquia, Hutchinson en Paraguay, Coopelesca en Costa Rica, Telecsa en Ecuador y por supuesto será el esquema de trabajo a utilizar en Emcali EICE-ESP en el desarrollo del punto de integración entre OpenSmartFlex y Energis.

4.3.1 Modelo de integración con sistemas externos uno de los puntos fuertes dentro del producto OpenSmartFlex es la modularidad y fácil integración que puede hacerse con otras aplicaciones del mercado. Esta característica se logró

mediante la abstracción del modelo de datos, reglas de negocio e interfases de usuario, los cuales componen las 3 capas principales dentro del producto.

Debido a esta arquitectura de desarrollo, se pueden desarrollar tanto componentes de alto nivel (denominados dentro del proyecto como API – Aplication Program Interface) los cuales acceden a objetos con reglas de negocios (dentro del proyecto conocidos como BS - Business Services) los cuales para realizar sus procesos utilizan componentes que de manera mas particular realizan funciones particulares dentro de la aplicación (internamente llamados BO – Business Objects). Estos últimos acceden a las estructuras de almacenamiento mediante un último componente de acceso a datos (conocidos como DAO – Data Access Objects) los cuales realizan la interacción directamente con el motor de base de datos.

Figura 1. Arquitectura de desarrollo para el producto OpenSmartFlex



Esta jerarquización de unidades de software es una de las principales premisas dentro del desarrollo de aplicaciones escalables y modulares, en donde sin importar la complejidad del negocio o modelo, se permite trabajar con unidades de código pequeñas y muy entendibles, en donde se encapsulan los procesos

permitiendo la posterior modificación o mejora del proceso sin afectar los componentes que utilizan sus servicios.

Por la anterior razón, la integración con sistemas externos se debe hacer respetando la arquitectura de desarrollo, por lo que se estandarizan 2 maneras de publicación de información a sistemas externos:

- Mediante API: la cual debe contener los llamados a los Business Services correspondientes para realizar su proceso.
- Mediante Archivos Planos: los cuales deben estar construidos de manera entendible para el sistema externo y que sean escritos por componentes del sistema.

De ninguna manera se debe permitir a sistemas externos el acceso directo a tablas, Business Services, Business Object o Data Access Objects, con el fin de proteger la integridad y confidencialidad del sistema.

5. METODOLOGÍA

Para el cumplimiento de los objetivos planteados en el proyecto se concretarán en primera instancia una serie de reuniones con personal de Open Systems para conocer de modo general el modelo funcional de una empresa de servicios públicos, haciendo énfasis en los procesos que le competen a los módulos del proyecto: La generación de consumos y las solicitudes de los clientes.

Por otra parte, se contará con una etapa de autoformación donde cada estudiante bajo la tutoría de un desarrollador de Open, se familiarizará con las herramientas de desarrollo utilizadas en la empresa y aprenderá diferentes técnicas de rendimiento y buen desarrollo que después serán aplicadas en cada una de las respectivas alternativas propuestas como solución final.

Teniendo claros los conceptos del modelo de negocio, se procederá con la puesta en marcha de las etapas convencionales de desarrollo de software usando el Proceso de Desarrollo Unificado[6] (Unified Proces) como modelo de desarrollo.

Las fases a tener en cuenta del desarrollo del proyecto son las siguientes:

- Especificación de requerimientos

El requerimiento del cual se originó el proyecto se definió inicialmente por el cliente de Open Systems, Emcali, razón por la cual se puede omitir esta etapa.

- Análisis

En esta fase se definen las principales funcionalidades de la aplicación y los actores involucrados en ellas. Se hace un bosquejo a grandes rasgos de las operaciones, componentes del sistema y flujo de control de la plataforma. Al finalizar esta etapa se realiza una verificación con el cliente para confirmar que lo que se ha definido en análisis sea realmente lo que él necesita.

- Diseño

En esta etapa se define con mayor detalle el documento resultado del análisis, especificando detalles de la implementación del software como: estructuras de almacenamiento, manejo de la transaccionalidad, estructuras de datos y realización de diagramas de secuencia.

- Implementación

Fase en la cual se desarrolla el código fuente de la aplicación basándose en los documentos obtenidos del diseño.

- Pruebas

Por último se diseña y ejecuta un plan de pruebas conformado por matrices de pruebas, los cuales tienen como objetivo asegurar la calidad de la aplicación

[6] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. Part II: The Core Workflows. En: The Unified Software Development Process (Feb.1999), p 109.

mediante la comprobación de que cada una de las funcionalidades especificadas funciona correctamente y que además se ha cumplido con cada uno de los estándares establecidos por la empresa Open.

5.1 PROPÓSITO DEL PROYECTO

Diseñar e implementar una solución informática que permita la integración entre las aplicaciones OpenSmartFlex y Energis, de manera que se publique la información de peticiones y consumos.

5.2 ALCANCE DEL PROYECTO

La solución informática que se desarrolle para la integración de los módulos de peticiones y consumos entre OpenSmartFlex y Energis tendrá como limitantes las restricciones de seguridad y de acceso a datos establecidas en los estándares de programación y de confidencialidad de información de Open Systems, ofreciendo dos maneras de publicar los datos, una mediante llamado directo desde Energis a un API dentro de OpenSmartFlex y otra mediante la publicación de un archivo plano que tendrá una estructura definida en este proyecto.

Además de los anteriores mecanismos de integración, se deben elaborar los reportes correspondientes a cada uno de los módulos, que permitan la impresión o visualización por pantalla de los datos generados.

Los componentes de software que se elaboren deben estar soportados por los estándares de desarrollo y por la arquitectura de capas implementada en el producto OpenSmartFlex.

6. DESCRIPCION GENERAL

Actualmente Emcali realizó la adquisición del producto OpenSmarFlex, software de misión crítica para empresas de servicios públicos, este sistema tiene dentro sus diferenciadores, la posibilidad de integrarse con otros sistemas de información; actualmente Energis es el software que gestiona tareas de campo, análisis de posibles fraudes (como consumos por encima de lo facturado) y estadísticas de peticiones que no han sido atendidas en Emcali; este necesita estar integrado con el producto recientemente adquirido para recibir información de consumos y peticiones. Con este proyecto, se pretende realizar un desarrollo de software (interfaz) que permita la comunicación entre estos dos sistemas.

6.1 DESCRIPCIÓN DE LA SITUACIÓN

Para el desarrollo de esta interfaz, se cuenta con lo siguiente:

- Herramientas de Desarrollo
  - Base de datos Oracle 9i (Sentencias SQL y Lenguaje PL/SQL)
  - Oracle Developer Forms and Reports
  - Lenguaje Pro *C
- Hardware
  - Servidor Prolaint ML350 G5 y computadores IBM en red, bajo una arquitectura cliente-servidor.
  - Impresora en red.
- Técnicas de Rendimiento
  - Se cuenta con manuales, libros y referencias sobre Afinamiento de Sentencias SQL.
  - Técnicas de granularidad, know-how de Open.
- Formación
  - Existencia de un programa de capacitación de herramientas de desarrollo y técnicas de rendimiento.
  - Entrenamiento del ciclo productivo de Open.
- Documentación
  - DAA – Documento de alcance de arquitectura, donde se documentan los requerimientos, análisis y diseño del desarrollo a realizar.
  - Ayuda en red para Oracle 9i y Oracle Developer Forms and Reports 6i.

Además, hoy la base de datos del producto OpenSmartFlex cuenta con la información pertinente a consumos medidos y peticiones de los clientes de Emcali; igualmente se tiene documentado en el DAA los requerimientos levantados por los ingenieros de Servicios Profesionales de Open que tienen

las necesidades del producto Energis para poder realizar la gestión de trabajo de campo.

6.2 MODULO DE CONSUMOS

6.2.1 Descripción General En las empresas de servicios públicos, los consumos medidos son la parte fundamental para el proceso de facturación, por medio de estos se hace posible el cobro de la factura a un cliente. Los consumos medidos en empresas de servicios públicos se presentan en algunos o todos los servicios prestados por esta. Algunos de los servicios más frecuentes son: telefonía local o larga distancia, telefonía celular, energía, acueducto, alcantarillado, gas, televisión por cable, Internet, entre otros.

Los consumos medidos poseen características, que hacen que se diferencien unos de otros, y a su vez la diferencia de los tipos de clientes que posee una empresa prestadora de servicios públicos, como clientes industriales o personas naturales. Los tipos de consumo dependen del tipo de registro de consumo que se presente por ejemplo:

- Consumo energía activa: es el consumo convencional medido en kilovatios que se presenta en áreas residenciales, oficinas entre otras.
- Consumo energía reactiva: demanda extra de energía medida en kilovatios, que algunos equipos de carácter inductivo como motores, transformadores, iluminarías, necesitan para su funcionamiento lo cual se ve más frecuentemente en el área industrial.
- Consumo energía demanda: consumo de energía medida en kilovatios durante algún lapso que representa un costo adicional.

Estos y otros más, como consumo de internet, consumo de impulsos hacen parte de todos los diferentes tipos de consumo que pueden existir.

Los consumos se miden con base en las lecturas realizadas a los elementos de medición por parte de los equipos de trabajo, el elemento de medición es el encargado de medir el consumo para cada producto que el cliente haya contratado. Después de las lecturas tomadas a los elementos de medición con el fin de registrar el consumo de estos, existen métodos de cálculo de que nos ayudan a identificar el consumo depende el caso:

Consumo por diferencias de lecturas: este consumo se presenta cuando no fue posible la toma de una lectura para un mes dado, por tanto se realiza una diferencia de lecturas con el mes actual y el antepasado para luego promediar y obtener el valor del mes que no se logro lectura.

Consumo corregido: se presenta cuando habido un error en alguna lectura tomada y se ajustan las unidades, debido a la presencia de un reclamo.

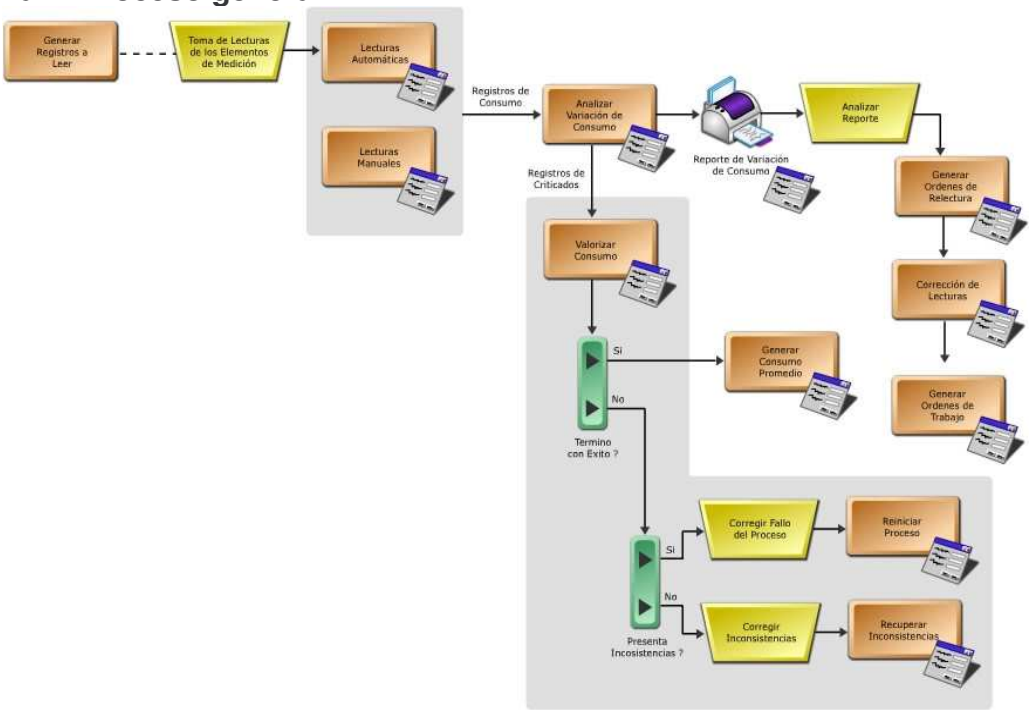
Consumo estimado: dado el caso que no sea posible la toma de lectura al elemento de medición, se estima un consumo basado en el promedio de los últimos seis meses.

Consumo recuperado: se presenta cuando se necesita recuperar consumos de meses anteriores por exceso o defecto en el mes actual.

Consumo Aforado: algunos clientes no poseen instrumento de medición idóneo por lo que se hace necesario la estimación de un consumo para dicho usuario por medio de un proceso de investigación por parte de un experto, a esta estimación se le reconoce como aforo.

6.2.2 Flujo de Negocio el proceso de consumos medidos soporta los procesos relacionados con la generación de los cobros por concepto de consumo, basándose en las lecturas de los registros de los elementos de medición asociados a cada cliente. Adicionalmente, permite detectar posibles inconsistencias en el proceso de toma de lecturas que varíen el patrón de consumo del cliente.

Figura 2. Proceso general



Durante el período de facturación se deben registrar las unidades consumidas por el cliente para un servicio que haya contratado. Estas unidades (lecturas) son tomadas directamente de los elementos de medición y cargadas en la aplicación en forma manual (lectura por lectura de cada elemento de medición) o automática (lecturas provenientes de diferentes elementos de medición que se encuentran contenidas en un archivo). Las lecturas pueden ser tomadas una o varias veces dentro de un mismo período de facturación.

Una vez registradas las lecturas pueden detectarse desviaciones con respecto al consumo promedio de un cliente a través del proceso de análisis de variación del consumo, que genera un reporte con los clientes con consumo considerado anormal. Estos consumos pueden deberse a: error en la toma de la lectura, error en el registro de la lectura, fraudes o problemas técnicos. Si al analizar los consumos anormales, la causa es una lectura mal ingresada, el valor puede ser corregido para evitar un reclamo de facturación posterior por parte del cliente.

Cuando las lecturas se encuentran correctas se pueden generar los cobros por consumo que se cobran al cliente en su estado de cuenta. Si el consumo no puede ser hallado con base en las lecturas, entonces se puede obtener a través del:

- Consumo promedio del cliente para ese producto
- Consumo aforado para el caso de clientes que compartan un mismo elemento de medición.

Si el consumo no puede ser generado por fallas técnicas, el proceso puede ser reiniciado partiendo desde el punto en el que ocurrió la falla. Si el consumo no puede ser generado por falta de información se genera la respectiva inconsistencia para que sea corregida y nuevamente procesada.

Proceso de Consumos de Telemedidos
Para los grandes clientes de energía se aplica el proceso de telemedición, el cual consiste en establecer una conexión con el medidor y recoger la información de sus consumos para realizar posteriormente la liquidación de la energía consumida o de los cargos por el uso de las redes.

Figura 3. Proceso de Telemedidos

El proceso se inicia con la toma de lecturas por el telemedidor; estos registros son cargados al OpenSmartFlex mediante un API; de igual forma es posible modificar o eliminar estos registros. Una vez se tiene esta información cargada se realiza el análisis de variación de consumo el cual se puede ejecutar con una periodicidad diaria o mensual. Es posible generar reportes con la información obtenida en la variación para su respectivo análisis.

6.2.3 Datos a Publicar se debe publicar la información de los consumos medidos de los servicios suscritos según las siguientes definiciones:

Definición Operativa:
La información que se debe publicar al sistema externo Energis, el usuario puede operar esta funcionalidad de manera programada o asistida (de forma inmediata).

Definición Funcional:
El proceso de publicación consiste en generar los consumos medios registrados para un día específico con base en los siguientes métodos de cálculo de consumo: consumo por diferencia lecturas, consumo corregido, consumo estimado, consumo recuperado y consumo aforado.

Estos métodos de cálculo de consumo podrán configurarse de modo que el usuario decida cual de los métodos desea publicar en la interfaz, de igual manera podrá configurar los tipos de consumo que también quisiera publicar.

Definición de Estructura:
La estructura de los datos a publicar al sistema externo Energis es la siguiente:

Tabla 1. Estructura de consumos medidos

Atributo	Acotación
Consecutivo	Corresponde a una secuencia.
Fecha Proceso	Corresponde a la fecha de registro de consumo. Será guardada en formato AAAAMMDD.
Código Producto	Código del producto (servicio suscrito) de energía. Corresponde a key de integración. Será el campo que viajará al sistema externo.
Tipo de consumo	Tipo de consumo. Se define que solo va el tipo de consumo de energía.
Método cálculo de consumo	Método de cálculo de consumo. Los métodos a publicar son los descritos en el punto anterior.
Unidades de consumo	Número de unidades de consumo.
Fecha inicial de consumo	Corresponde a la fecha inicial del periodo de consumo. Si el producto fue instalado durante el periodo de consumo se enviará la fecha de instalación. En caso que el método sea RECUPERADO se publicará la fecha inicial del periodo de consumo del periodo de consumo recuperado.

Fecha final de consumo	Corresponde a la fecha final del periodo de consumo. Si el producto fue retirado durante el periodo de consumo se enviará la fecha de retiro. En caso que el método sea RECUPERADO se publicará la fecha final del periodo de consumo del periodo de consumo recuperado.

Definición de Reproceso:
Se brindará la posibilidad de generar información de consumos medidos para un día que ya se ha generado o para un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. En cualquiera de estos casos, el proceso deberá efectuar la eliminación de los registros correspondientes al día enviado como parámetro y enseguida ejecutará el proceso de generación de consumos medidos.

Definición de Reportes:
Se generaran dos tipos de reportes, detallado o resumido por cantidad de unidades de consumo el cual tendrá rompimiento por día, donde los parámetros de entrada son:

- Fecha inicial: Día proceso inicial
- Fecha final: Día proceso final

Dicho reporte tendrá los siguientes campos:

- Día de proceso
- Tipo de consumo
- Método de cálculo de consumo
- Producto (Solo cuando es detallado)
- Sumatoria de unidades

6.3 MODULO DE PETICIONES

6.3.1 Descripción General dentro del sistema OpenSmartFlex, el módulo encargado de atender todas las inquietudes, solicitudes, reclamos, quejas y demás interacciones que el cliente tenga con la empresa de servicios públicos es el módulo de *Atención al Cliente* (*CRM – Customer Requirements Management*), en donde uno de sus componentes es el Módulo de Peticiones.

Esta sección del CRM se encarga de registrar solicitudes de clientes como modificación de contrato, cambio de un producto contratado o adquisición de uno nuevo; estas solicitudes son registradas y evaluadas por el sistema para validar su posible trámite. Estas peticiones son creadas y configuradas por el cliente según sus necesidades y de igual manera las acciones que deben seguirse para su solución. Las acciones están creadas mediante el Manejador de Flujo de Negocios

(*Workflow),* en el cual mediante reglas de negocio configuradas y eventos del usuario se tramitan las peticiones registradas.

Una vez dichas solicitudes sean validadas y las ordenes de trabajo que se hayan generado de ellas estén cumplidas, se efectúan los cargos de las mismas a la factura del cliente y se finaliza el proceso de atención de peticiones.

6.3.2 Flujo de Negocio el ciclo de vida de una petición inicia en el CRM, mediante el cual el usuario registra una solicitud en particular, la cual puede ser:

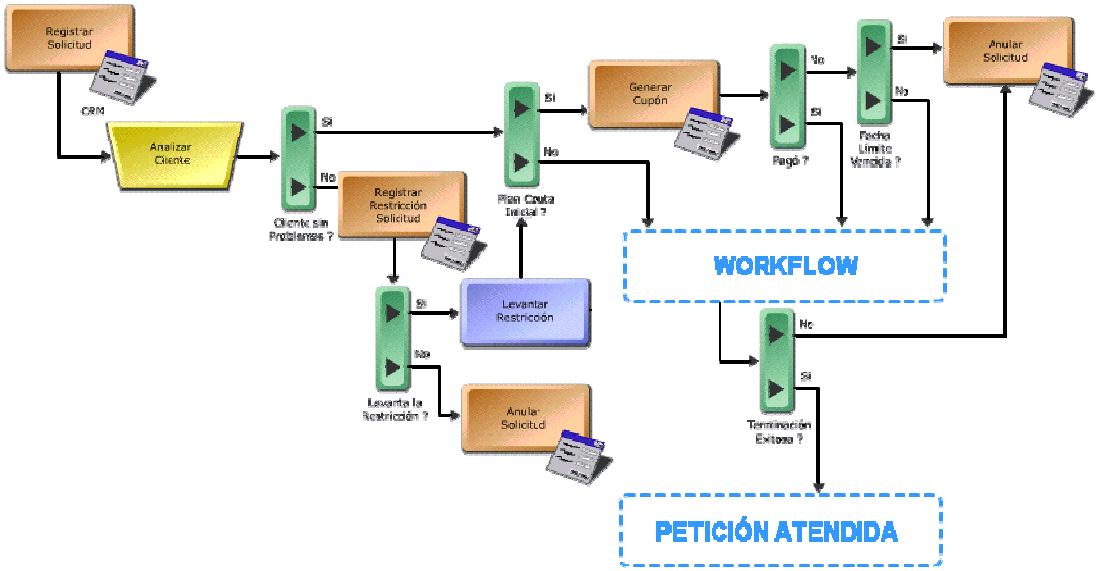
- Cambio de dirección de residencia
- Cambio de un producto
- Reclamo en la facturación de un mes
- Solicitud de un nuevo servicio
- Financiación de deuda pendiente

El cliente que realiza esta solicitud pasa a ser evaluado por el sistema, en donde se hacen una serie de validaciones, como si es un cliente activo, si tiene saldos en mora, si presenta un buen estado crediticio; una vez se hace la validación y se verifica la ausencia de restricciones[*] se verifica si la petición tiene un costo para el cliente, en caso que lo tenga se imprime y entrega el cupón de pago. Una vez se haya realizado el pago la solicitud para a ser manejada por el motor de Flujo de Negocio *Workflow*, en donde a través de reglas y procedimientos almacenados y previamente configurados se efectúa la activación de la solicitud. Una vez la solicitud genera todas las ordenes de trabajo (por ejemplo cambio de medidor, actualización automática de datos del cliente, amortización de una deuda pendiente) la petición queda en estado atendido y se generan los cargos correspondientes. En caso que alguna de las condiciones no se cumpla, la petición pasa a cancelarse y se le notifica al usuario el caso por el cual no pudo realizarse la petición.

En el siguiente gráfico se muestra de manera clara el ciclo productivo de una petición en el sistema OpenSmartFlex.

[*] Posibles causas que desmejoren la calidad del cliente e impidan la atención a una petición.

Figura 4. Ciclo de Peticiones



6.3.3 Datos a Publicar se debe publicar la información de las peticiones o novedades de los clientes según las siguientes definiciones:

Definición Operativa:
La información que se debe publicar al sistema externo Energis, el usuario puede operar esta funcionalidad de manera programada o asistida (de forma inmediata).

Definición Funcional:
Este proceso consiste en generar las novedades registradas para un día específico. Los tipos de novedades a publicar son:

Tabla 2. Tipos de novedades a publicar

TIPOS DE NOVEDADES A PUBLICAR		
TIPO DE NOVEDAD	**NOVEDAD**	**ACOTACIÓN**
1	Cambio de datos de usuario	Corresponde a cambio en los datos de producto
2	Cambio de dirección	Corresponde al cambio dirección del producto
3	Cambio de medidor	Corresponde al cambio del elemento de medición del producto
4	Cambio ciclo de consumo	Corresponde al cambio de ciclo de medición del usuario
5	Cambio de estrato	Corresponde al cambio de categoría del usuario
6	Suspensiones de servicio	Corresponde a suspensiones por atención

		al cliente y por no pago
7	Retiro de servicio	Corresponde a retiros por atención al cliente y por no pago
8	Reconexión de servicio	Corresponde a las reconexiones por atención al cliente o por no pago efectuadas al producto.
9	Instalación nuevo medidor	Corresponde a la instalación de medidor a un usuario directo. No corresponde a un cambio de medidor ya que en este caso el usuario no tenía medidor anteriormente.

Cada novedad podrá tener un costo equivalente a N salarios mínimos legales diarios, el cual deberá ser cargado al producto en el que efectuó la novedad. El concepto de cobro de la novedad deberá ser parametrizable y el usuario podrá ingresar su valor libremente.

Por otra parte, no se contempla la publicación de las siguientes novedades:

- Cambio de Punto de Amarre por cambio de infraestructura: No forma parte de esta fase.
- Cambio de tarifa: Corresponde a un cambio de estrato.
- Reinstalación: Aunque OpenSmartFlex puede publicarla no se encuentra definido el modelo de negocio de dicho trámite.

Definición de Estructura:
Con el fin de cumplir el requerimiento y almacenar las novedades para su posterior publicación en el sistema Energis, se crea una entidad a nivel de base de datos, la cual tendrá la siguiente estructura:

Tabla 3. Estructura de novedades

ESTRUCTURA DE NOVEDADES					
Atributo	**Tipo dato**	**Precisión**	**Escala**	**Acotación**	**Tipo de novedad**
Consecutivo	Numérico	15		Corresponde a una secuencia.	No depende de ningún tipo de novedad a publicar.
Fecha proceso	Numérico	8		Corresponde a la fecha de atención de la novedad. Se guarda en formato AAAAMMDD	No depende de ningún tipo de novedad a publicar.
Código producto	Numérico	10		Código del producto (servicio suscrito) de energía.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Nombre	Varchar2	100		Nombre del usuario que usa el producto.	1

Dirección	Varchar2	200		Dirección de instalación del producto.	2
Teléfono	Varchar2	50		Número telefónico donde se encuentra instalado el producto.	2
Ciclo de consumo	Numérico	4		Ciclo de consumo del producto.	4
Sector	Numérico	4		Código del sector operativo al cual pertenece el producto.	4
Ruta	Numérico	20		Código de la ruta a la cual pertenece el producto.	4
Consecutivo de ruta	Numérico	15		Consecutivo del predio dentro de la ruta.	4
Fecha novedad	Date			Fecha de atención de la novedad.	Aplica para todas las novedades.
Tipo novedad	Numérico	4		Corresponde al tipo de novedad. Ver tabla de Novedades a Publicar.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Departamento	Numérico	4		Corresponde al código de departamento del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Localidad	Numérico	6		Corresponde al código de localidad del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Uso del servicio	Numérico	2		Corresponde a la categoría del producto.	5
Estrato	Numérico	2		Corresponde a la sub-categoría del producto.	5
Clase de servicio	Numérico	4		Corresponde al código de la clase de servicio del componente medidor del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Consumo promedio	Numérico	15	3	Ultimo consumo promedio calculado para el producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Carga instalada	Numérico	15	2	Carga declarada al momento de la instalación del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Serie medidor	Varchar2	50		Serie del elemento de medición	3,9

Marca medidor	Varchar2	50		Marca del elemento de medición	3,9
Modelo medidor	Varchar2	50		Modelo del elemento de medición	3,9
Tipo elemento de medición	Numérico	4		Tipo del elemento de medición	3,9
Clase elemento de medición	Numérico	4		Clase del elemento de medición	3,9

Definición de Reproceso:
Se brindará la posibilidad de generar información de las peticiones para un día que ya se ha generado o para un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. En cualquiera de estos casos, el proceso deberá efectuar la eliminación de los registros correspondientes al día enviado como parámetro y enseguida ejecutará el proceso de generación de peticiones.

Definición de Reportes:
Se generaran dos tipos de reportes, detallado o resumido el cual tendrá rompimiento por día, donde los parámetros de entrada son:
Fecha inicial : Día proceso inicial
Fecha final : Día proceso final

Dicho reporte tendrá los siguientes campos:
- Día de proceso
- Tipo de novedad
- Producto : Solo cuando es detallado.
- Nombre : Solo cuando es detallado.
- Cantidad

6.4 MÉTODOS DE INTEGRACIÓN
El avance tecnológico y la necesidad de optimizar la gestión de la información han obligado a las empresas a realizar continuas inversiones en adquisición y desarrollo de software con el fin de mejorar su competitividad. No obstante, la adquisición de software no asegura por si misma un funcionamiento eficiente de los procesos de una compañía, puesto que el éxito de la implantación del software depende de diversos factores, entre ellos y quizás el que más se pasa por alto: la integración del software adquirido con aplicaciones ya existentes en la empresa e incluso con sistemas externos a la misma (proveedores, clientes, etc.).

La integración entre aplicaciones hace referencia a los mecanismos que permiten establecer un flujo de información entre un software y otras aplicaciones, y surge

al detectarse la necesidad de compartir o transferir información de un sistema a otro; en este orden de ideas, los métodos de integración son todos los medios a través de los cuales una aplicación se comunica con ella misma y con otros sistemas externos.

Cabe resaltar que los datos compartidos desde una aplicación hacia los sistemas externos son limitados, pues hay datos que no le conciernen a éstos, ya sea porque es información confidencial para la primera aplicación o porque sencillamente es información sin utilidad para el otro sistema; debido a esto se deben contemplar todos los mecanismos de seguridad que permitan publicar sólo la información necesaria por el sistema externo, pues los datos o invocaciones que se reciben de sistemas externos sobre los que no se tiene control son posibles fuentes de ataque o de violación ya que pueden intentar acceder a información a la cual no tienen permiso.

Algunos escenarios en los que se presenta integración entre aplicaciones son:

- Interfaz entre una aplicación bancaria y una aplicación Web para habilitar pagos en línea.
- Interfaz entre un sistema generador de cargos por consumos y una aplicación externa que imprima las facturas correspondientes a dichos cargos.
- Publicación de consumos via Web. Por ejemplo, habilitando una interfaz Web a través de la cual un cliente determinado pueda consultar el consumo efectuado sobre la bolsa de minutos de su plan de telefonía celular.
- Interacción de aplicaciones con sistemas unificadores cuya función es transformar diversos formatos de entrada en un formato estandarizado de salida, normalmente se comparte información a través de archivos planos (Caso muy común en sistemas relacionados con telecomunicaciones).

Los métodos de integración requeridos por Emcali para compartir información con el sistema externo Energis son: interfaces diseñadas para que Energis acceda única y exclusivamente a determinados objetos de la base de datos (APIs) y archivos planos alojados en el servidor.

6.4.1 Publicación de información mediante entidades customizadas en la base de datos de OpenSmartFlex para el cliente Emcali, se crean dos entidades para la publicación de la información hacia el sistema externo Energis, una para la publicación de consumos medidos y otra para la publicación de peticiones. El proceso consiste en tomar todos los datos almacenados en estas entidades y publicarlos al sistema externo por medio de un API donde se define un cursor referenciado, que permite capturar masivamente los datos y exportarlos hacia Energis.

En nuestro caso se tendrá un API para cada entidad a publicar donde la definición del cursor referenciado conservará la misma estructura de la entidad definida posteriormente, así Energis podrá leer los consumos y peticiones consignados en la base de datos de OpenSmartFlex.

6.4.2 Publicación de información mediante archivos planos en el desarrollo del presente proyecto se implementaran los componentes necesarios para extraer en archivos la información generada sobre consumos y peticiones. El objetivo de estos archivos es publicar los consumos y/o peticiones registrados en el producto OpenSmartflex para que el sistema externo Energis pueda disponer de dicha información.

El desarrollo de esta funcionalidad para la publicación de información mediante archivos planos permitiá generar archivos planos de consumos y de peticiones de manera asistida y programada, es decir, el usuario de la aplicación OpenSmartFlex tiene la posibilidad de programar una fecha y hora de ejecución sí desea que la generación de registros se ejecute en un momento posterior a la configuración de los datos de entrada del proceso.

Cabe resaltar que en el requerimiento solicitado por Emcali especifica la necesidad de realizar un mapeo de la información, es por esta razón que durante la extracción se ha implementado un proceso de conversión de la información, dicho proceso consiste en tomar el contenido de ciertos campos de la entidad de peticiones (el mapeo se presenta en la generación de archivos planos de peticiones mas no en la generación de archivos de consumos) y pasarlo por un conversor que genera su respectivo equivalente. Posterior al proceso de mapeo, la información resultante será almacenada en un archivo plano. El objetivo de mapear la información es poder brindarle al sistema externo Energis los datos en la forma que el los reconoce.

Los archivos planos son almacenados del lado del servidor en una ruta para los archivos de consumos y en otra diferente para los archivos planos de peticiones.

6.4.2.1 Definición de la estructura para el archivo de consumos el archivo plano donde se publican los consumos presenta la misma estructura que la entidad de consumos medidos, es decir, cada campo de la entidad ocupa en el archivo plano los mismos caracteres definidos por la precisión del campo. La extensión definida para los archivos es txt (texto plano) y la nomenclatura utilizada para el nombramiento de los mismos es la siguiente:

<Funcionalidad>_<Fecha generada>_<Fecha de ejecución><Hora de ejecución>.txt

Donde:

- Funcionalidad: Mnemónico del programa ejecutado.
- Fecha generada: Fecha ingresada como parámetro de entrada del proceso en formato AAAAMMDD.
- Fecha de ejecución: Fecha de ejecución del proceso en formato AAAAMMDD.
- Hora de ejecución: Hora de ejecución del proceso en formato HH24MISS.

Ejemplo: COME_20051122_20060125175601.txt

Tabla 4. Estructura del archivo plano de consumos medidos.

Campo	Posición inicial	Posición final	Long.	Tipo de dato	Observación/Restricción
Consecutivo	1	15	15	Numérico	Va en la posición 1
Fecha proceso	16	24	8	Fecha	Ninguna
Código producto	25	35	10	Numérico	Ninguna
Tipo de consumo	36	40	4	Numérico	Ninguna
Método de cálculo de consumo	41	45	4	Numérico	Ninguna
Unidades de consumo	46	61	15	Numérico	Campo con precisión decimal 3. El valor registrado en el archivo es el valor real en la entidad de consumos multiplicado por 1000.
Fecha inicial de consumo	62	76	14	Fecha	Formato AAAAMMDDHH24MISS
Fecha final de consumo	77	91	14	Fecha	Formato AAAAMMDDHH24MISS

6.4.2.2 Definición de formato de archivo para peticiones el archivo plano donde se publican las peticiones presenta la misma estructura que la entidad de peticiones; cada campo de esta entidad ocupa en el archivo plano los mismos caracteres definidos por la precisión del campo a excepción de los campos mapeados, los cuales ocupan la cantidad de caracteres definida por la suma de sus atributos de mapeo (máximo 9 caracteres para la clase de servicio, 10 para el código del producto y 4 para la clase del elemento de medición), dado el caso que la suma de los atributos de mapeo tenga una cantidad de caracteres mayor a las ya especificadas para cada campo, la cadena es truncada.

Si no existe regla de mapeo para un código específico, en el archivo plano se registra el código original almacenado en la entidad de peticiones.

Los campos en el archivo plano de peticiones que requieren ser mapeados son:

- Código del producto
- Clase de servicio
- Clase de elemento de medición

Cada uno de estos tres campos tiene un número asociado de atributos de mapeo y una regla de mapeo; la generación de reglas de mapeo es configurable, es decir, se le permite al usuario del producto OpenSmartFlex definir las reglas de mapeo para cada uno de los campos que lo requieran.

Tabla 5. Regla de mapeo y sus diferentes atributos

Campo	Código a mapear	Atributo de mapeo 1	Atributo de mapeo 2	Atributo de mapeo N
Código de producto	1	12	34	56
Clase de servicio	1	ab	cd	ef
Clase del elemento de medición	25	2	38	105

Tomando como ejemplo la primera fila de la tabla XXX el código de producto registrado en el archivo de peticiones sería la suma de todos los atributos de mapeo existentes, es decir, el código en el archivo sería el número 123456 en lugar del código 1 original, en el caso de la clase de servicio se registraría en el archivo el valor abcdef en lugar del código 1 y finalmente, la clase del elemento de medición en el archivo de peticiones llevaría el código 2381 en lugar del código 25. Cabe resaltar que la cadena resultado de sumar los atributos de mapeo de la clase del elemento de medición es truncada a sólo 4 posiciones ya que este es el máximo de caracteres definido para este campo.

La extensión definida para los archivos de peticiones es txt (texto plano) y la nomenclatura utilizada para el nombramiento de los mismos es la siguiente:

<Funcionalidad>_<Fecha generada>_<Fecha de ejecución><Hora de ejecución>.txt

Donde:

- Funcionalidad: Mnemónico del programa ejecutado.
- Fecha generada: Fecha ingresada como parámetro de entrada del proceso en formato AAAAMMDD.
- Fecha de ejecución: Fecha de ejecución del proceso en formato AAAAMMDD.
- Hora de ejecución: Hora de ejecución del proceso en formato HH24MISS.

Ejemplo: NOVE_20051122_20060125175601.txt

Tabla 6. Estructura del archivo plano de peticiones.

Campo	Posición inicial	Posición final	Long.	Tipo de dato	Operación/Restricción
Consecutivo	1	15	15	Numérico	Va en la posición 1
Fecha proceso	16	24	8	Numérico	Ninguna
Código producto	25	35	10	Numérico	Campo mapeado. Posee un atributo de mapeo.

Nombre	36	136	100	Alfanumérico	Ninguna
Dirección	137	337	200	Alfanumérico	Ninguna
Teléfono	338	388	50	Alfanumérico	Ninguna
Ciclo de consumo	389	393	4	Numérico	Ninguna
Sector	394	398	4	Numérico	Ninguna
Ruta	399	419	20	Numérico	Ninguna
Consecutivo de ruta	420	435	15	Numérico	Ninguna
Fecha novedad	436	450	14	Fecha	Ninguna
Tipo novedad	451	455	4	Numérico	Ninguna
Departamento	456	460	4	Numérico	Ninguna
Localidad	461	467	6	Numérico	Ninguna
Uso del servicio	468	470	2	Numérico	Ninguna
Estrato	471	473	2	Numérico	Ninguna
Clase de servicio	474	483	9	Numérico	Campo mapeado. Posee tres atributos de mapeo.
Consumo promedio	484	499	15	Numérico	Campo con componente decimal (15, 3).
Carga instalada	500	515	15	Numérico	Campo con componente decimal (15, 3).
Serie medidor	516	566	50	Alfanumérico	Ninguna
Marca medidor	567	617	50	Alfanumérico	Ninguna
Modelo medidor	618	668	50	Alfanumérico	Ninguna
Tipo elemento de medición	669	673	4	Numérico	Ninguna
Clase elemento de medición	674	678	4	Numérico	Campo mapeado. Posee un atributo de mapeo.

7. ALCANCE FUNCIONAL

Se desarrollarán utilidades que permitan consultar:
- Consumos medidos
- Novedades de clientes

Para ello se crearán procesos de generación de información cuya salida será almacenada de manera persistente en entidades de la base de datos (tabla de consumos medidos y tabla de novedades).

Una vez generada, la información podrá ser consultada de las siguientes maneras:
- Generando un reporte
- API
- Generando archivos planos

7.1 GENERACIÓN DE CONSUMOS MEDIDOS

Los consumos medidos se pueden generar de dos maneras:
- Asistida: El proceso se ejecuta inmediatamente.
- Programada: El proceso se ejecuta en un día y hora especificados por el usuario.

Se creará una forma a partir de la cual se generará la información sobre consumos medidos de un día en específico. Esta forma se utilizará tanto para la generación asistida como para la generación programada de consumos medidos. La diferencia entre los dos casos es que la generación asistida no requiere los campos Fecha y Hora de ejecución, por lo tanto cuando se desee generar información sobre consumos medidos de este modo, dichos campos deben dejarse vacíos.

7.1.1 Definición de la estructura de datos se creará la tabla de consumos medidos en la base de datos. La estructura de los datos a publicar será la siguiente:

Tabla 7. Estructura de consumos medidos a publicar

ESTRUCTURA CONSUMOS MEDIDOS				
Atributo	**Tipo dato**	**Precisión**	**Escala**	**Acotación**
Consecutivo	Numérico	15		Corresponde a una secuencia.
Fecha Proceso	Numérico	8		Corresponde a la fecha de registro de consumo. Será guardada en formato AAAAMMDD.

Código Producto	Numérico	10		Código del producto (servicio suscrito) de energía. Corresponde a key de integración. Será el campo que viajará al sistema externo.
Tipo de consumo	Numérico	4		Tipo de consumo. Se define que solo va el tipo de consumo de energía.
Método cálculo de consumo	Numérico	4		Método de cálculo de consumo. Los métodos a publicar son los descritos anteriormente.
Unidades de consumo	Numérico	15	3	Número de unidades de consumo.
Fecha inicial de consumo	Date			Corresponde a la fecha inicial del periodo de consumo. Si el producto fue instalado durante el periodo de consumo se enviará la fecha de instalación. En caso que el método sea RECUPERADO se publicará la fecha inicial del periodo de consumo del periodo de consumo recuperado.
Fecha final de consumo	Date			Corresponde a la fecha final del periodo de consumo. Si el producto fue retirado durante el periodo de consumo se enviará la fecha de retiro. En caso que el método sea RECUPERADO se publicará la fecha final del periodo de consumo del periodo de consumo recuperado.

Esta será una entidad customizada[*], donde la precisión es la longitud del tipo de dato y la escala es el número de decimales en el caso de que el tipo de dato sea numérico.

7.1.2 Generación asistida de consumos medidos la generación asistida se ejecutará desde la forma de generación de consumos medidos. Para el tipo de ejecución asistida deben dejarse nulos los campos fecha y hora para que el proceso se ejecute inmediatamente.

La forma recibirá como parámetros de entrada:

- Fecha de consumo deseada (Día de proceso): Corresponde a la fecha en la que se registró el consumo.
- Servicio: Corresponde al tipo de producto.
- El tipo de consumo.
- Fecha y hora: Campos nulos si se desea ejecutar el proceso de inmediato.
- Método de cálculo de consumo: Los métodos de cálculo de consumo tenidos en cuenta para publicación serán:
  - Diferencia de lecturas
  - Corregido
  - Estimado
  - Recuperado
  - Aforado

Nota: La interfaz permitirá que se elija uno o varios métodos de cálculo de consumo, así como también será posible elegir uno o varios tipos de consumo.

[*] Entidad que se crea para la solución especifica de un requerimiento y no hace parte del producto OpenSmartFlex.

La información generada se almacenará en la tabla de la base de datos descrita en el ítem 7.1.1

OpenSmartFlex brindará la posibilidad de generar información de consumos medidos para un día que ya se haya generado o para un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. En cualquiera de estos casos, el proceso deberá efectuar la eliminación de los registros correspondientes al día enviado como parámetro y enseguida ejecutará el proceso de generación de consumos medidos.

7.1.3 Generación programada de consumos medidos la generación programada se ejecutará desde la forma de generación de consumos medidos. Adicionalmente, se habilitarán los medios para permitir que esta ejecución sea programada.

Para este fin, la ejecución del proceso de generación de consumos medidos se programará a través de jobs, empleando la utilidad de Jobs de OpenSmartFlex.

Los parámetros necesarios para la generación programada de consumos medidos son los mismos requeridos para la generación asistida, solo que en este caso el campo hora, el campo fecha o ambos deben ser ingresados por el usuario.

El comando de ejecución del proceso tendrá la siguiente sintaxis:

```
PROCEDURE SUBMIT (
        Qué          IN    VARCHAR2,
        Fecha_sig    IN    DATE,
        Intervalo    IN    VARCHAR2 DEFAULT NULL,
        trabajo      OUT BINARY INTEGER
        );
```

Donde:

- *trabajo*: Es el código de trabajo a programar.
- *Qué*: Es el llamado al package que inicia el proceso. (Con parámetros de proceso incluidos).
- *Fecha_sig:* Fecha en la que se va a ejecutar el proceso por primera vez (SYSDATE en el caso que se desee ejecutar inmediatamente).
- *Intervalo:* Función que calcula cuando se ejecutará el proceso de nuevo (NULL en el caso que se desee ejecutar una sola vez).

La información generada por el proceso se almacenará en la tabla de la base de datos descrita en el ítem 7.1.1.

Al igual que en la generación asistida, se brindará la posibilidad de generar información de consumos medidos para un día que ya se haya generado o para un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. En cualquiera de estos casos, el proceso deberá efectuar la eliminación de los registros correspondientes al día enviado como parámetro y enseguida ejecutará el proceso de generación de consumos medidos.

7.1.4 Implementación del API se implementará un API a través del cual se habilitará la extracción de información sobre los consumos medidos de un producto en particular. El API recibirá un producto (servicio suscrito) como parámetro de entrada y devolverá como resultado un cursor referenciado con todos los consumos medidos de dicho producto encontrados en la entidad customizada. Los campos que devolverá el cursor serán los mismos campos definidos en la estructura de la tabla consumos medidos descrita en el ítem 7.1.1.

7.1.5 Generación del Reporte Cantidad de unidades de consumo se creará un reporte que muestre la cantidad de unidades de consumo en un rango de fechas dadas, para este fin se empleará la utilidad Reports de Oracle. Dicho reporte se podrá generar detallado o resumido y estará agrupado por día.

El contenido del reporte puede ser generado de las siguientes maneras:
- Formato archivo.
- Vista preliminar en pantalla.
- HTML.

Se desarrollará una forma por medio de la cual se permitirá generar el reporte, esta forma tendrá como parámetros de entrada:
- Fecha inicial: Día inicial del proceso.
- Fecha final: Día final del proceso.
- Flag tipo reporte: Permite identificar si el reporte a generar es detallado o resumido.

Las columnas a listar en el reporte son:
- Día de proceso.
- Tipo de consumo (Código y descripción).
- Método de cálculo (Código y descripción).
- Producto – Solo en el caso de que el reporte sea detallado.
- Sumatoria de unidades.

7.2 GENERACIÓN DE NOVEDADES

Las novedades se pueden generar de dos maneras:

- Asistida: El proceso se ejecuta inmediatamente.
- Programada: El proceso se ejecuta en una fecha y hora especificada por el usuario.

Se creará una forma que permita generar la información sobre novedades de un día en específico. Esta forma se utilizará tanto para la generación asistida como para la generación programada de novedades.
Para el caso de la generación programada debe ingresarse una hora y/o fecha de ejecución, mientras que en la generación asistida no son necesarios estos campos ya que el proceso se ejecuta de inmediato, por lo tanto, dichos campos deben ser nulos.

7.2.1 Definición de la estructura de datos se creará la tabla de Novedades en la base de datos. La estructura de los datos a publicar será la siguiente:

Tabla 8. Estructura de novedades a publicar

ESTRUCTURA DE NOVEDADES					
Atributo	**Tipo dato**	**Precisión**	**Escala**	**Acotación**	**Tipo de novedad**
Consecutivo	Numérico	15		Corresponde a una secuencia.	No depende de ningún tipo de novedad a publicar.
Fecha proceso	Numérico	8		Corresponde a la fecha de atención de la novedad. Se guarda en formato AAAAMMDD	No depende de ningún tipo de novedad a publicar.
Código producto	Numérico	10		Código del producto (servicio suscrito) de energía.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Nombre	Varchar2	100		Nombre del usuario que usa el producto.	
Dirección	Varchar2	200		Dirección de instalación del producto.	
Teléfono	Varchar2	50		Número telefónico donde se encuentra instalado el producto.	
Ciclo de consumo	Numérico	4		Ciclo de consumo del producto.	
Sector	Numérico	4		Código del sector operativo al cual pertenece el producto.	
Ruta	Numérico	20		Código de la ruta a la cual pertenece el producto.	
Consecutivo de ruta	Numérico	15		Consecutivo del predio dentro de la ruta.	
Fecha novedad	Date			Fecha de atención de la novedad.	Aplica para todas las novedades.
Tipo novedad	Numérico	4		Corresponde al tipo de novedad. Ver tabla.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Departamento	Numérico	4		Corresponde al código de departamento del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.

Localidad	Numérico	6		Corresponde al código de localidad del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Uso del servicio	Numérico	2		Corresponde a la categoría del producto.	
Estrato	Numérico	2		Corresponde a la sub-categoría del producto.	
Clase de servicio	Numérico	4		Corresponde al código de la clase de servicio del componente medidor del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Consumo promedio	Numérico	15	3	Ultimo consumo promedio calculado para el producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Carga instalada	Numérico	15	2	Carga declarada al momento de la instalación del producto.	No depende de ningún tipo de novedad a publicar. Se publica la información actual del producto.
Serie medidor	Varchar2	50		Serie del elemento de medición	
Marca medidor	Varchar2	50		Marca del elemento de medición	
Modelo medidor	Varchar2	50		Modelo del elemento de medición	
Tipo elemento de medición	Numérico	4		Tipo del elemento de medición	
Clase elemento de medición	Numérico	4		Clase del elemento de medición	

Esta será una entidad customizada, donde la precisión es la longitud del tipo de dato y la escala es el número de decimales en el caso de que el tipo de dato sea numérico.

7.2.2 Configuración de las novedades a publicar teniendo en cuenta que las novedades tienen asociado un costo equivalente a N salarios mínimos legales diarios, se creará una entidad de novedades a publicar, esta entidad permitirá almacenar la cantidad de salarios mínimos diarios que se le cobrarán al cliente por cada tipo de novedad. La estructura de datos de dicha entidad será la siguiente:

Tabla 9. Estructura configuración de novedades a publicar

ESTRUCTURA CONFIGURACIÓN DE NOVEDADES A PUBLICAR				
Atributo	**Tipo dato**	**Precisión**	**Escala**	**Acotación**
Id del tipo de novedad	Numérico	4		Id del tipo de motivo. Corresponde a key de integración
Factor de cobro	Numérico	4	2	Corresponde a la cantidad N de salarios mínimos diarios a cobrar por la novedad.

Las novedades publicadas en la entidad al momento de generar novedades serán las únicas que se contemplarán en el proceso de generación. Un ejemplo de esta seria:

Tabla 10. Novedades a publicar

NOVEDADES A PUBLICAR	
Asociación id del motivo	**Salarios mínimos legales diarios a cobrar (N)**
37	0
66	0
70	2
32	3
56	3
55	1
13	4

Se creará la forma de novedades a publicar, a través de la cual se podrá añadir otra novedad a publicar así como también se podrá actualizar la cantidad de salarios mínimos diarios legales a cobrar por cada tipo de novedad.

7.2.3 Generación asistida de Novedades la generación asistida de novedades se ejecuta desde la forma de Generación de novedades. La forma recibirá como parámetros de entrada:

- Fecha de proceso: Corresponde a la fecha de atención de la novedad.
- Servicio: Corresponde al tipo de producto.
- Tipo de consumo.

La información generada se almacenará en la entidad de la base de datos descrita en el ítem 7.2.1. Adicionalmente se creará un proceso donde a partir de la novedad registrada, se genere un cargo al producto asociado con un monto correspondiente a la cantidad de salarios mínimos diarios definida para dicha novedad. (Ver tabla de novedades a publicar). Además no se generarán cargos donde el factor de cobro sea cero.

Se brindará la posibilidad de generar información de Novedades para un día que ya se haya generado o para un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. Dado este caso, el proceso deberá eliminar los registros correspondientes al día enviado como parámetro (se

eliminarán tanto las novedades como los cargos generados al producto) y enseguida ejecutará el proceso de generación de Novedades.

7.2.4 Generación programada de Novedades la generación programada se ejecutará desde la forma de generación de novedades. Adicionalmente, se habilitarán los medios para permitir que esta ejecución sea programada.

Los parámetros necesarios para la generación programada de novedades son los mismos requeridos en la generación asistida, solo que en este caso el campo fecha y el campo hora de ejecución deben ser ingresados por el usuario y determinarán el momento de inicio de la ejecución del proceso.

Para este fin, la ejecución del proceso de generación de novedades se programará a través de jobs, empleando la utilidad de Jobs de OpenSmartFlex.

El comando de ejecución del proceso tendrá la siguiente sintaxis:

```
PROCEDURE SUBMIT (
    Qué          IN   VARCHAR2,
    Fecha_sig    IN   DATE,
    Intervalo    IN   VARCHAR2 DEFAULT NULL,
    trabajo      OUT BINARY INTEGER
    );
```

Donde:

- *trabajo*: Es el código de trabajo a programar.
- *Qué:* Es el llamado al package que comienza el proceso. (Con parámetros de proceso incluidos).
- *Fecha_sig:* Fecha en la que se va a ejecutar el proceso por primera vez (SYSDATE en el caso que se desee ejecutar inmediatamente).
- *Intervalo:* Función que calcula cuando se ejecutará el proceso de nuevo (NULL en el caso que se desee ejecutar una sola vez).

La información generada por el proceso se almacenará en la tabla de la base de datos descrita en el ítem 7.2.1. También se le generará un cargo al producto asociado a la novedad registrada.

En la generación programada de novedades también será posible generar información de un día que ya se haya generado o de un día que se haya generado de manera incompleta debido a una interrupción inesperada del proceso. En cualquiera de estos casos, el proceso deberá efectuar la eliminación de los registros correspondientes a la fecha de proceso enviada como parámetro (se eliminarán tanto las novedades como los cargos generados al producto) y enseguida generará la información correspondiente al día enviado.

7.2.5 Implementación del API se construirá un API a través del cual se habilitará la extracción de información de las novedades de un producto en particular. El API recibirá como parámetro de entrada un producto (servicio suscrito) y devolverá como resultado un cursor referenciado con todas las novedades de dicho producto encontradas en la entidad customizada. Los campos que devolverá el cursor serán los mismos campos definidos en la estructura de la tabla novedades descrita en el ítem 7.2.1.

7.2.6 Generación del Reporte de Novedades se creará un reporte que muestre las novedades existentes en un rango de fechas dadas, para este fin se empleará la utilidad Reports de Oracle. Dicho reporte se podrá generar detallado o resumido y estará agrupado por día. El contenido del reporte puede ser generado de las siguientes maneras:

- Formato archivo.
- Vista preliminar en pantalla.
- HTML.

Se desarrollará una forma por medio de la cual se permitirá generar el reporte, esta forma tendrá como parámetros de entrada:

- Fecha inicial: Día inicial del proceso.
- Fecha final: Día final del proceso.
- Flag tipo reporte: Permite identificar si el reporte a generar debe ser detallado o resumido.

Las columnas a listar en el reporte son:

- Día de proceso.
- Tipo de novedad: Código y descripción.
- Producto: Código del producto asociado a la novedad - Sólo en el reporte detallado.
- Nombre: Nombre del cliente que usa el producto – Sólo en el reporte detallado.
- Cantidad: La sumatoria del tipo novedad – Sólo en el reporte resumido.

7.3 GENERACIÓN DE ARCHIVOS PLANOS

OpenSmartFlex implementará los componentes necesarios para extraer la información generada sobre consumos medidos y novedades. El proceso de extracción de consumos medidos es independiente de la extracción de novedades, por lo tanto, cada proceso manejará sus propios componentes. El proceso de extracción de la información recibe como parámetro de entrada el día de proceso.

Se crearán dos formas: Una para generación de archivos de consumos y otra para la generación de archivos de novedades. Tanto para consumos medidos como

para novedades se permitirá generar archivos planos de manera asistida y también de manera programada, es por esta razón que las dos formas ya mencionadas le permitirán al usuario programar una fecha y hora de ejecución del proceso sí lo que se desea es que la generación se ejecute en un momento posterior, de lo contrario estos campos deben dejarse nulos.

Durante la extracción se implementará un proceso que realice un mapeo de la información, dicho proceso consistirá básicamente en tomar el contenido de ciertos campos de la entidad customizada a la cual se le va a extraer la información y pasarlo por un conversor que generará su respectivo equivalente. Posterior al proceso de mapeo, la información resultante será almacenada en un archivo plano. El objetivo de generar los archivos planos es registrar en ellos la información sobre consumos medidos y/o novedades para que el sistema externo Energis pueda disponer de dicha información.

Los archivos planos se crearán haciendo uso de la utilidad pkUtlFileMgr de OpenSmartFlex y serán almacenados del lado del servidor en un path* definido por el parámetro RUTA_ARCH_CONSMEDI (para el caso de los archivos de consumos medidos) y en el path definido por el parámetro RUTA_ARCH_NOVEDADS (en el caso de los archivos planos de novedades). Ambos parámetros se deben configurar en la entidad PARAMETR.

Los archivos tendrán la misma estructura que tiene los campos de las respectivas tablas (consumos medidos y novedades) teniendo en cuenta que:

- Los campos numéricos se alinearán a la derecha y los caracteres faltantes se completarán con espacios a la izquierda.
- Los campos alfanuméricos se alinearán a la izquierda y los caracteres faltantes se completarán con espacios en blanco.
- Las fechas se escribirán en formato YYYYMMDDHH24MISS sin ningún tipo de separador.
- Los campos de tabla que presenten componente decimal, al extraerlos hacia el archivo plano su valor será multiplicado por un factor calculado a partir de la cantidad definida para la parte decimal.

Ejemplo: Para un campo definido number (15,3) que tenga un registro con valor 500, el valor registrado en el archivo plano será 500000, resultado de multiplicar el valor quinientos (500) por mil (1000).

Adicionalmente OpenSmartFlex ofrecerá la posibilidad de que el operador del sistema pueda ejecutar cuantas veces desee el proceso de extracción para un día que ya ha sido generado.

* Ubicación o ruta dentro de la estructura de directorios del equipo

7.3.1 Generación asistida de archivos planos de consumos medidos la generación asistida de archivos planos de consumos medidos se ejecutará desde la forma de generación de archivos planos de consumos.

La forma recibirá como parámetros de entrada la fecha de proceso (correspondiente a la fecha de registro de consumo almacenada en la entidad customizada de consumos medidos) y los campos fecha y hora deben dejarse nulos para que la ejecución se realice inmediatamente.

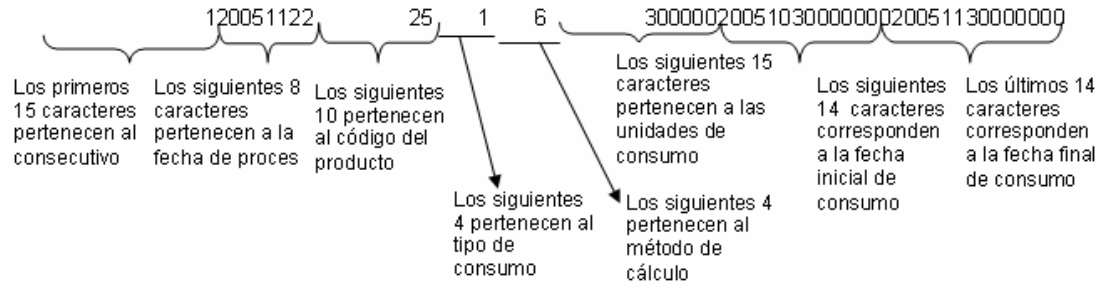
7.3.1.1 Mapeo de datos para la extracción de consumos medidos como ya se mencionó anteriormente, la información extraída de la entidad customizada debe pasar por un conversor para que le realice el correspondiente mapeo a los campos; en el caso de los consumos medidos ningún campo requiere ser mapeado.

7.3.1.2 Estructura del archivo plano de consumos medidos debido a que los campos de la entidad consumos medidos no requieren mapeo, la estructura del archivo plano será la misma que la estructura de la entidad de consumos medidos definida en el ítem 7.1.1. Cada campo de la entidad ocupará en el archivo plano los mismos caracteres definidos por la precisión del campo. Los campos alfanuméricos en el archivo plano se alinearán a la izquierda y los campos numéricos se alinearán a la derecha.

A continuación se muestra un prototipo del archivo plano generado para el día 22 de noviembre de 2005 con los siguientes registros en la entidad de consumos medidos:

Tabla 11. Prototipo de archivo plano de consumos medidos

CONSUMOS MEDIDOS							
Consecutivo	**Fecha proceso**	**Código producto**	**Tipo de consumo**	**Método de cálculo de consumo**	**unidades de consumo**	**Fecha inicial de consumo**	**Fecha final de consumo**
1	20051122	25	1	6	300	30/10/2005	30/11/2005
2	20050824	1256	1	1	1450	30/07/2005	30/08/2005
3	20051128	1285	2	4	855	30/10/2005	30/11/2005



Teniendo en cuenta que el proceso de extracción de información se puede realizar varias veces para un día ya generado, se considera que el nombre del archivo plano debe contener la hora de creación del mismo para así poder diferenciar entre diferentes archivos creados el mismo día de proceso y evitar que se sobrescriban unos a otros.

La nomenclatura del archivo plano de consumos medidos será la siguiente: <funcionalidad><fecha generada>_<fecha de ejecución del proceso><hora de ejecución>.<extensión>. Ejemplo: COME20051122_20060125175601.txt

7.3.2 Generación programada de archivos planos de consumos medidos A través de la forma de generación de archivos planos de consumos se permitirá configurar la fecha y la hora de ejecución del proceso, estos campos tienen como función permitirle al usuario definir un momento de ejecución posterior.

La ejecución del proceso de generación de archivos planos de consumos se podrá programar a través de jobs, empleando la utilidad de Jobs de OpenSmartFlex. El comando de ejecución del proceso tendrá la siguiente sintaxis:

```
PROCEDURE SUBMIT (
        Qué          IN    VARCHAR2,
        Fecha_sig    IN    DATE,
        Intervalo    IN    VARCHAR2 DEFAULT NULL,
        trabajo      OUT BINARY INTEGER,
);
```

Donde:

- *trabajo*: Es el código de trabajo a programar.
- *Qué:* Es el llamado al package que comienza el proceso. (Con parámetros de proceso incluidos).
- *Fecha_sig:* Fecha en la que se va a ejecutar el proceso por primera vez (SYSDATE en el caso que se desee ejecutar inmediatamente).
- *Intervalo:* Función que calcula cuando se ejecutará el proceso de nuevo (NULL en el caso que se desee ejecutar una sola vez).

El mapeo de la información y la estructura del archivo plano para la extracción programada de consumos medidos serán los mismos ya definidos en los puntos 7.3.1.1 y 7.3.1.2 para la extracción asistida.

7.3.3 Generación asistida de archivos planos de Novedades la generación asistida de archivos planos de novedades se ejecuta desde la forma de generación de archivos planos de novedades.

Esta forma recibirá como parámetro de entrada la fecha de proceso (correspondiente a la fecha de atención de la novedad almacenada en la entidad de novedades).

7.3.3.1 Mapeo de datos para la extracción de Novedades la información extraída de la entidad Novedades debe pasar por un conversor para que le realice el respectivo mapeo a los campos de la entidad; para el caso de las novedades los siguientes campos requieren ser mapeados:

- Código del producto
- Clase de servicio
- Clase de elemento de medición

Cada uno de estos tres campos tiene un número asociado de atributos de mapeo y una regla de mapeo; la generación de reglas de mapeo será configurable, es decir, se le permitirá al usuario definir las reglas de mapeo para cada uno de los campos que lo requieran.

Se creará una entidad que almacenará los diferentes identificadores de los campos a mapear con su respectiva descripción.

Tabla 12. Campos a mapear

CAMPOS A MAPEAR		
Campo	**Precisión**	**Acotación**
identificador del campo a mapear	Varchar2(2)	Identifica el campo susceptible a ser mapeado.

Descripción del campo	Varchar2(30)	Corresponde a la descripción del campo a mapear.

Igualmente se creará una entidad genérica de mapeo que permita almacenar las reglas de mapeo de cada uno de los campos que requieran ser mapeados.

Tabla 13. Valores de mapeo

VALORES DE MAPEO		
Campo	**Precisión**	**Acotación**
identificador del campo a mapear	Varchar2(2)	Indica a que campo pertenece el atributo de mapeo.
Código a mapear	15	Corresponde al código original que se desea mapear.
Orden del atributo	2	Indica el orden del atributo de mapeo.
Atributo de Mapeo	Varchar2(50)	Corresponde al valor con el que será reemplazado el código original en el archivo plano.

A continuación se muestra un ejemplo de cómo podría estar configurada la entidad de valores de mapeo:

Tabla 14. Valores a mapear

VALORES A MAPEAR			
identificador del campo a mapear	**Código a mapear**	**Orden del atributo**	**Atributo de Mapeo**
CS	12	1	a
CS	12	2	b
CS	12	3	cd
PR	50128	1	25
CE	1	1	001

Así pues, si en un registro de la entidad de novedades existe la clase de servicio 12, el producto 50128 y la clase del elemento de medición 1; en el archivo plano generado se imprimirá ‘abcd’, ‘25’ y ‘001’ en lugar de los códigos originales.

Adicionalmente se creará la forma de configuración de reglas de mapeo, a través de la cual el usuario podrá definir nuevas reglas de mapeo o actualizar atributos de mapeo ya existentes.

7.3.3.2 Estructura del archivo plano de novedades la estructura del archivo plano de novedades será la misma que la estructura de la entidad novedades

definida en el ítem 7.2.1. Cada campo de la entidad ocupará en el archivo plano los mismos caracteres definidos por la precisión del campo a excepción de los campos mapeados, los cuales ocuparán la cantidad de caracteres definida por la suma de sus atributos de mapeo (máximo 9 caracteres para la clase de servicio, 10 para el código del producto y 4 para la clase del elemento de medición), en el caso de que la suma de los atributos de mapeo tenga una cantidad de caracteres mayor a las ya especificadas, la cadena se truncará.

Los campos alfanuméricos en el archivo plano se alinearán a la izquierda y los campos numéricos se alinearán a la derecha.

Si no existe regla de mapeo para un código específico, en el archivo plano se registrará el código original de la entidad.

Al igual que en el proceso de extracción de consumos medidos, se ofrecerá la posibilidad de extraer información de novedades varias veces para un día ya generado, debido a esto, se considera que el nombre del archivo plano debe contener la hora de creación del mismo para así poder diferenciar entre diferentes archivos creados el mismo día de proceso y evitar que se sobrescriban unos a otros.

La nomenclatura del archivo plano de novedades será la siguiente:
<funcionalidad><fecha generada>-<fecha de ejecución del proceso>-<hora de ejecución>.<extensión>. Ejemplo: NOVE20051122_20060125140000.txt

7.3.4 Generación programada de archivos planos de novedades la forma de generación de archivos de Novedades permitirá además de ingresar la fecha de extracción, definir fecha y hora de ejecución del proceso.

La fecha y hora de programación definida por el usuario será la fecha exacta en la que el proceso se ejecute en el servidor; para realizar esta función se hará uso de jobs, empleando la utilidad de Jobs de OpenSmartFlex. El comando de ejecución del proceso tendrá la siguiente sintaxis:

```
PROCEDURE SUBMIT (
        Qué             IN    VARCHAR2,
        Fecha_sig   IN   DATE,
        Intervalo     IN   VARCHAR2 DEFAULT NULL,
        trabajo        OUT BINARY INTEGER,
);
```

Donde:

- *Trabajo*: Es el código de trabajo a programar.

- *Qué*: Es el llamado al package que comienza el proceso. (Con parámetros de proceso incluidos).
- *Fecha_sig*: Fecha en la que se va a ejecutar el proceso por primera vez (SYSDATE en el caso que se desee ejecutar inmediatamente).
- *Intervalo:* Función que calcula cuando se ejecutará el proceso de nuevo (NULL en el caso que se desee ejecutar una sola vez).

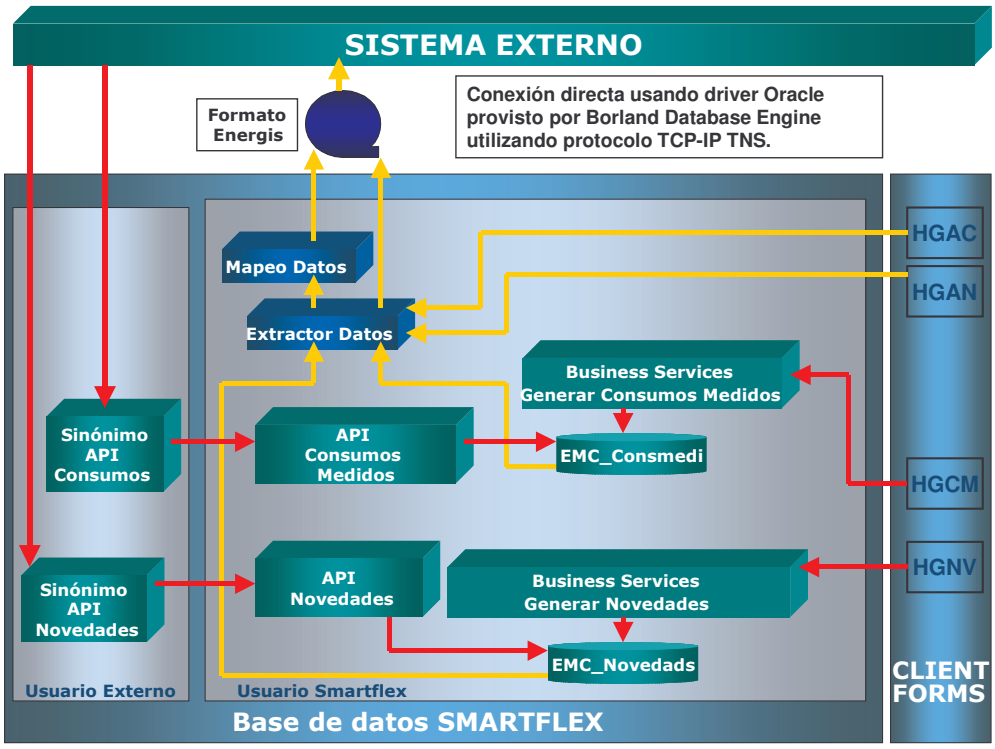
El mapeo de la información y la estructura del archivo plano para la extracción programada de Novedades serán los mismos ya definidos en los puntos 7.3.3.1 y 7.3.3.2 para la extracción asistida.

8. ARQUITECTURA DE LA SOLUCIÓN

8.1 ARQUITECTURA DE LA SOLUCIÓN

El sistema contará con una arquitectura de conectividad directa entre el sistema externo y OpenSmartFlex, a través del Driver ORACLE provisto por Borland Database Engine, utilizando protocolo TCP/IP - TNS.

Figura 5. Arquitectura de la solución



8.2 TRANSACCIONALIDAD

Los APIs de Consumos medidos y Novedades son de solo consulta, por lo tanto no tienen transaccionalidad.

Durante la generación tanto de consumos medidos como de novedades, se define que el proceso debe realizar commit cada 100 registros procesados.

En el caso de que se ejecute el proceso de generación de consumos medidos o de novedades para una fecha ya enviada anteriormente, se efectuará un reproceso.

La transaccionalidad que operará en el reproceso será la siguiente:

- Se eliminarán de la tabla customizada todos los registros que coincidan con los parámetros ingresados.
- Después de borrados los anteriores registros se dará paso a la inserción de los nuevos registros.
- Se realizará commit cada 100 registros durante la operación de borrado.
- Se realizará commit cada 100 registros durante la operación de inserción.

8.3 RESTRICCIONES

- La información que se suministrará a través de las APIs de novedades y consumos medidos no se mapeará. Los datos entregados por el API serán los mismos que estén en ese momento registrados en la entidad customizada.
- OpenSmartFlex no controlará que al momento de extraer información en archivos planos de un día en especifico no se encuentre en ejecución el proceso de generación de datos de ese mismo día. Si se extrae la información del día que se esta generando, simplemente se mostrará la información que hasta ese momento se haya procesado.

9. INTERFACES DE APLICACIÓN – API

Se proveerán APIs para habilitar el acceso de sistemas externos a la información sobre consumos medidos y a la información sobre novedades.

9.1 API CONSULTA DE CONSUMOS MEDIDOS

El API tendrá la siguiente especificación:

Tabla 15. API de consulta de consumos medidos

API – OS_EMC_GetCnsmpsByPrd					
Parámetros	**Tipo Dato**	**Precisión**	**Escala**	**E/S**	**Descripción**
Código de producto	Numérico	10		E	Código del producto (servicio suscrito).
Cursor Referenciado ORACLE	Cursor			S	Cursor referenciado de los consumos medidos del producto dado.
Consecutivo	Numérico	15		S	Código secuencial.
Fecha Proceso	Numérico	8		S	Fecha de registro de consumo. Será guardada en formato AAAAMMDD.
Código Producto	Numérico	10		S	Código del producto.
Tipo de consumo	Numérico	4		S	Tipo de consumo.
Método cálculo de consumo	Numérico	4		S	Método de cálculo de consumo.
Unidades de consumo	Numérico	15	3	S	Cantidad de unidades de consumo.
Fecha inicial de consumo	Date				Fecha inicial del periodo de consumo.
Fecha final de consumo	Date				Fecha final del periodo de consumo.
Código de Error	NUMBER	8		S	Código de error. Donde : 0 -> Termino con éxito <> 0 -> Termino con errores.
Mensaje de Error	VARCHAR2	2000		S	Mensaje de error. Donde : '-' -> Terminó con éxito Mensaje de error -> Terminó con errores.

La información de salida almacenada en el cursor referenciado será obtenida de la entidad customizada de consumos medidos. Los campos guardados en el cursor serán los mismos definidos en el ítem 7.1.1 en la estructura de la entidad consumos medidos.

9.2 API CONSULTA DE NOVEDADES

El API tendrá la siguiente especificación:

Tabla 16. API de consulta de novedades

API – OS_EMC_GetNovsByPrd						
Parámetros		**Tipo Dato**	**Precisión**	**Escala**	**E/S**	**Descripción**
Código de producto		Numérico	10		E	Código del producto (servicio suscrito).
Cursor Referenciado ORACLE		Cursor			S	Cursor referenciado de las novedades del producto dado.
	Consecutivo	Numérico	15		S	Código secuencial.
	Fecha proceso	Numérico	8		S	Fecha de atención de la novedad. Formato AAAAMMDD.
	Código producto	Numérico	10		S	Código del producto.
	Nombre	Varchar2	100		S	Nombre del usuario que usa el producto.
	Dirección	Varchar2	200		S	Dirección de instalación del producto.
	Teléfono	Varchar2	50		S	Número telefónico donde se encuentra instalado el producto.
	Ciclo de consumo	Numérico	4		S	Ciclo de consumo del producto.
	Sector	Numérico	4		S	Código del sector operativo al cual pertenece el producto.
	Ruta	Numérico	20		S	Código de la ruta a la cual pertenece el producto.
	Consecutivo de ruta	Numérico	15		S	Consecutivo del predio dentro de la ruta.
	Fecha novedad	Date			S	Fecha de atención de la novedad.
	Tipo novedad	Numérico	4		S	Tipo de novedad.
	Departamento	Numérico	4		S	Código de departamento del producto.
	Localidad	Numérico	6		S	Código de localidad del producto.
	Uso del servicio	Numérico	2		S	Corresponde a la categoría del producto.
	Estrato	Numérico	2		S	Sub-categoría del producto.
	Clase de servicio	Numérico	4		S	Código de la clase de servicio del componente medidor del producto.
	Consumo promedio	Numérico	15	3	S	Ultimo consumo promedio calculado para el producto.
	Carga instalada	Numérico	15	2	S	Carga declarada al momento de la instalación del producto.
	Serie medidor	Varchar2	50		S	Serie del elemento de medición
	Marca medidor	Varchar2	50		S	Marca del elemento de medición
	Modelo medidor	Varchar2	50		S	Modelo del elemento de medición
	Tipo elemento de medición	Numérico	4		S	Tipo del elemento de medición
	Clase elemento de medición	Numérico	4		S	Clase del elemento de medición
Código de Error		NUMBER	8		S	Código de error. Donde : 0 -> Termino con éxito <> 0 -> Termino con errores.
Mensaje de Error		VARCHAR2	2000		S	Mensaje de error. Donde : '-' -> Terminó con éxito Mensaje de error -> Terminó con errores.

La información de salida almacenada en el cursor referenciado será obtenida de la entidad customizada Novedades.

10. MODELO ENTIDAD RELACION

10.1 ENTIDADES DE BASES DE DATOS

10.1.1 Entidad nueva para consumos medidos

Tabla 17. Entidad EMC_CONSMEDI

Entidad : EMC_CONSMEDI – CONSUMOS MEDIDOS					
Atributos	**Tipo Dato**	**Precisión**	**Escala**	**Nulidad**	**Descripción**
Comdcodi	Number	15		N	Código secuencial
Comdfepr	Number	10		N	Fecha de proceso. Corresponde a la fecha de registro del consumo en formato AAAAMMDD
Comdsesu	Number	10		N	Código del producto de energía
Comdtico	Number	4		N	Tipo de consumo
Comdmecc	Number	4		N	Método de cálculo de consumo
Comdunco	Number	15	3	Y	Número de unidades del consumo
Comdfeic	Date			N	Fecha inicial de consumo
Comdfefc	Date			N	Fecha final de consumo

Donde:
Llave primaria: Comdcodi
Llaves foráneas:

- Comdsesu – referencia a servsusc.sesunuse
- Comdtico – referencia a tipocons.tconcodi
- Comdmecc – referencia a mecacons.mecccodi

Se crearán los siguientes índices en la tabla EMC_CONSMEDI:

- *ix_emc_consmedi01:* Índice por fecha de proceso. (Útil durante el reproceso y para la generación de archivos planos).
- *ix_emc_consmedi02:* Índice por producto. (Para extracción por API).

Adicionalmente se creará una secuencia en la base de datos con la cual se le asignará el valor al campo Comdcodi.

- *sq_Emc_Consmedi _Comdcodi*

10.1.2 Entidades nuevas para novedades

Tabla 18. Entidad EMC_NOVEDADES

Entidad : EMC_NOVEDADES – NOVEDADES					
Atributos	**Tipo Dato**	**Precisión**	**Escala**	**Nulidad**	**Descripción**
Novecodi	Number	15		N	Código secuencial.
Novefepr	Number	8		N	Fecha de proceso en formato AAAAMMDD. Corresponde a la fecha de atención de la novedad.
Novesesu	Number	10		N	Código del producto de energía.
Novessno	Varchar2	100		Y	Nombre del usuario que usa el producto.
Novedire	Varchar2	200		Y	Dirección de instalación del producto.
Novetele	Varchar2	50		N	Número telefónico donde se encuentra instalado el producto.
Novecico	Number	4		Y	Ciclo de consumo del producto.
Novesect	Number	4		Y	Código del sector operativo al cual pertenece el producto.
Noveruta	Number	20		Y	Código de la ruta a la cual pertenece el producto.
Novecoru	Number	15		Y	Consecutivo del predio dentro de la ruta.
Novefeno	Date			N	Fecha de atención de la novedad.
Novetino	Number	4		N	Tipo de novedad. Corresponde al código del tipo de novedad de la tabla novedades a publicar.
Novedept	Number	4		N	Corresponde al código del departamento del producto.
Noveloca	Number	6		N	Corresponde al código de localidad del producto.
Novecate	Number	2		N	Corresponde a la categoría del producto.
Novesuca	Number	2		N	Corresponde a la sub - categoría del producto.
Noveclse	Number	4		Y	Corresponde al código de la clase de servicio del componente medidor del producto.
Novecopr	Number	15	3	N	Ultimo consumo promedio calculado para el producto.
Novecain	Number	15	2	Y	Carga declarada al momento de la instalación del producto.
Noveseme	Varchar2	50		Y	Serie del elemento de medición.
Novemame	Varchar2	50		Y	Marca del elemento de medición.
Novemome	Varchar2	50		Y	Modelo del elemento de medición.
Novetiem	Number	4		Y	Tipo del elemento de medición.
Noveclem	Number	4		Y	Clase del elemento de medición.

Donde:
Llave primaria: Novecodi
Llaves foráneas:

- Novesesu - referencia a servsusc.sesunuse

- Novecico - referencia a ciclcons.cicocodi
- Novetino - referencia a ps_motive_type.motive_type_id
- Noveclem - referencia a claselme.clemcodi
- Novetiem - referencia a tipoelme.tiemcodi
- Novecate, novesuca – referencia a subcateg (sucacate, sucacodi)
- Novedept, noveloca – referencia a localida (locadepa, locacodi)

Se crearán los siguientes índices en la tabla EMC_NOVEDADS:
- *ix_emc_novedads01*: Índice por fecha de proceso. (Para generación de archivos planos y reproceso).
- *ix_emc_novedads02:* Índice por producto. (Para extracción por API).

Adicionalmente se creará una secuencia en la base de datos con la cual se le asignará el valor al campo Novecodi.
- *Sq_Emc_novedads_Novecodi*

Tabla 19. Entidad EMC_NOVEPUBL

Entidad : EMC_NOVEPUBL – NOVEDADES A PUBLICAR					
Atributos	**Tipo Dato**	**Precisión**	**Escala**	**Nulidad**	**Descripción**
Nopucono	Number	4		N	Código de la novedad.
Nopufaco	Number	4	2	N	Factor de cobro. Corresponde a la cantidad de salarios mínimos diarios a cobrar por la novedad.

Donde:
Llave primaria: Nopucono
Llave foránea: Nopucono – referencia a *ps_motive_type.motive_type_id*

10.1.3 Entidades nuevas para generación de archivos planos

Tabla 20. Entidad EMC_VALOPAME

Entidad : EMC_VALOMAPE – VALORES DE MAPEO					
Atributos	**Tipo Dato**	**Precisión**	**Escala**	**Nulidad**	**Descripción**
Vamacama	Varchar2	2		N	Identificador del campo a mapear.
Vamacoma	Number	15		N	Código original a mapear.
Vamaordn	Number	2		N	Orden del atributo de mapeo.
Vamavalr	Varchar2	50		N	Valor del atributo de mapeo.

Donde:
Llave primaria: vamacama, vamacoma, vamaordn.

Tabla 21. Entidad EMC_CAMPMAPE

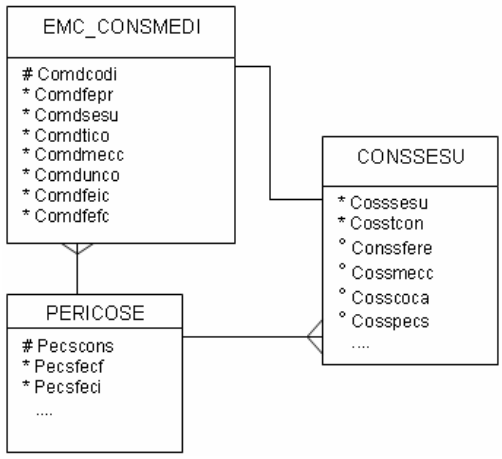
Entidad : EMC_CAMPMAPE – CAMPOS A MAPEAR					
Atributos	**Tipo Dato**	**Precisión**	**Escala**	**Nulidad**	**Descripción**
Camanomb	Varchar2	2		N	Identificador del campo a mapear.
Camadesc	Varchar2	30		N	Descripción del campo.

Donde:
Llave primaria: camanomb.

10.2 DIAGRAMA ENTIDAD RELACION

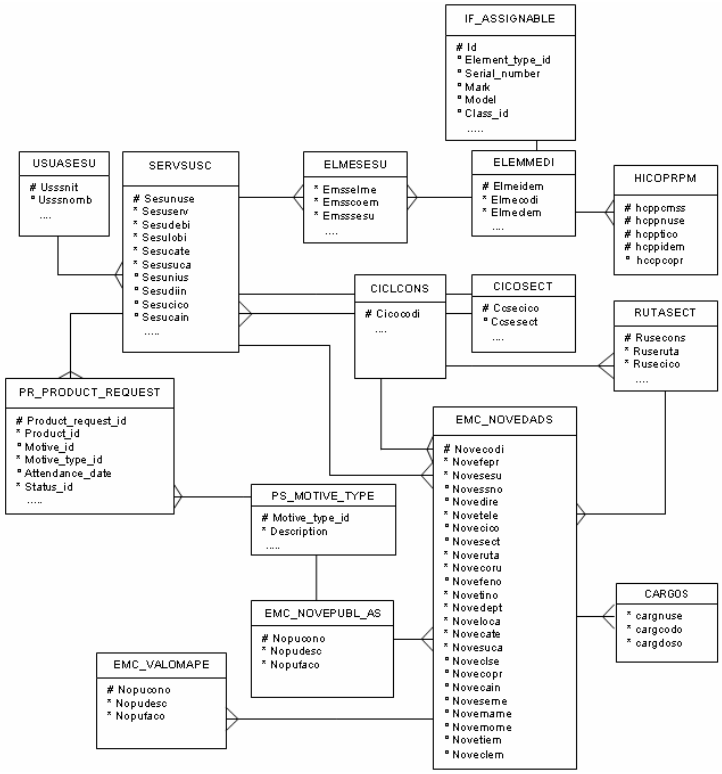
10.2.1 Diagrama entidad relación de consumos medidos

Figura 6. Diagrama entidad relación de consumos medidos



10.2.2 Diagrama entidad relación de novedades

Figura 7. Diagrama entidad de relación de novedades



11. INTERFACES DE USUARIO

11.1 DEFINICION Y ALCANCE

11.1.1 Formas nuevas

11.1.1.1 Forma de generación de consumos medidos se creará una forma a través de la cual se podrán generar los consumos medidos.

La funcionalidad de esta forma es registrar el proceso de generación para que se ejecute en una fecha y hora determinados por el usuario (en el caso que se desee programar la ejecución del proceso para un momento posterior), en el caso contrario estos campos deben dejarse nulos (el proceso se ejecuta de inmediato). A través de esta forma se podrá también consultar los procesos que están en cola de ejecución.

Al presionar el botón de Adicionar, la forma presentará otra pantalla que contará con tres pestañas en las cuales el usuario podrá configurar los parámetros del proceso:

- Tipo de producto
- Fecha de proceso
- Tipos de consumo
- Métodos de cálculo

Esta forma llamará a un proceso que finalmente insertará en la tabla de consumos medidos (emc_consmedi) los datos generados.

11.1.1.2 Formas de generación de novedades

11.1.1.2.1 Forma de novedades a publicar se creará una forma nueva por medio de la cual se permitirá configurar las novedades a publicar. La forma presentará el código de la novedad a publicar, la descripción de la misma y la cantidad de salarios mínimos a cobrar al cliente por la novedad.

Esta forma permitirá insertar, modificar y consultar registros.

11.1.1.2.2 Forma de generación de novedades se creará una forma a través de la cual se podrán generar las novedades de un día en específico.

La funcionalidad de esta forma es registrar el proceso de generación para que se ejecute en una fecha determinada por el usuario (en el caso que se desee programar la ejecución del proceso para un momento posterior), en el caso contrario los campos definidos para la fecha y hora se deben dejar nulos (el proceso se ejecuta de inmediato). A través de esta forma se podrá también consultar los procesos que están en cola de ejecución.

Al presionar el botón de Adicionar, la forma presentará otra pantalla donde el usuario podrá configurar los parámetros del proceso:

- Fecha de proceso
- Tipo de producto
- Tipo de consumo

Esta forma hará el llamado a un proceso que finalmente insertará en la tabla de novedades (Emc_Novedads) los datos generados.

11.1.1.3 Formas de generación de archivos planos

11.1.1.3.1 Formas de generación de archivos planos de consumos medidos se desarrollará una forma para la generación de archivos planos de consumos medidos. Esta forma recibirá la fecha a procesar y permitirá configurar la fecha y hora de ejecución del proceso (Campos nulos si se desea ejecutar el proceso inmediatamente), consultar los procesos ya programados y adicionar un proceso nuevo de generación de archivo plano.

Al presionar el botón adicionar, se ejecutará un proceso (el momento de ejecución depende de si es generación asistida o programada) que consultará en la tabla de consumos medidos (emc_consmedi) todos los consumos medidos registrados en la fecha ingresada como parámetro en la forma y posteriormente se generará un archivo plano con los registros obtenidos.

11.1.1.3.2 Forma de generación de archivos planos de novedades se creará una forma para la generación de archivos planos de novedades. Dicha forma recibirá la fecha a procesar y permitirá configurar la fecha de ejecución del proceso (Campo nulo si se desea ejecutar el proceso de inmediato) así como también adicionar un proceso nuevo de generación de archivo plano de novedades.

Al presionar el botón adicionar, se ejecutará un proceso (el momento de ejecución depende de si es generación asistida o programada) que consultará en la tabla de novedades (Emc_Novedads) todas las novedades que hayan sido atendidas en la fecha ingresada como parámetro en la forma y posteriormente se generará un archivo plano con los registros obtenidos.

11.1.1.3.3 Forma de configuración de reglas de mapeo se creará una forma que permita configurar las reglas de mapeo para cada campo de la entidad de novedades que requiera ser mapeado durante el proceso de extracción y generación archivos planos.

La forma es del tipo maestro detalle y en ella se permitirá visualizar los atributos de mapeo correspondientes a cada uno de los campos que se hayan configurado para ser mapeados. La forma permitirá consultar, insertar y actualizar las reglas de mapeo.

11.1.2 Reportes nuevos

11.1.2.1 Reporte de cantidad de unidades de consumo se contará con una nueva forma la cual solicitará los datos requeridos para la ejecución del reporte de cantidad de unidades de consumo. La forma a través de una casilla de selección le permitirá al usuario elegir sí el reporte a generar será detallado o resumido.

Los Prototipos de reportes de cantidad de unidades de consumo detallado y resumido son:

Figura 8. Reporte de consumos medidos resumido

EMCALI EMPRESA DE SERVICIOS PUBLICOS 01/09/2005 03:01:42
800-000-000 Página 1 de 1

REPORTE CANTIDAD DE UNIDADES DE CONSUMO

FGCM **Resumen**

FECHA INICIAL: 10/08/05
FECHA FINAL: 13/08/05

Dia de proceso 10/08/05

Tipo de consumo	Método de cálculo	Unidades de consumo
1 ENERGIA PICO	1 LEIDO	300
2 ENERGIA NO PICO	1 LEIDO	650
1 ENERGIA PICO	2 CORREGIDO	550
Total dia de proceso:		**1,500**

Dia de proceso 11/08/05

Tipo de consumo	Método de cálculo	Unidades de consumo
1 ENERGIA PICO	1 LEIDO	365
1 ENERGIA PICO	1 LEIDO	823
1 ENERGIA PICO	6 AFORADO	1212
Total dia de proceso:		**2,400**

Dia de proceso 13/08/05

Tipo de consumo	Método de cálculo	Unidades de consumo
2 ENERGIA PICO	5 RECUPERADO	825
2 ENERGIA PICO	1 LEIDO	1024
2 ENERGIA PICO	5 RECUPERADO	1275
Total dia de proceso:		**3,124**

Total reporte: **7,024**

Figura 9. Reporte de consumos medidos detallado

EMCALI EMPRESA DE SERVICIOS PUBLICOS 01/09/2005 17:01:42
800-000-000 Página 1 de 1

REPORTE CANTIDAD DE UNIDADES DE CONSUMO
FGCM **Detalle**

FECHA INICIAL: 10/08/05
FECHA FINAL: 13/08/05

Dia de proceso 10/08/05

Tipo de consumo	Método de cálculo	Producto	Unidades de consumo
1 ENERGIA PICO	1 LEIDO	1	300
2 ENERGIA NO PICO	1 LEIDO	12	650
1 ENERGIA PICO	2 CORREGIDO	3	550
Total dia de proceso:			**1,500**

Dia de proceso 11/08/05

Tipo de consumo	Método de cálculo	Producto	Unidades de consumo
1 ENERGIA PICO	1 LEIDO	3	365
1 ENERGIA PICO	1 LEIDO	24	823
1 ENERGIA PICO	6 AFORADO	12	1212
Total dia de proceso:			**2,400**

Dia de proceso 13/08/05

Tipo de consumo	Método de cálculo	Producto	Unidades de consumo
2 ENERGIA PICO	5 RECUPERADO	1	825
2 ENERGIA PICO	1 LEIDO	25	1024
2 ENERGIA PICO	5 RECUPERADO	1	1275
Total dia de proceso:			**3,124**

Total reporte: **7,024**

11.1.2.2 Reporte de novedades se contará con una nueva forma que solicitará los datos necesarios para la ejecución del reporte de Novedades. La forma a través de una casilla de selección le permitirá al usuario elegir sí el reporte a generar será detallado o resumido.

Los Prototipos de reportes de novedades detallado y resumido son:

Figura 10. Reporte de novedades resumido

EMCALI EMPRESA DE SERVICIOS PUBLICOS 1/10/2005 03:01:42 a.m

800-000-000 Página 1 de 1

REPORTE DE NOVEDADES

FGNV **Resumen**

FECHA INICIAL: 22/09/05
FECHA FINAL: 23/09/05

Dia de proceso 22/09/05

Tipo de novedad	Cantidad
1 Cambio de datos de usuario	10
2 Cambio de dirección	2
3 Cambio de medidor	2
4 Cambio ciclo de consumo	1
5 Cambio de estrato	1
6 Suspensiones de servicio	3
7 Retiro de servicio	4
8 Reconexión de servicio	2
9 Instalación de nuevo medidor	0
Total dia de proceso:	**25**

Dia de proceso 23/09/05

Tipo de novedad	Cantidad
1 Cambio de datos de usuario	2
2 Cambio de dirección	1
3 Cambio de medidor	1
4 Cambio ciclo de consumo	0
5 Cambio de estrato	3
6 Suspensiones de servicio	3
7 Retiro de servicio	7
8 Reconexión de servicio	1
9 Instalación de nuevo medidor	1
Total dia de proceso:	**19**

Total reporte: **44**

Figura 11. Reporte de novedades detallado

EMCALI EMPRESA DE SERVICIOS PUBLICOS 01/10/2005 03:06:42 a.m.
800-000-000 Página 1 de 1

REPORTE DE NOVEDADES
FGNV **Detalle**

FECHA INICIAL: 09/09/05
FECHA FINAL: 10/09/05

Dia de proceso 09/09/05

Tipo de Novedad	Producto	Usuario del producto
1 Cambio de datos de usuario	1	Andrés Sánchez C.
4 Cambio ciclo de consumo	23	
4 Cambio ciclo de consumo	1245	Diana Vega

Total dia de proceso: 3

Dia de proceso 10/09/05

Tipo de Novedad	Producto	Usuario del producto
1 Cambio de datos de usuario	5	Sandra López
2 Cambio de dirección	12	Pedro Méndez
3 Cambio de medidor	5	Sandra López
5 Cambio de estrato	5	Sandra López
5 Cambio de estrato	1	Andrés Sánchez C.
7 Retiro de servicio	128	

Total dia de proceso: 6

Total reporte: 9

11.2 PROTOTIPO NAVEGABLE

11.2.1 Forma de generación de Consumos Medidos

Primera pantalla: Permite configurar la fecha y hora de ejecución (Nulos si se quiere ejecutar el proceso inmediatamente), revisar los procesos en cola de ejecución y permite también configurar un nuevo proceso de generación de consumos medidos (Botón adicionar – llamado a la segunda pantalla).

Figura 12. Generación de Consumos Medidos

Segunda pantalla: Se compone de tres pestañas. Permite configurar los parámetros de entrada del proceso.

Figura 13. Programación de consumos medidos – Tipo de producto

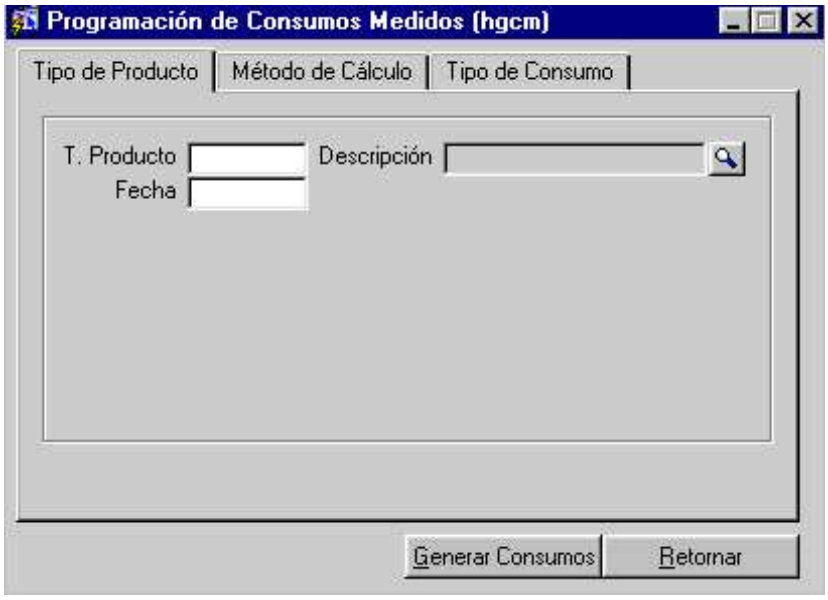
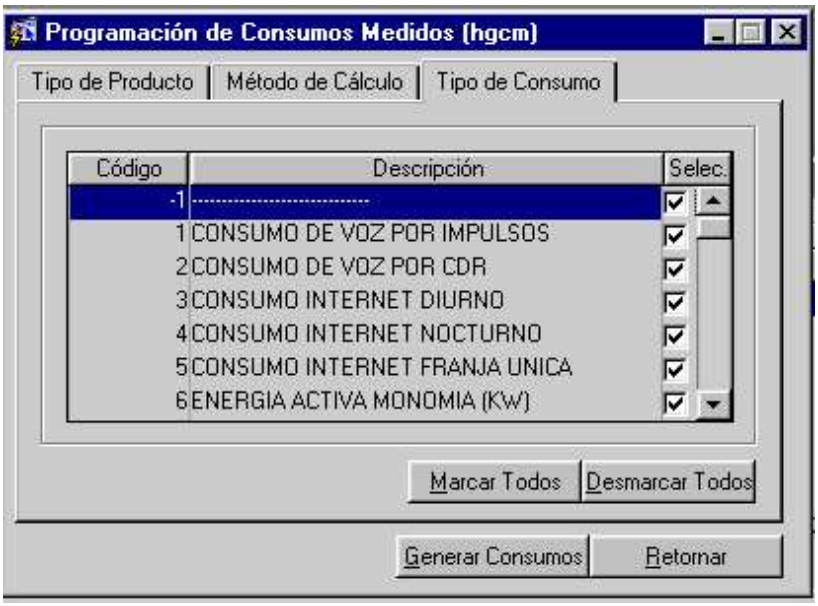


Figura 14. Programación de consumos medidos – Método de cálculo

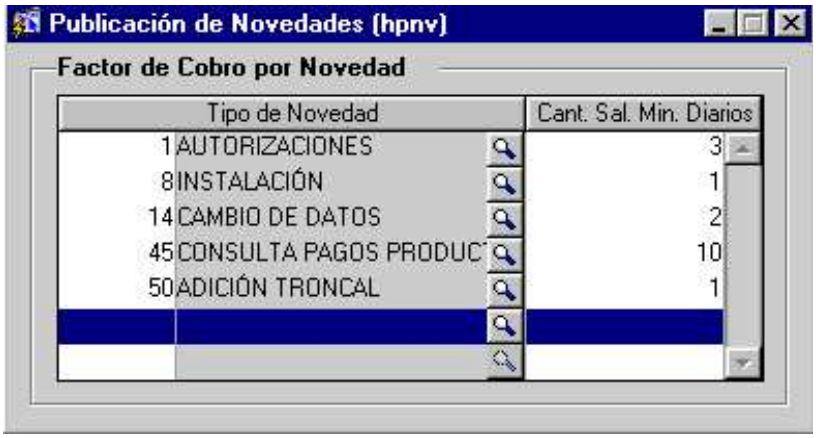
Figura 15. Programación de consumos medidos – Tipo de consumo



11.2.2 Formas de generación de Novedades

11.2.2.1 Forma de Novedades a publicar

Figura 16. Publicación de novedades



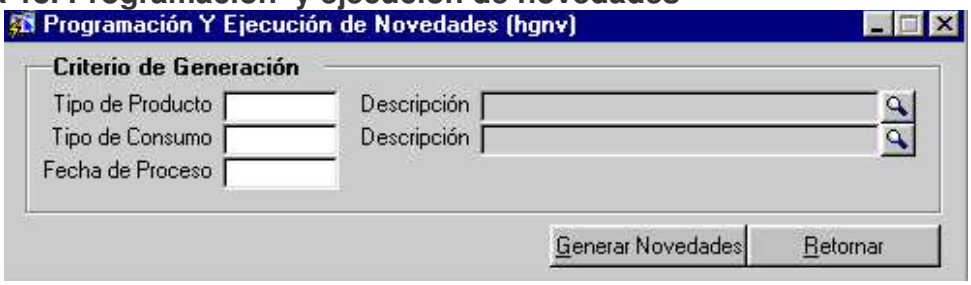
11.2.2.2 Forma de generación de Novedades

Primera pantalla: Permite configurar la fecha y hora de ejecución (Nulos si se desea ejecutar inmediatamente), revisar los procesos en cola de ejecución y permite también configurar un nuevo proceso de generación de novedades (Botón adicionar – llamado a la segunda pantalla).

Figura 17. Generación de novedades

Segunda pantalla: Permite configurar los parámetros de entrada del proceso.

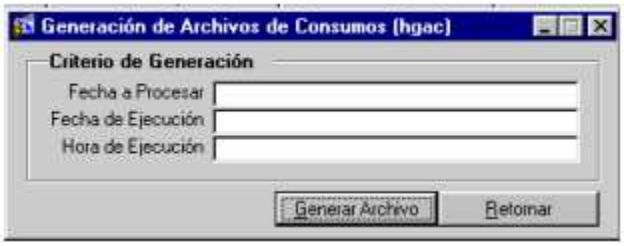
Figura 18. Programación y ejecución de novedades



11.2.3 Formas de generación de archivos planos

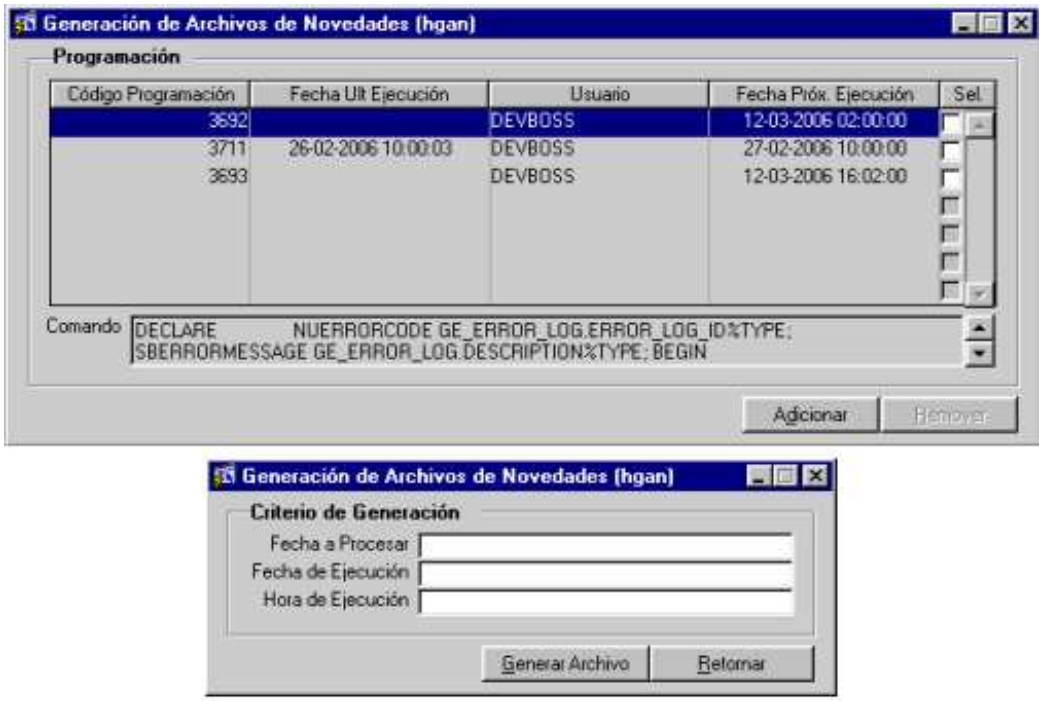
11.2.3.1 Forma de generación de archivos planos de Consumos Medidos

Figura 19. Generación de archivos de consumos medidos



11.2.3.2 Forma de generación de archivos planos de Novedades

Figura 20. Generación de archivos de novedades



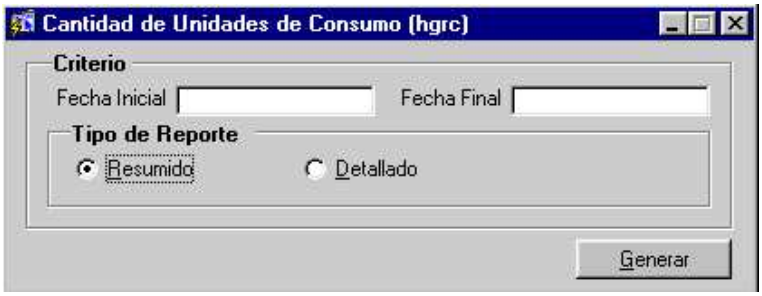
11.2.3.3 Forma de mapeo

Figura 21. Configuración de reglas de mapeo

11.2.4 Reportes

11.2.4.1 Reporte de cantidad de Unidades de Consumo

Figura 22. Cantidad de unidades de consumo



11.2.4.2 Reporte de Novedades

Figura 23. Reporte de novedades

12. CONFIGURACION FUNCIONAL

12.1 APLICATIVO

Debido a que el monto de dinero que se cobra en el cargo por concepto de la novedad se calcula en factor del salario mínimo legal diario, será necesario extraer el valor del salario mínimo legal mensual de la tabla de configuración del sistema (SISTEMA.SISTSAMI).

12.1.1 Configuración de objetos se debe crear un TOKEN en el paquete PKBILLCONS que permita identificar por medio del documento de soporte los cargos generados a razón de las novedades. El TOKEN será llamado: csbTOKEN_NOVEDADS.

12.1.2 Parámetros nuevos se creará el parámetro RUTA_ARCH_CONSMEDI el cual permitirá configurar la ruta del servidor donde se almacenarán los archivos planos de consumos medidos.

Se creará el parámetro RUTA_ARCH_NOVEDADS el cual permitirá configurar la ruta del servidor donde se almacenarán los archivos planos de novedades.

Cabe resaltar que las rutas indicadas en los anteriores parámetros deben existir, por lo tanto es preciso que se creen en el servidor los directorios necesarios y que adicionalmente se les otorguen a estos los permisos de lectura y escritura pertinentes.

Se creará un parámetro identificado como BIL_CONC_NOVEDAD el cual identificará el código del concepto asociado a los cargos derivados de las novedades.

Igualmente se crearán 3 parámetros nuevos para el proceso de extracción y mapeo de archivos planos de novedades, cada uno de estos parámetros identificara un campo a mapear.

- MAPEO_PRODUCTO
- MAPEO_CLASE_SERVICIO
- MAPEO_CLASE_ELEMENTO

12.1.3 Mensajes nuevos se creara un nuevo mensaje para el caso de los reprocesos, el cual será: “El proceso ya fue ejecutado para la fecha ingresada**.** Desea realizar un reproceso?”.

12.2 BASE DE DATOS

12.2.1 Creación de sinónimo se deben crear sinónimos de los siguientes objetos:

- OS_EMC_GetCnsmpsByPrd
- OS_EMC_GetNovsByPrd

12.2.2 Creación de usuario se debe crear un usuario Energis que tenga privilegios solo sobre los siguientes objetos:

- OS_EMC_GetCnsmpsByPrd
- OS_EMC_GetNovsByPrd

13. PROCESOS

Se identificaron 8 casos de uso:

- Generación de consumos medidos
- Extracción de información de consumos medidos a través del API
- Generación de archivo plano de consumos medidos
- Generación del reporte cantidad de unidades de consumo
- Generación de novedades
- Extracción de información de novedades a través del API
- Generación de archivo plano de novedades
- Generación del reporte de novedades

13.1 GENERACIÓN DE CONSUMOS MEDIDOS

Tabla 22. Generación de consumos medidos

Nombre caso de uso	1. Generación de consumos medidos
Descripción	Se generan los consumos medidos correspondientes a una fecha de proceso dada. El proceso se ejecuta desde la forma de generación de consumos medidos (HGCM) y puede programarse o ejecutarse de inmediato.
Guión	
Actor	**Sistema**
1. El usuario selecciona la fecha y hora en caso de que desee programar la ejecución. 2. El usuario presiona el botón Adicionar.	3. Despliega la pantalla de configuración de parámetros del proceso.

4. El usuario selecciona el servicio y la fecha de proceso. 5. El usuario selecciona los métodos de cálculo de consumo que desea incluir. 6. El usuario selecciona los tipos de consumo que desea incluir. 7. El usuario presiona el botón Generar.	8. Se crea un Job que dependiendo de lo ingresado en el campo Fecha y Hora ejecuta el proceso en un momento determinado. Si estos campos son nulos se ejecuta inmediatamente, de lo contrario se ejecutará en un momento posterior. 9. Valida que la fecha ingresada no sea mayor que la fecha actual. 10. Valida que exista el servicio entrante. 11. Transforma la fecha de proceso ingresada al formato numérico AAAAMMDD. 12. Revisa si se trata de un reproceso. 12.1 Elimina los registros de la entidad de consumos medidos que cumplan con lo siguiente: Tienen la misma fecha de proceso en formato AAAAMMDD que la fecha entrante. 12.2 Realiza la operación de commit cada 100 registros eliminados. 13. Extrae los servicios suscritos donde el código del servicio (SERVSUSC.SESUSERV) sea igual al servicio entrante. 14. Extrae los consumos por servicio suscrito donde: El código del servicio suscrito en la tabla de consumos (CONSSESU.COSSESU) sea igual a cada unos de los servicios suscritos extraídos en el paso anterior, El código del método de cálculo de consumo (CONSSESU.COSSMECC) sea igual a alguno de los métodos de cálculo de consumo seleccionados, El código del tipo de consumo (CONSSESU.COSSTCON) sea igual a alguno de los tipos de consumo seleccionados y donde la fecha de consumo (CONSSESU.COSSFERE) sea igual a la fecha de proceso ingresada. 15. Extrae para cada consumo obtenido en el paso anterior la fecha inicial y la fecha final del periodo de consumo (PERICOSE.PECSFECI y PERICOSE.PECSFECF). 16. Obtiene la fecha inicial de consumo. 16.1 Si la fecha de instalación del producto (SERVSUSC.SESUFEIN) se encuentra dentro del rango determinado por las fechas inicial y final del periodo de consumo, la fecha que se toma como fecha inicial de consumo será la fecha de instalación del producto. 16.2 En caso contrario, la fecha inicial de consumo será la fecha inicial del periodo de consumo (PERICOSE.PECSFECI). 17 Obtiene la fecha final de consumo. 17.1 Si el servicio suscrito se encuentra retirado y la fecha de retiro (SERVSUSC.SESUFERE) está dentro del rango determinado por las fechas inicial y final del periodo de consumo, la fecha que se toma como fecha final de consumo será la fecha de retiro del producto. 17.2 En caso contrario, la fecha final de consumo será la fecha final del periodo de consumo (PERICOSE.PECSFECF). 18 Almacena en la entidad de consumos medidos (EMC_CONSMEDI) para cada consumo obtenido: Un número consecutivo tomado de una secuencia en la base de datos, la fecha de proceso en formato numérico, el código del servicio suscrito, el código del tipo de consumo, el código del método de cálculo de consumo de cálculo de consumo, las unidades de consumo, la fecha inicial de consumo y la fecha final de consumo. 19 Realiza la operación de commit cada 100 registros insertados. 20 Termina.

13.2 EXTRACCIÓN DE INFORMACIÓN DE CONSUMOS MEDIDOS A TRAVÉS DEL API

Tabla 23. Extracción de información de consumos medidos API

Nombre caso de uso	2. Extracción de información de consumos medidos a través del API
Descripción	A través de un API, un sistema externo accede a la información de consumos medidos. El API recibe un código de producto y retorna los registros de consumos medidos asociados a dicho producto.
Guión	
Actor	**Sistema**
1. El sistema externo accede al API 2. El sistema externo ingresa un código de producto.	3. Valida que exista el código de producto ingresado. 4. Selecciona los registros de la entidad de consumos medidos (EMC_CONSMEDI) asociados al producto ingresado y los almacena en un cursor referenciado. 5. Retorna el cursor referenciado al sistema externo. 6. Termina.

13.3 GENERACIÓN DE ARCHIVO PLANO DE CONSUMOS MEDIDOS

Tabla 24. Generación del archivo plano de consumos medidos

Nombre caso de uso	3. Generación del archivo plano de consumos medidos
Descripción	Se genera un archivo plano con los consumos medidos de una fecha ingresada como parámetro. El proceso se ejecuta desde la forma de generación de archivo plano de consumos medidos (HGAC).
Guión	
Actor	**Sistema**
1. El usuario selecciona la fecha y hora de ejecución del proceso en el caso que desee que sea programado. 2. El usuario ingresa la fecha a procesar. 3. El usuario presiona el botón Generar.	4. Se crea un Job que dependiendo de lo ingresado en los campos Fecha y hora ejecuta el proceso en un momento determinado. Si el campo es nulo se ejecuta inmediatamente, de lo contrario se ejecutará en un momento posterior. 5. Valida que la fecha ingresada no sea mayor que la fecha actual. 6. Convierte la fecha ingresada al formato numérico AAAAMMDD. 7. Selecciona todos los registros de la entidad de consumos medidos en los que el campo fecha de proceso (EMC_CONSMEDI.COMDFEPR) sea igual a la fecha de proceso ingresada después de la conversión al formato AAAAMMDD. 8. Selecciona del parámetro RUTA_ARCH_CONSMEDI el path donde se almacenan los archivos planos. 1. Abre el archivo en la ruta obtenida y le asigna el siguiente formato de nombre: COME<fecha procesada>_<fecha creacion><hora>.txt Formato: COMEYYYYMMDD_YYYYMMDD24HHMISS.txt 9. Selecciona campo a campo cada registro obtenido en el paso anterior y lo escribe en el archivo plano (un registro por línea) teniendo en cuenta que cada campo en el archivo plano ocupará la misma longitud que la precisión del campo en la tabla de consumos medidos (EMC_CONSMEDI). 10. Cierra el archivo plano creado. 11. Termina.

13.4 GENERACIÓN DEL REPORTE CANTIDAD DE UNIDADES DE CONSUMO

Tabla 25. Generación del reporte de cantidad de unidades de consumo

Nombre caso de uso	4. Generación del reporte de cantidad de unidades de consumo
Descripción	Se genera en pantalla un reporte con la información sobre consumos medidos, el proceso recibe dos fechas y selecciona los consumos medidos que se encuentren dentro del rango de fechas dado. El proceso se ejecuta desde la forma de Generación de reporte de unidades de consumo (HGRC)
Guión	
Actor	**Sistema**
1. El usuario ingresa la fecha inicial. 2. El usuario ingresa la fecha final. 3. El usuario selecciona si desea reporte detallado o resumido. 4. El usuario presiona el botón Generar Reporte.	5. Valida que ninguna de las dos fechas sea nula. 6. Valida que la fecha final sea mayor que la fecha inicial. 7. Valida que la fecha inicial sea menor que la fecha actual. 8. Realiza un llamado a la forma (*grep)* para habilitar la configuración del reporte.
9. El usuario selecciona la configuración deseada y presiona el botón Generar reporte.	10. Realiza el llamado al reporte. 11. Selecciona los consumos de la entidad de consumos medidos donde su fecha de registro (EMC_CONSMEDI.COMDFEPR) se encuentre incluida en el rango de fechas recibidas como parámetro. 12. Muestra en pantalla el reporte. 13. Termina.

13.5 GENERACIÓN DE NOVEDADES

Tabla 26. Generación de novedades

Nombre caso de uso	5. Generación de Novedades
Descripción	Se generan las novedades correspondientes a una fecha de proceso dada. El proceso se ejecuta desde la forma de generación de novedades (HGNV) y puede programarse o ejecutarse de inmediato.
Guión	
Actor	**Sistema**
1. El usuario selecciona la fecha y hora en caso de que desee programar la ejecución. 2. El usuario presiona el botón Adicionar.	3. Valida que la fecha ingresada sea mayor que la fecha del sistema. 4. Despliega la pantalla de configuración de parámetros del proceso.

5	El usuario selecciona la fecha de proceso, el servicio y el tipo de consumo.	7	Valida que exista el servicio.
		8	Transforma la fecha de proceso ingresada al formato numérico AAAAMMDD.
6	El usuario presiona el botón Generar.	9	Revisa si se trata de un reproceso.
		9.1	Obtiene los consecutivos de los registros de la entidad EMC_NOVEDADS que se van a eliminar. (Aquellos que tienen la misma fecha de proceso en formato AAAAMMDD que la entrante).
		9.2	Elimina los cargos generados con motivo de las novedades a borrar. Estos cargos serán los que tengan documento de soporte 'NV<FECHA DE PROCESO>' (CARGOS.CARGDOSO) y consecutivo del documento (CARGOS.CARGCODO) igual al consecutivo obtenido de la novedad
		9.3	Elimina los registros de novedades ya procesados.
		9.4	Realiza la operación de commit cada 100 registros eliminados.
		10	Extrae los servicios suscritos donde el código del servicio (SERVSUSC.SESUSERV) sea igual al servicio entrante. (Granulado).
		11	Extrae las novedades asociadas al producto.
		11.1	Extrae de la entidad de peticiones por producto (PR_PRODUCT_REQUEST) las *novedades* asociadas a los productos seleccionados y que su fecha de atención sea igual a la fecha ingresada como parámetro.
		11.2	De las novedades obtenidas en el paso anterior, selecciona aquellas en las que el código del tipo de novedad en la entidad de peticiones por producto (PR_PRODUCT_REQUEST.MOTIVE_TYPE_ID) sea igual a alguno de los códigos de tipo de novedad configurados en la entidad de novedades a publicar (EMC_NOVEPUBL.NOPUCONO).

Tabla 27. Continuación de generación de novedades

Nombre caso de uso	5. Generación de Novedades
Descripción	Continuación del caso de uso número 5
Guión	
Actor	**Sistema**
	12 Obtiene el elemento de medición activo del servicio suscrito. 13 Extrae del elemento de medición por servicio suscrito, el número telefónico de donde se encuentra instalado el producto (ELMESESU.EMSSCOEM). 14 Extrae el último consumo promedio del elemento de medición asociado al producto (HICOPRPM.HCPPCOPR). 15 A partir del elemento de medición se obtiene el consecutivo de la ruta (CONSRUSE.CORSCOPR) y posteriormente el código de la ruta (RUTASECT.RUSERUTA). 16 A través del ciclo de consumo del producto extrae el sector operativo del producto (CICOSECT.CCSESECT). 17 A partir del producto obtiene el nombre del usuario del producto (USUASESU.USSSNOMB). 18 A partir del producto obtiene el nombre del usuario del producto (USUASESU.USSSNOMB). 19 Almacena en la entidad de novedades (EMC_NOVEDADS) para cada novedad obtenida: Un número consecutivo tomado de una secuencia en la base de datos, el código de la novedad (código correspondiente en la entidad EMC_NOVEPUBL), la fecha de proceso en formato numérico, los datos tomados del servicio suscrito (SERVSUSC), el nombre del usuario, el teléfono donde se encuentra instalado el producto, el último consumo promedio, el sector, la ruta, el consecutivo de ruta, la fecha de atención de la novedad (parámetro de entrada), la clase de servicio (nula sí la novedad es sobre el producto y no sobre un componente) y los datos del elemento de red medidor (nulos sí la novedad es sobre el producto y no sobre un componente). 20 Para cada novedad registrada, le genera un cargo al producto correspondiente. 20.1 Extrae el factor de cobro asociado al código de la novedad ocurrida. (EMC_NOVEPUBL.NOPUFACO) Cantidad de salarios mínimos diarios. 20.2 Selecciona el salario mínimo mensual vigente de la entidad SISTEMA (SISTEMA.SISTSAMI) y lo divide en 30 para obtener el valor del salario mínimo diario. 20.3 Multiplica el factor de cobro por el valor del salario mínimo diario obtenido y le genera un cargo al producto por el monto calculado. 20.4 El documento de soporte del cargo (CARGOS.CARGDOSO) se registrará como 'NV<FECHA DE PROCESO>' para identificar los cargos generados por las novedades y el consecutivo del documento (CARGOS.CARGCODO) será el mismo código consecutivo de cada novedad en la entidad EMC_NOVEDADS. 21 Realiza la operación de commit cada 100 registros procesados. 22 Termina.

13.6 EXTRACCIÓN DE INFORMACIÓN DE NOVEDADES A TRAVÉS DEL API

Tabla 28. Extracción de información de novedades a través del API

Nombre caso de uso	6. Extracción de información de novedades a través del API
Descripción	A través de un API, un sistema externo accede a la información de novedades. El API recibe un código de producto y retorna los registros de novedades asociados a dicho producto.
Guión	
Actor	**Sistema**
1. El sistema externo accede al API 2. El sistema externo ingresa un código de producto.	3. Valida que exista el código de producto ingresado. 4. Selecciona los registros de la entidad de novedades (EMC_NOVEDADS) asociados al producto ingresado y los almacena en un cursor referenciado. 5. Retorna el cursor referenciado al sistema externo. 6. Termina.

13.7 GENERACIÓN DEL ARCHIVO PLANO DE NOVEDADES

Tabla 29. Generación del archivo plano de novedades

Nombre caso de uso	7. Generación del archivo plano de novedades
Descripción	Se genera un archivo plano con las novedades de una fecha ingresada como parámetro. El proceso se ejecuta desde la forma de generación de archivo plano de novedades (HAPN).
Guión	
Actor	**Sistema**
1. El usuario presiona el botón adicionar para adicionar un nuevo Job.	2. Despliega la pantalla de configuración del proceso.

3. El usuario selecciona la fecha a procesar. 4. El usuario selecciona la fecha y hora en caso de que desee programar la ejecución. 5. El usuario presiona el botón Generar.	6. Crea un Job que dependiendo de lo ingresado en los campos fecha y Hora ejecuta el proceso en un momento determinado. Si el campo es nulo se ejecuta inmediatamente, de lo contrario se ejecutará en un momento posterior. 7. Convierte la fecha ingresada al formato numérico AAAAMMDD. 8. Selecciona todos los registros de la entidad de novedades en los que el campo fecha de proceso (EMC_NOVEDADS.NOVEFEPR) sea igual a la fecha de proceso ingresada después de la conversión al formato AAAAMMDD. 9. Selecciona del parámetro RUTA_ARCH_NOVEDADS el path donde se almacenan los archivos planos. 10. Abre el archivo en la ruta obtenida y le asigna el siguiente formato de nombre: NOVE<fecha procesada>_<fecha creación><hora>.txt 11. Formato: NOVEYYYYMMDD-YYYYMMDD-24HHMI.txt 12. Selecciona campo a campo cada registro obtenido en el paso anterior y lo escribe en el archivo plano (un registro por línea) teniendo en cuenta que cada campo en el archivo plano ocupará la misma longitud que la precisión del campo en la tabla de consumos medidos (EMC_NOVEDADS). 13. Mapeo de la información. 13.1 Al extraer de cada registro de novedad el código del producto (EMC_NOVEDADS.NOVESESU), busca en la entidad de mapeo (EMC_VALOMAPE) la clave de mapeo equivalente al código de producto registrado en la novedad; si encuentra regla de mapeo para ese código, escribe dicha clave de mapeo en el archivo plano en lugar de escribir el código original del producto, si no existe, escribe el código original. 13.2 Al momento de extraer de las novedades obtenidas la clase de servicio (EMC_NOVEDADS.NOVECLSE), busca en la entidad de mapeo (EMC_VALOMAPE) la clave de mapeo equivalente al código de la clase de servicio registrado en la novedad; si encuentra regla de mapeo para esa clase de servicio, escribe dicha clave de mapeo en el archivo plano en lugar de escribir el código original de la clase de servicio, si no existe, escribe el código original. 13.3 Al extraer de cada registro de novedad el código de la clase del elemento de medición (EMC_NOVEDADS.NOVECLEM), busca en la entidad de mapeo de la clase del elemento de medición (EMC_VALOMAPE) la clave de mapeo equivalente a la clase registrada en la novedad; si encuentra regla de mapeo para esa clase de elemento de medición, escribe dicha clave de mapeo en el archivo plano en lugar de escribir el código original de la clase de elemento de medición. 14 Cierra el archivo plano creado. 15 Termina.

13.8 GENERACIÓN DEL REPORTE DE NOVEDADES

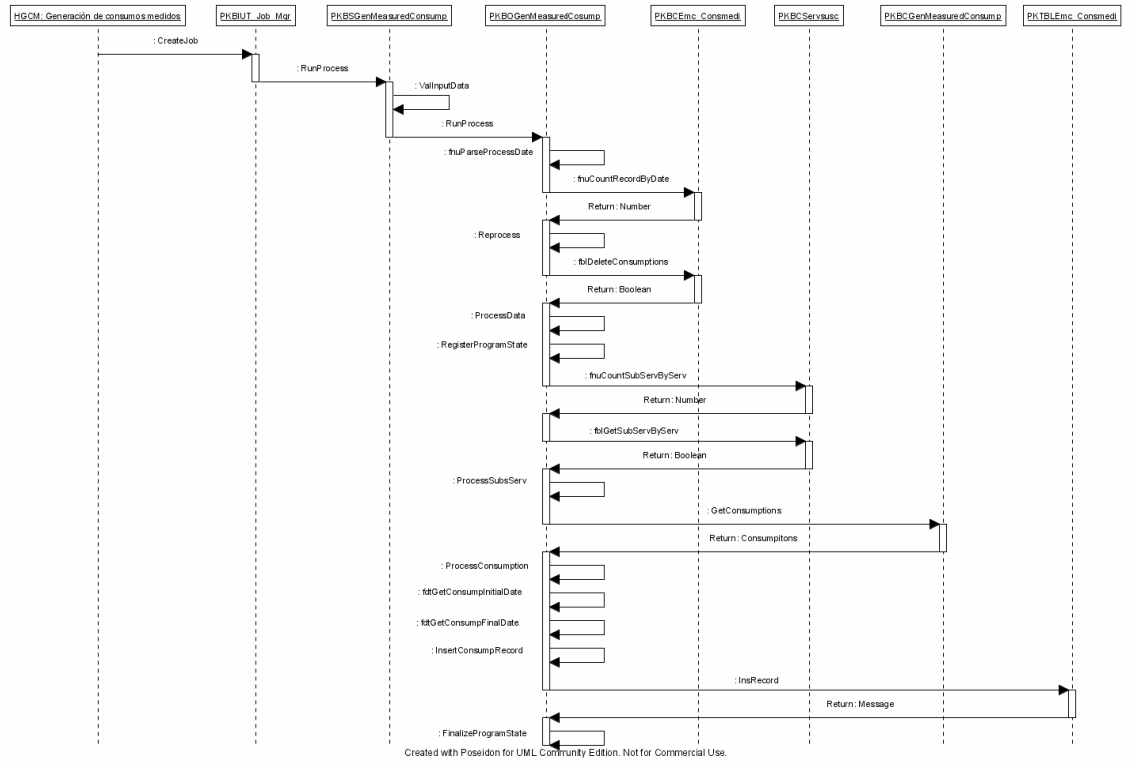
Tabla 30. Generación del reporte de novedades

Nombre caso de uso	8. Generación del reporte de novedades
Descripción	Se genera en pantalla un reporte con la información sobre novedades, el proceso recibe dos fechas y selecciona las novedades que se encuentren dentro del rango de fechas dado. El proceso se ejecuta desde la forma de Generación de reporte de novedades (HGRN).
Guión	
Actor	**Sistema**
1. El usuario ingresa la fecha inicial. 2. El usuario ingresa la fecha final. 3. El usuario selecciona si desea reporte detallado o resumido. 4. El usuario presiona el botón Generar Reporte.	5. Valida que la fecha final sea mayor que la fecha inicial. 6. Valida que la fecha inicial sea menor que la fecha actual. 7. Realiza un llamado a la forma *grep* para habilitar la configuración del reporte.
8. El usuario selecciona la configuración deseada y presiona el botón Generar reporte.	9. Realiza el llamado al reporte de novedades. 10. Selecciona las novedades de la entidad EMC_NOVEDADS donde su fecha de registro (EMC_NOVEDADS.NOVEFEPR) se encuentre incluida en el rango de fechas recibidas como parámetro. 11. Muestra en pantalla el reporte de novedades. 12. Termina.

14. DIAGRAMAS DE SECUENCIA

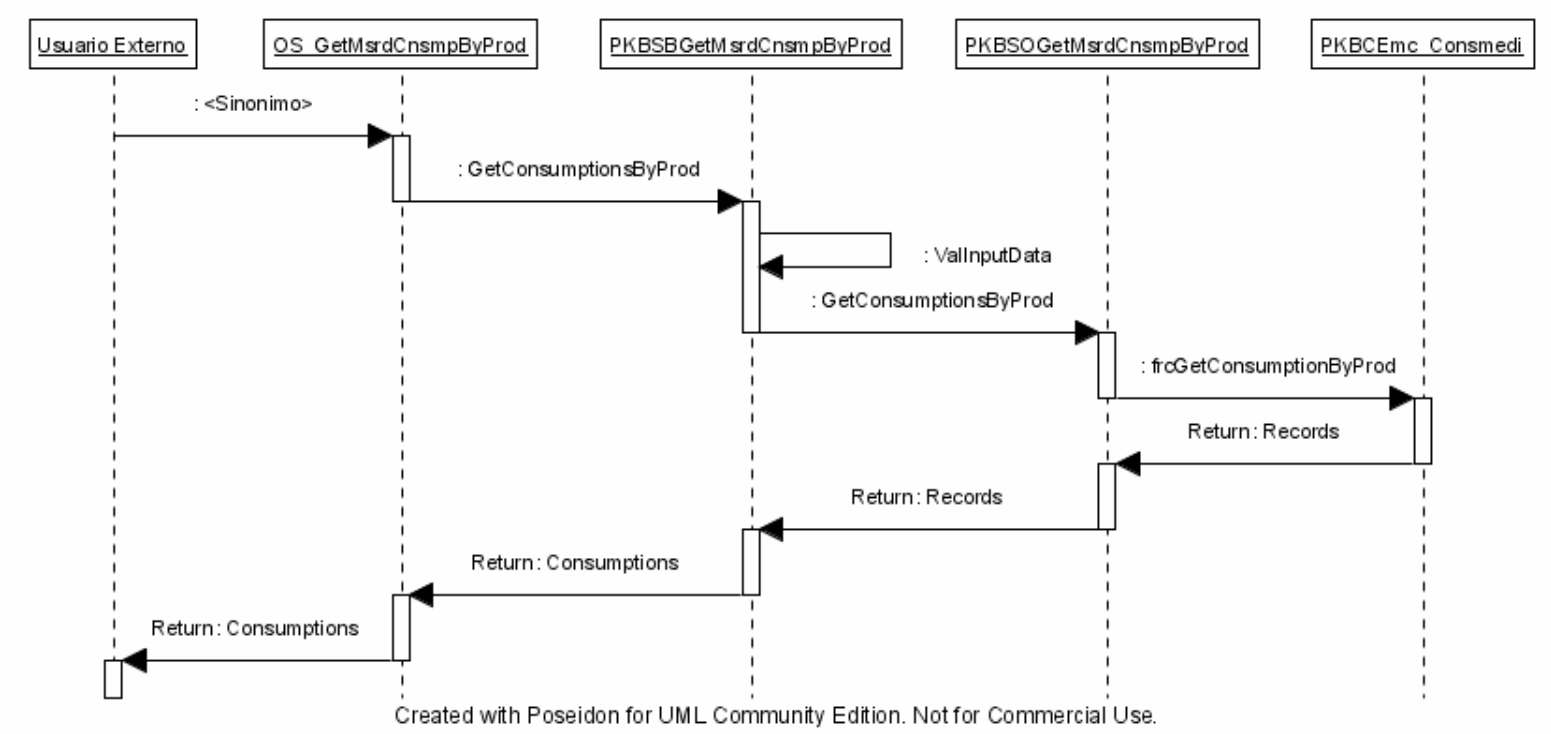
14.1 GENERACIÓN DE CONSUMOS MEDIDOS

Figura 24. Generación de Consumos Medidos



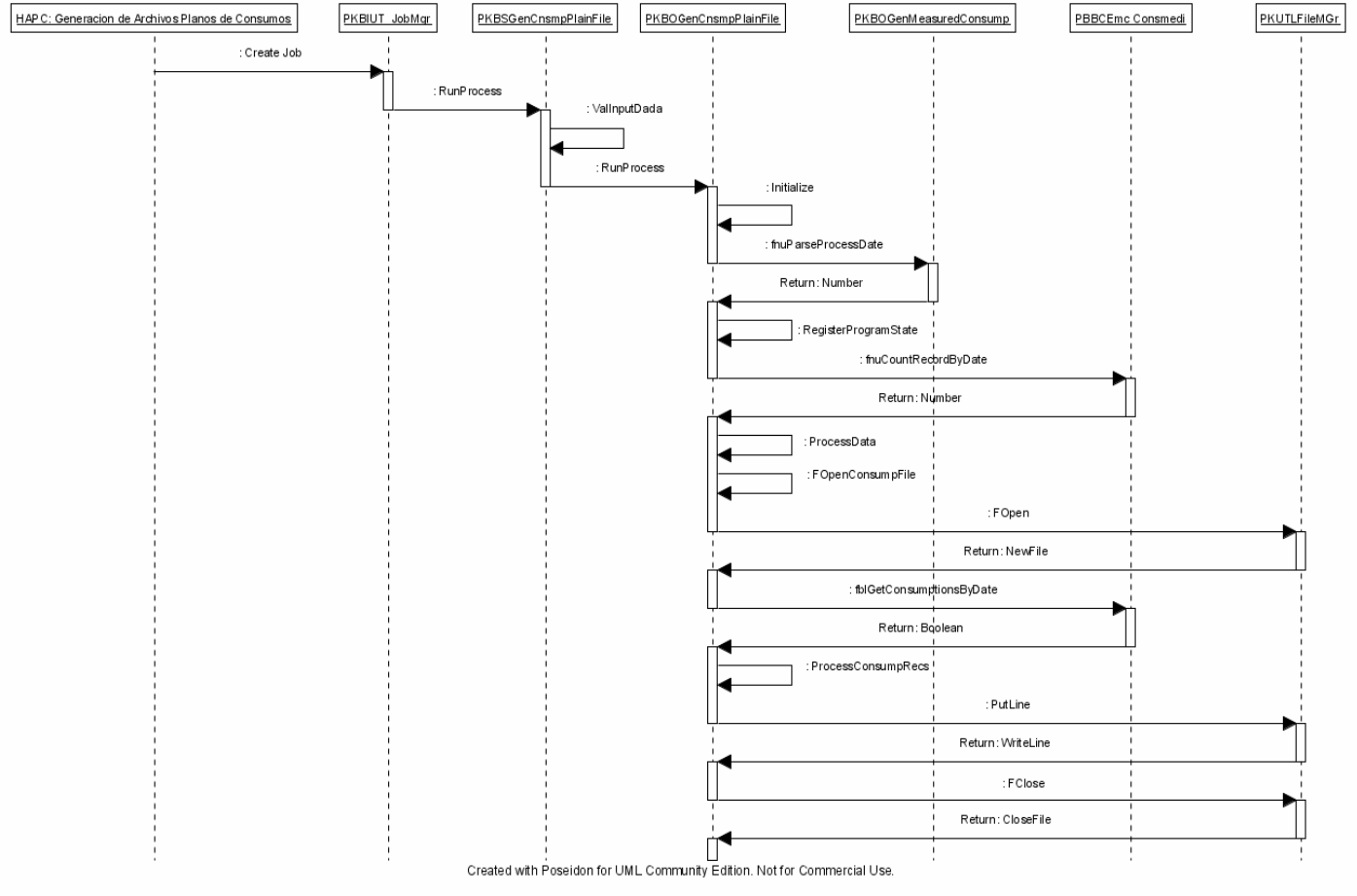
14.2 EXTRACCIÓN DE INFORMACIÓN DE CONSUMOS MEDIDOS A TRAVÉS DEL API

Figura 25. Extracción de información de consumos medidos a través del API



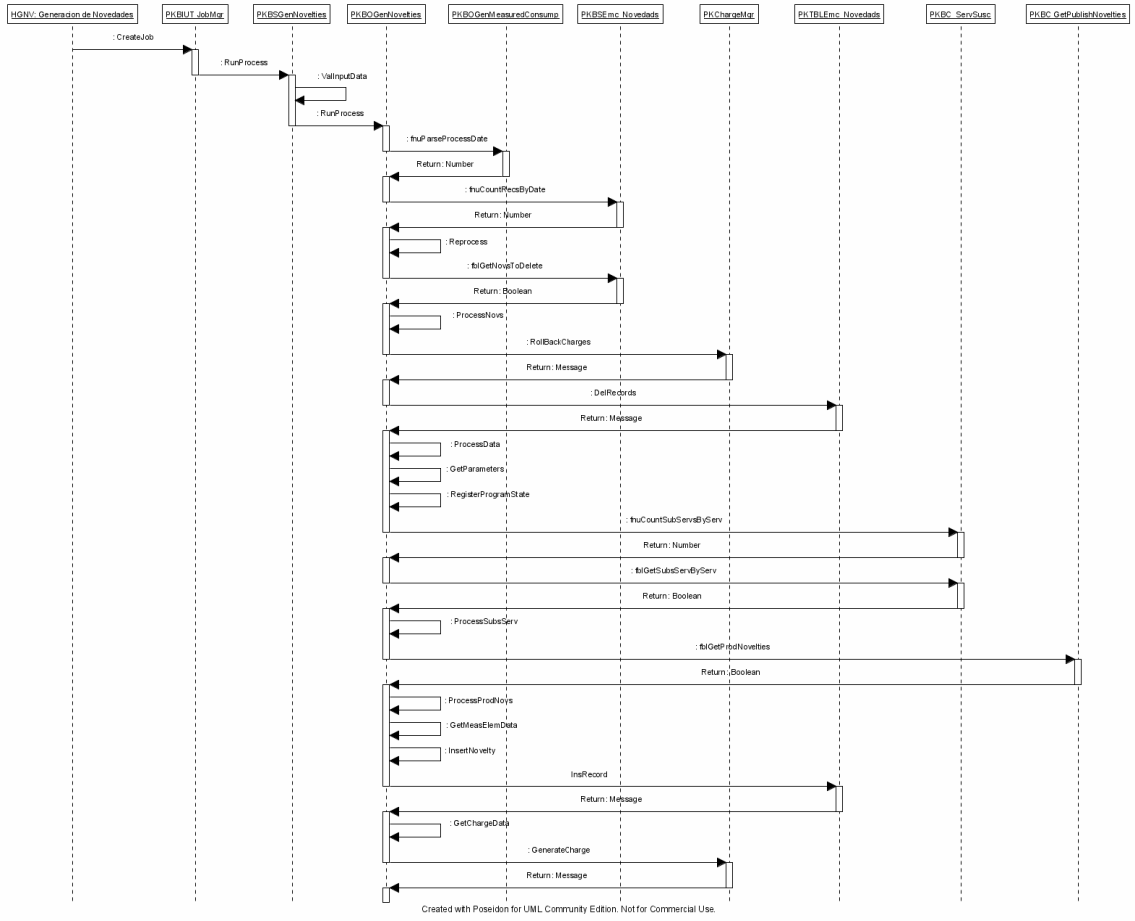
14.3 GENERACIÓN DE ARCHIVO PLANO DE CONSUMOS MEDIDOS

Figura 26. Generación del archivo plano de consumos medidos



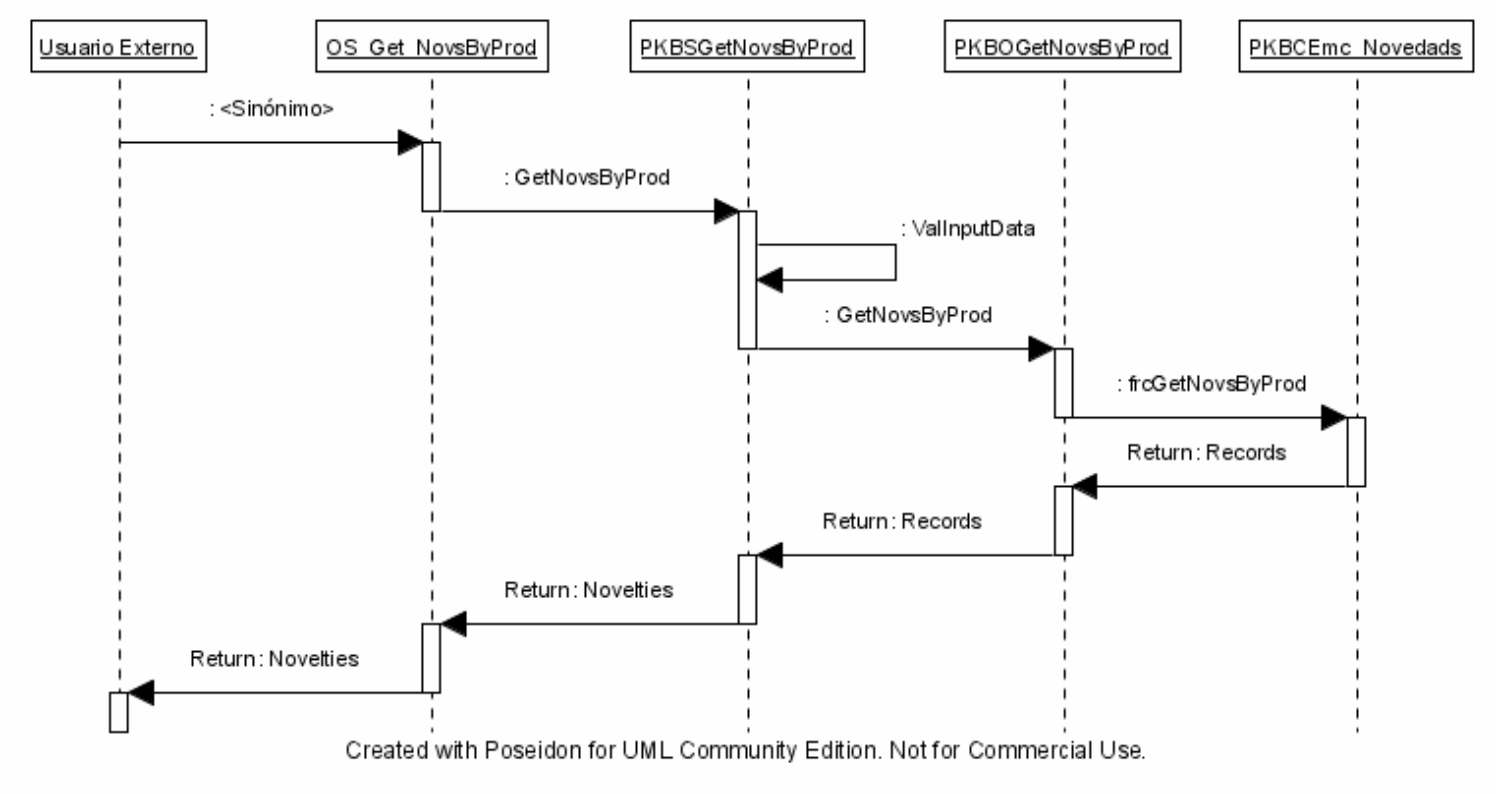
14.4 GENERACIÓN DE NOVEDADES

Figura 27. Generación de novedades



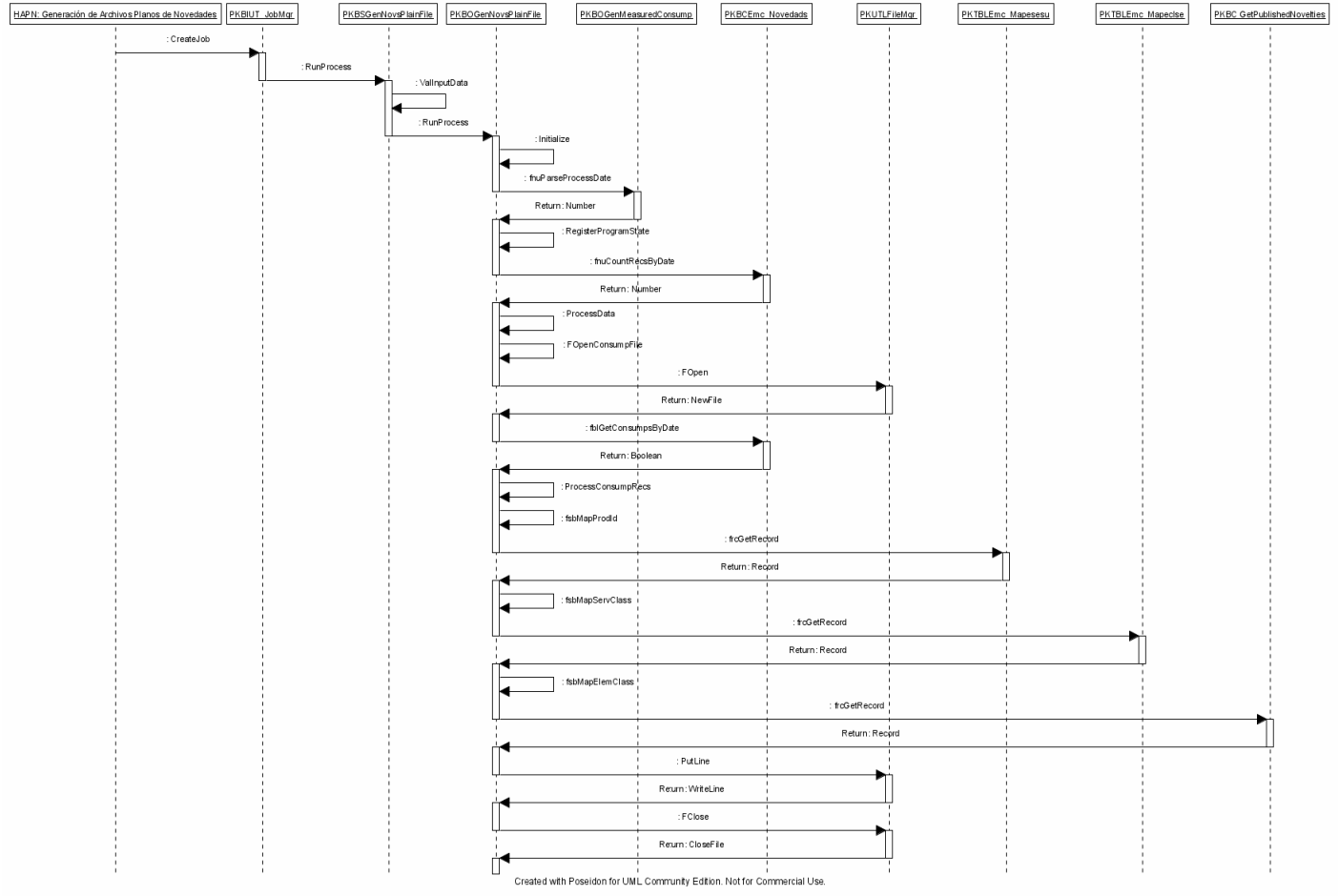
14.5 EXTRACCIÓN DE INFORMACIÓN DE NOVEDADES A TRAVÉS DEL API

Figura 28. Extracción de información de novedades a través del API



14.6 GENERACIÓN DEL ARCHIVO PLANO DE NOVEDADES

Figura 29. Generación del archivo plano de novedades



15. ANÁLISIS COMPARATIVO DE SENTENCIAS

A continuación se describe cada una de las tres soluciones implementadas y se explica detalladamente el plan de ejecución de la sentencia donde se realiza el acceso a los productos, esta sentencia es crítica dentro del proceso de generación de consumos puesto que de ella depende en gran medida la eficiencia o ineficiencia del proceso en cuestión.

La solución A corresponde al estudiante Julio Ledesma, la solución B al estudiante David Arias y las solución C al estudiante Andrés Sánchez.

15.1 SOLUCIÓN A

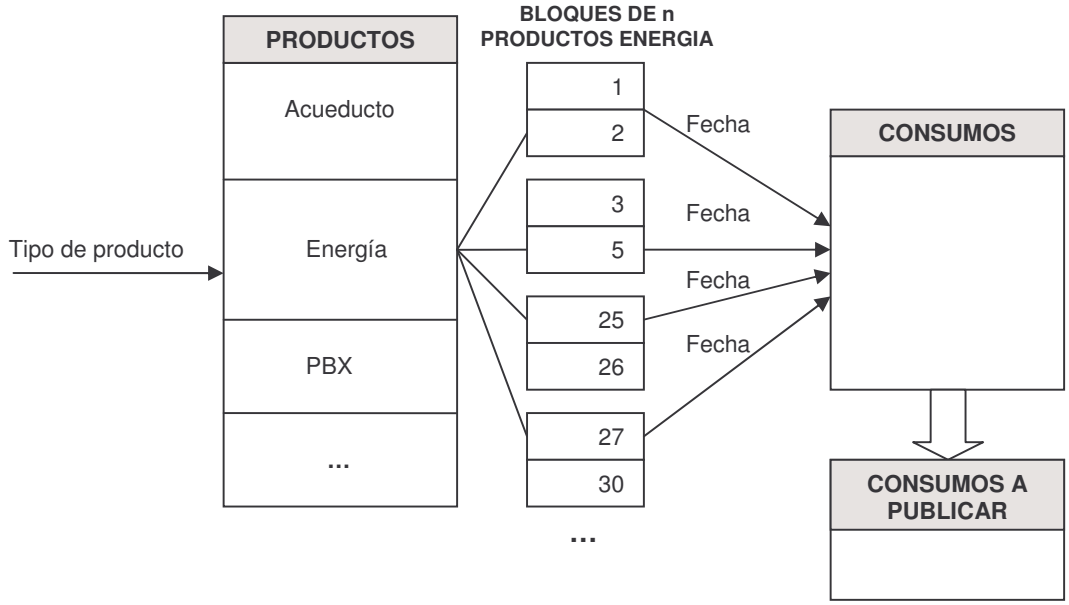
Parámetros de entrada del proceso:

- Tipo de producto
- Métodos de calculo de consumo
- Tipos de consumo
- Fecha de registro de los consumos

La solución A obtiene en bloques de n registros (granulación) los productos correspondientes al tipo de producto recibido como parámetro de entrada. El programa procesa los productos del primer bloque encontrando para cada uno de ellos los datos relacionados con los consumos registrados en la fecha ingresada como parámetro. Posteriormente, el programa inserta en la entidad de consumos medidos los consumos procesados y realiza la operación COMMIT al terminar de procesar el bloque de productos para hacer persistentes los datos en la base de datos.

Luego de terminar el procesamiento de los productos del primer bloque, el programa procede a obtener el siguiente grupo de productos y repite las operaciones ya mencionadas hasta que no existan mas bloques de productos por procesar.

Figura 30. Modelo de operación de la Solución A



15.1.1 Plan de ejecución para el acceso a productos

```
SELECT --+ INDEX (servsusc IX_SESU_SERV)
SESUNUSE, SESUFEIN, SESUFERE
FROM SERVSUSC
WHERE SESUSERV = :B2
AND SESUNUSE > :B1
ORDER BY SESUNUSE
```

Esta sentencia obtiene la información sobre productos (tabla SERVSUSC) pertenecientes al tipo de producto determinado por el parámetro de entrada del proceso (representado por la variable :B2).

El acceso a los productos en esta solución es granulado, por lo cual en la condición se utiliza un pivote (variable :B1) que determina el código del último producto procesado, esto con el objetivo de restringir el grupo de datos a trabajar y llevar un control de los registros que ya se han analizado.

Nota: La cantidad de registros a procesar por iteración (grupo de registros o BULK) es definido por el desarrollador durante la etapa de diseño dependiendo del tipo de proceso a construir y de la tabla a la cual se accede granuladamente, también puede basarse en estimaciones realizadas anteriormente en la empresa Open Systems para definir esta cantidad.

Específicamente para este desarrollo se definió un BULK de 100 registros.

Por otro lado, enseguida de la instrucción SELECT se usa un HINT con el objetivo de indicarle al motor de ORACLE el mejor camino para ejecutar la sentencia; los HINTS se usan de la siguiente forma:

SELECT --+(<Tabla a acceder> <Nombre del índice>)

En este caso, el HINT le indica al motor de ORACLE que debe utilizar el índice IX_SESU_SERV (Índice por tipo de producto) para resolver la consulta sobre productos.

Cabe resaltar que el motor de ORACLE recibe los HINTS como una sugerencia para determinar la manera de resolver una sentencia, pero en ningún momento el motor está obligado a utilizar determinado índice sí para su "juicio" existe una posibilidad más eficiente. En estos casos en los que el motor no sigue el camino indicado por el HINT, se utiliza una técnica para "engañar" al motor de la base de datos, esta técnica consiste en realizar operaciones (sumar cero a campos numéricos o concatenar una cadena vacía a un campo VARCHAR2) sobre los campos incluidos en el índice que ORACLE cree óptimo, de esta manera se evita que ORACLE decida el índice a utilizar pues él no está en capacidad de utilizar índices donde los campos que lo componen presenten operaciones.

Finalmente, la cláusula ORDER BY presente en la sentencia se usa para garantizar que no se omita ningún producto al momento de seleccionar el grupo de registros a procesar, ya que al usar el índice IX_SESU_SERV se acceden los productos por tipo de producto pero no se asegura que los productos obtenidos se encuentren ordenados por código.

El plan de ejecución de la sentencia mostrado por la herramienta TKPROF es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
  30070  SORT ORDER BY
4536070   TABLE ACCESS BY INDEX ROWID SERVSUSC
9051070    INDEX RANGE SCAN IX_SESU_SERV (object id 567541)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
  30070   SORT (ORDER BY)
4536070    TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF 'SERVSUSC'
9051070     INDEX   GOAL: ANALYZED (RANGE SCAN) OF 'IX_SESU_SERV'
                (NON-UNIQUE)
```

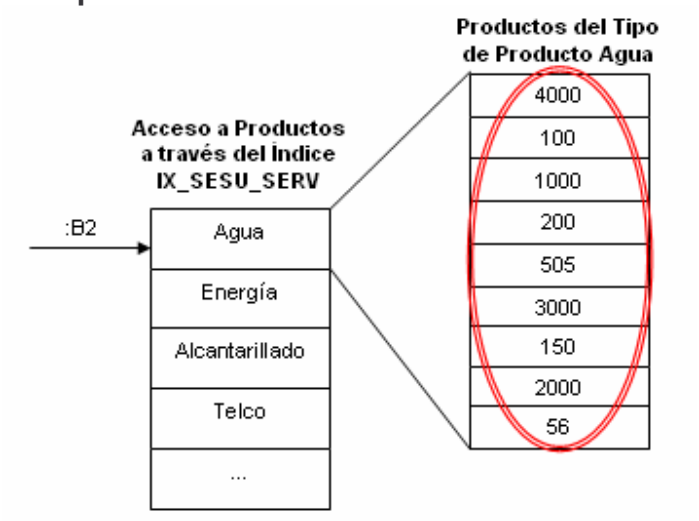
Interpretación del plan de ejecución:

- INDEX RANGE SCAN IX_SESU_SERV: El motor de ORACLE respeta el HINT usado, así que resuelve la sentencia a través del índice IX_SESU_SERV, de esta manera se hace un acceso por rangos a los registros de productos (RANGE SCAN).

Al resolver una sentencia por rangos, se tiene la seguridad de no realizar un barrido de todos los registros de la tabla (FULL SCAN) puesto que de entrada se descartan los registros que no corresponden al rango, en el caso de la sentencia en evaluación el rango lo determina el tipo de Producto.

Suponiendo que el tipo de producto ingresado como parámetro es Agua, el motor de ORACLE accederá únicamente el rango de registros correspondiente al tipo de producto Agua.

Figura 31. Acceso a productos en la solución A



- TABLE ACCESS BY INDEX ROWID SERVSUSC: Después de encontrar los productos correspondientes al rango, ORACLE obtiene el ROWID de cada registro (El ROWID está almacenado en el índice) y con este accede físicamente la tabla SERVSUSC para traer los campos solicitados en la sentencia para los registros que cumplen la condición:

AND SESUNUSE > :B1

Es decir, ORACLE solo accederá a disco por la información de los registros de producto cuyo código sea mayor al pivote (:B1). Cada vez que se ejecute esta sentencia el grupo de registros a acceder será menor.

• SORT (ORDER BY): Ejecuta el ordenamiento ascendente por código de producto para el conjunto de registros obtenido en el paso anterior.

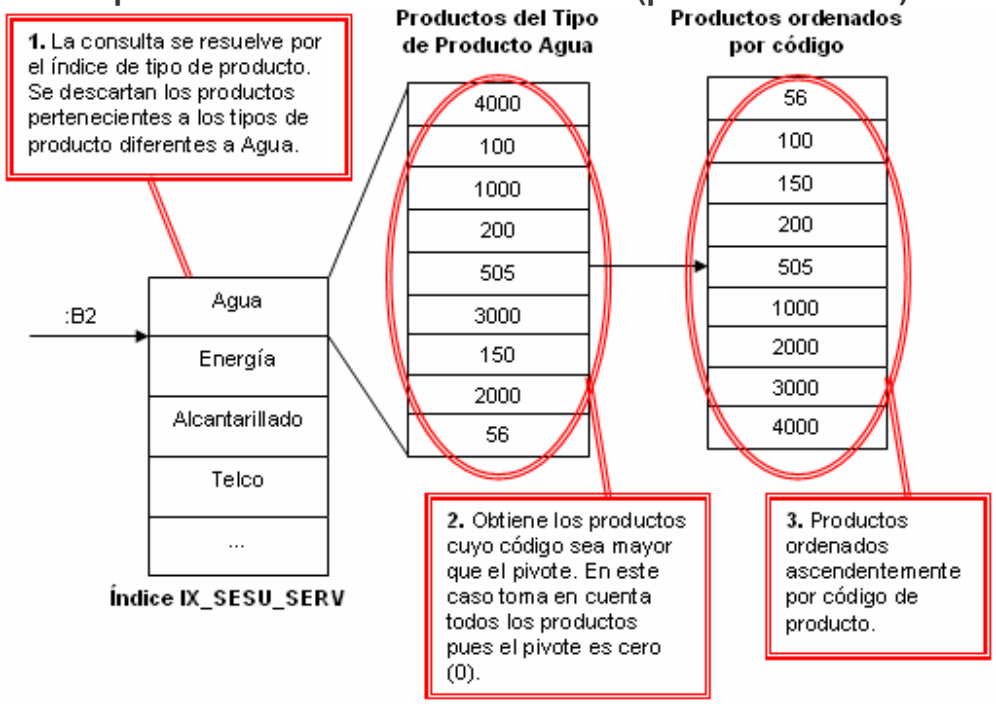
Con el siguiente ejemplo se ilustra claramente el plan de ejecución para la sentencia de acceso a productos de la solución A:

Ejemplo 1: Acceso a productos

• Situación inicial (t0): Primer acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 0
BULK = 3

Figura 32. Operación de la solución A – Caso 1 (primera iteración)



Análisis primera iteración t(0): Los registros de producto desplegados en la tabla del paso 3 se almacenan en un cursor para posteriormente obtener solo los N primeros registros de dicha estructura (la cantidad de registros a obtener la determina la constante BULK, 3 registros para efectos del ejemplo).

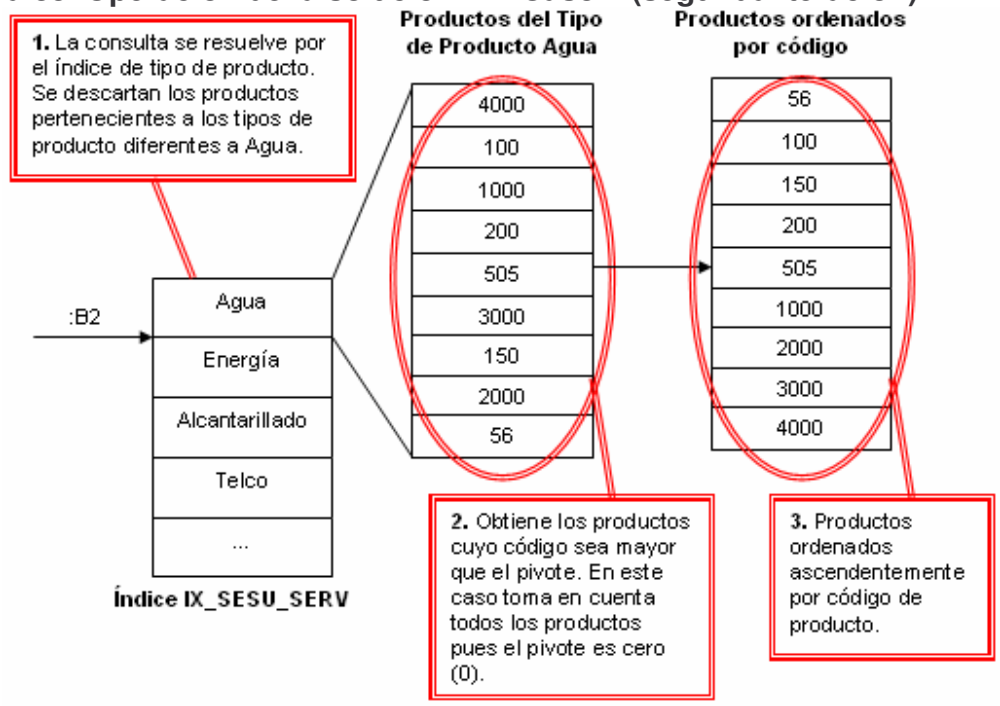
Los registros de producto obtenidos (56, 100 y 150) son almacenados en una estructura en memoria provista por ORACLE y para cada uno de ellos se obtienen los consumos asociados.

Al terminar de procesar los consumos de los tres registros, el último registro de producto presente en la estructura en memoria, se convierte en el producto pivote, es decir, el valor del producto pivote será 150.

- Segunda iteración (t1): Segundo acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 150
BULK = 3

Figura 33. Operación de la solución A – Caso 1 (segunda iteración)



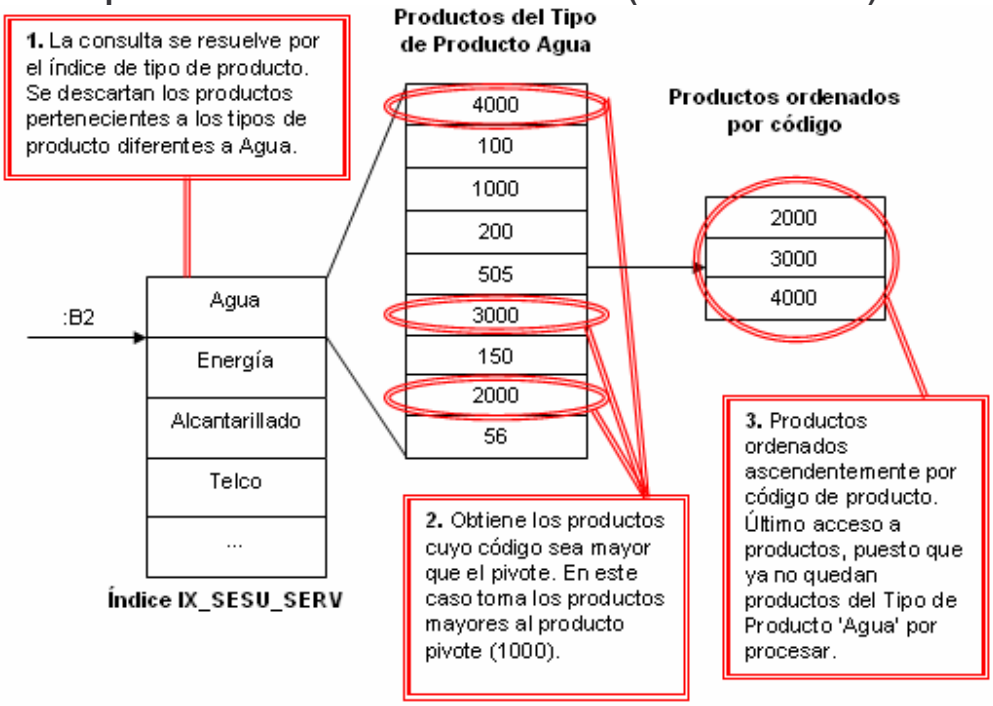
Análisis segunda iteración t(1): En esta iteración igualmente se obtienen del cursor los primeros tres registros de producto (200, 505 y 1000) y se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

Al terminar de procesar dichos consumos, el último registro de producto presente en la estructura en memoria, se convierte en el nuevo producto pivote, es decir, el valor del producto pivote será 1000.

- Tercera iteración (t2): Tercer acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 1000
BULK = 3

Figura 34. Operación de la solución A – Caso 1 (última iteración)



Análisis tercera iteración t(2): El resultado de la sentencia se almacena en el cursor, los primeros tres registros de producto presentes en el cursor (2000, 3000 y 4000) se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

Para este ejemplo la cantidad de productos obtenidos en el cursor coincide con la cantidad de registros a procesar en la estructura en memoria, sí la cantidad de registros obtenidos el cursor es menor a tres simplemente se procesan los consumos de los registros existentes, en el caso contrario, sí la cantidad de registros obtenidos el cursor es mayor a tres el programa se ve obligado a realizar una nueva iteración pues faltan productos por procesar.

Finalizando el procesamiento de los consumos para los productos 2000, 3000 y 4000; se actualiza el valor del pivote a 4000 y termina el acceso a productos pues ya no quedan más por procesar. Fin de lógica del programa

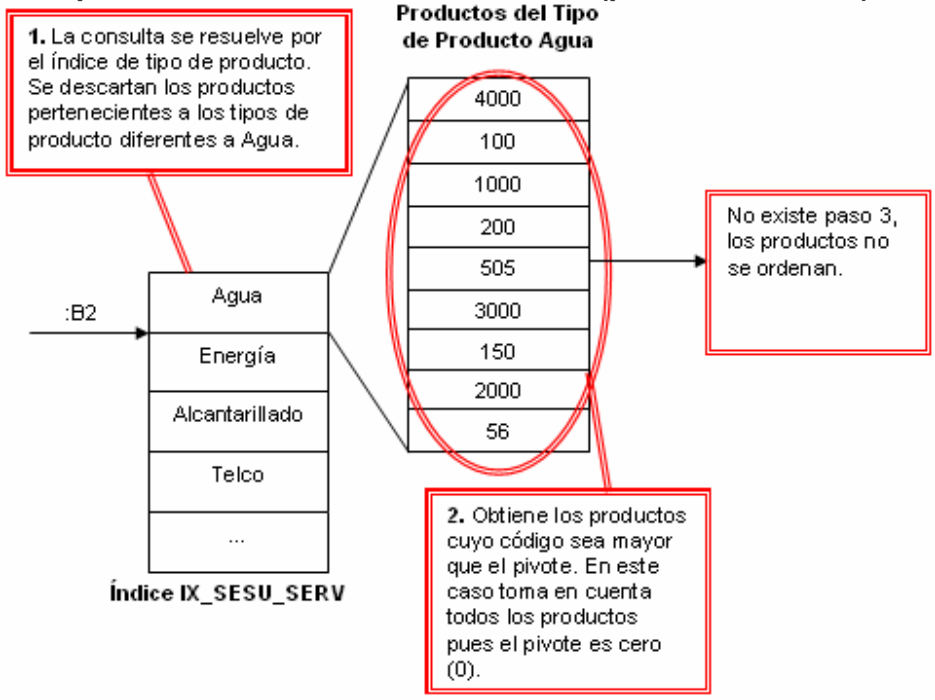
Ejemplo 2: Acceso a productos sin la cláusula ORDER BY

Con el fin de resaltar la importancia del uso de la cláusula ORDER BY en la sentencia de acceso a productos, a continuación se muestra el comportamiento de la solución A para el mismo ejemplo anterior dado el caso que no se utilice ORDER BY en la sentencia.

- Situación inicial (t0): Primer acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 0
BULK = 3

Figura 35. Operación de la solución A – Caso 2 (primera iteración)



Análisis primera iteración t(0): Los registros de producto desplegados en la tabla del paso 2 se almacenan en un cursor para posteriormente obtener solo los N primeros registros de dicha estructura (la cantidad de registros a obtener la determina la constante BULK, 3 registros para efectos del ejemplo).

Los registros de producto no se ordenan en el cursor por lo que los registros recuperados de éste para almacenarlos en la estructura en memoria son: 4000, 100 y 1000.

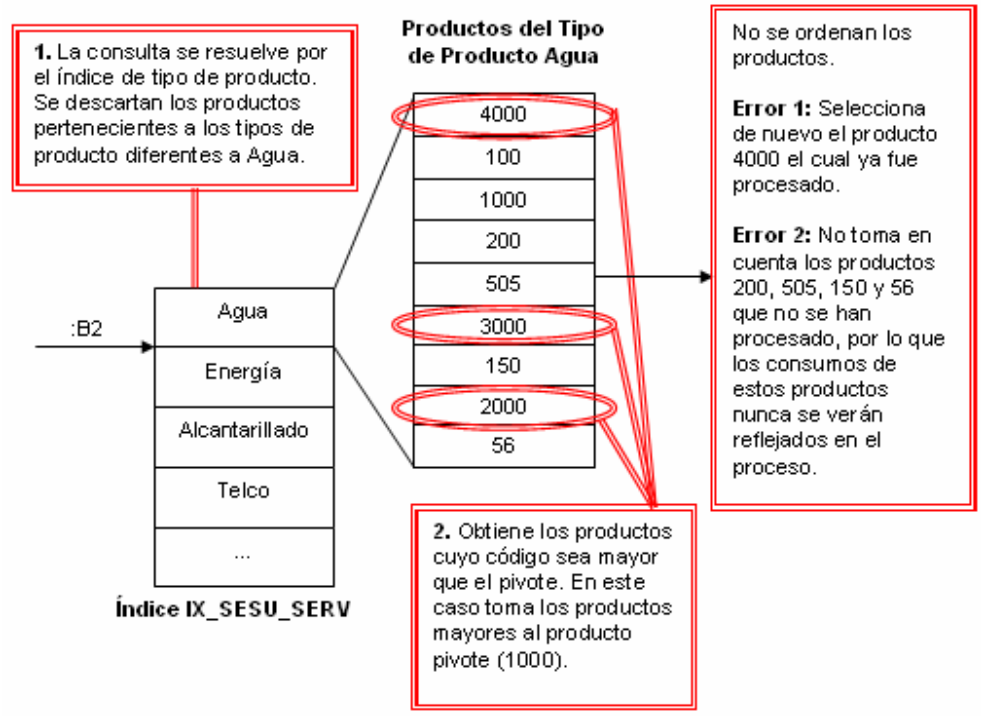
Al terminar de procesar los consumos de los productos 4000, 100 y 1000, el último registro de producto presente en la estructura en memoria, se convierte en el producto pivote, es decir, el valor del producto pivote será 1000.

- Segunda iteración (t1): Segundo acceso a los productos.

Tipo de producto (:B2) = Agua

Producto Pivote (:B1) = 1000
BULK = 3

Figura 36. Operación de la solución A – Caso 2 (segunda iteración)



Análisis segunda iteración t(1): Los registros de producto presentes en el cursor son: 4000, 3000 y 2000 (los registros mayores al pivote).

En el acceso a productos para esta iteración se presentan dos errores:

Error 1: Entre los productos seleccionados se encuentra el producto 4000. Este producto ya fue procesado en la iteración anterior, debido al error se generará información de consumos duplicada.

Error 2: Debido a que no se ordenó la consulta en la iteración anterior, el pivote terminó erróneamente con valor 1000 en el tiempo cero (t0), como consecuencia de esto la consulta en el tiempo dos (t1) no tuvo en cuenta registros menores a 1000 sin importar que estos no hayan sido procesados en iteraciones anteriores. Los registros de producto que quedaron sin procesar son: 200, 505, 150 y 56. Los consumos asociados a estos productos nunca se verán reflejados en la generación de consumos.

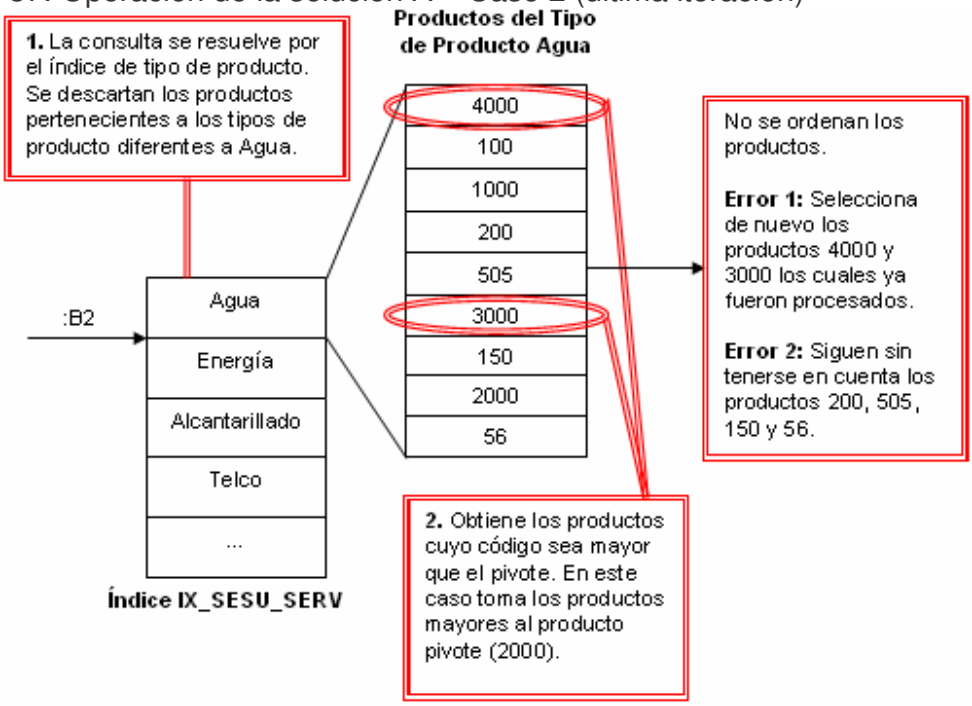
En esta iteración igualmente se obtienen del cursor los primeros tres registros de producto (4000, 3000 y 2000) y se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

Al terminar de procesar dichos consumos, el último registro de producto presente en la estructura en memoria, se convierte en el nuevo producto pivote, es decir, el valor del producto pivote será 2000.

- Tercera iteración (t2): Tercer acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 2000
BULK = 3

Figura 37. Operación de la solución A – Caso 2 (última iteración)



Análisis tercera iteración t(2): Los registros de producto presentes en el cursor son: 4000 y 3000 (los registros mayores al pivote 2000).

Siguen sin procesarse los consumos pertenecientes a los productos 200, 505, 150 y 56, por otro lado, los procesos del producto 3000 se procesan por segunda vez y los del producto 4000 por tercera vez.

En esta iteración igualmente se obtienen del cursor los primeros tres registros de producto (solo 4000 y 3000 en esta iteración) y se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

El pivote termina con valor 3000.

- Cuarta iteración (t3): Cuarto acceso a los productos.

Tipo de producto (:B2) = Agua
Producto Pivote (:B1) = 3000
BULK = 3

Para finalizar, se obtiene por última vez el producto 4000 y se registran sus consumos por cuarta vez consecutiva durante la ejecución del proceso de generación de consumos. Termina el acceso a productos pues ya no hay más productos por procesar según la lógica de la sentencia.

Claramente se puede evidenciar que aunque la cláusula ORDER BY puede incrementar un poco el rendimiento del programa debido al alto costo que le representa a ORACLE ejecutarlo, es necesario contar con los registros ordenados para garantizar que durante a ejecución del programa se procesen los registros que son y las veces que son.

15.2 SOLUCIÓN B

Parámetros de entrada del proceso:

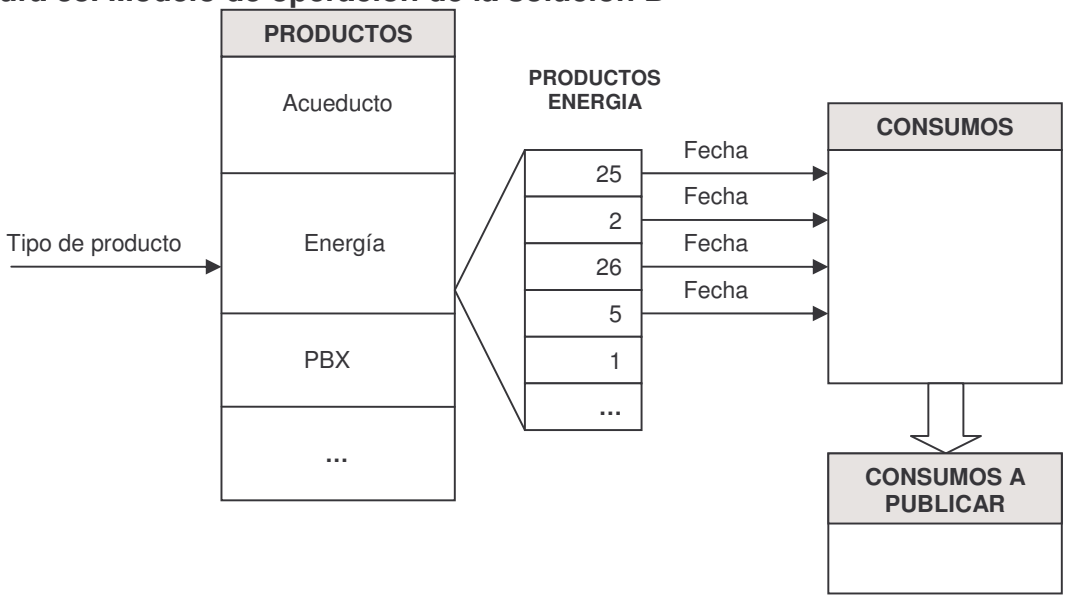
- Tipo de producto
- Métodos de calculo de consumo
- Tipos de consumo
- Fecha de registro de los consumos
- Id de seguimiento del proceso

La solución B recibe como parámetros de entrada los mismos argumentos que la solución A pero adicionalmente recibe un ID del proceso de generación de consumos a través del cual el usuario final puede realizar seguimiento del avance del programa.

La opción B consiste igualmente en acceder los productos por tipo de producto; la gran diferencia entre esta opción y la opción A es que en la solución B no se granula el acceso a los productos, es decir, el procesamiento no se realiza por bloques de productos sino que todos los productos se almacenan en una estructura en memoria que el programa recorre secuencialmente y extrae los consumos de la fecha para cada producto. Finalmente, el programa inserta en la entidad de consumos los registros procesados y realiza la operación COMMIT por

cada 100 productos procesados para asentar la información persistentemente en la base de datos.

Figura 38. Modelo de operación de la solución B



15.2.1 Plan de ejecución para el acceso a productos

```
SELECT --+ index(servsusc IX_SESU_SERV)
*
FROM SERVSUSC
WHERE SERVSUSC.SESUSERV = :B1
```

Esta sentencia obtiene los productos (tabla SERVSUSC) pertenecientes al tipo de producto determinado por el parámetro de entrada del proceso (representado por la variable :B1).
Al igual que en la solución A, la cláusula SELECT va acompañada de la instrucción HINT con el objetivo de indicarle al motor de ORACLE el mejor camino para ejecutar la sentencia.

El plan de ejecución de la sentencia mostrado por la herramienta TKPROF es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
  30070  TABLE ACCESS BY INDEX ROWID SERVSUSC
  30070   INDEX RANGE SCAN IX_SESU_SERV (object id 567541)

Rows     Execution Plan
-------  ---------------------------------------------------
```

```
    0  SELECT STATEMENT   GOAL: CHOOSE
30070   TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF 'SERVSUSC'
30070    INDEX   GOAL: ANALYZED (RANGE SCAN) OF 'IX_SESU_SERV'
                  (NON-UNIQUE)
```

Interpretación del plan de ejecución:

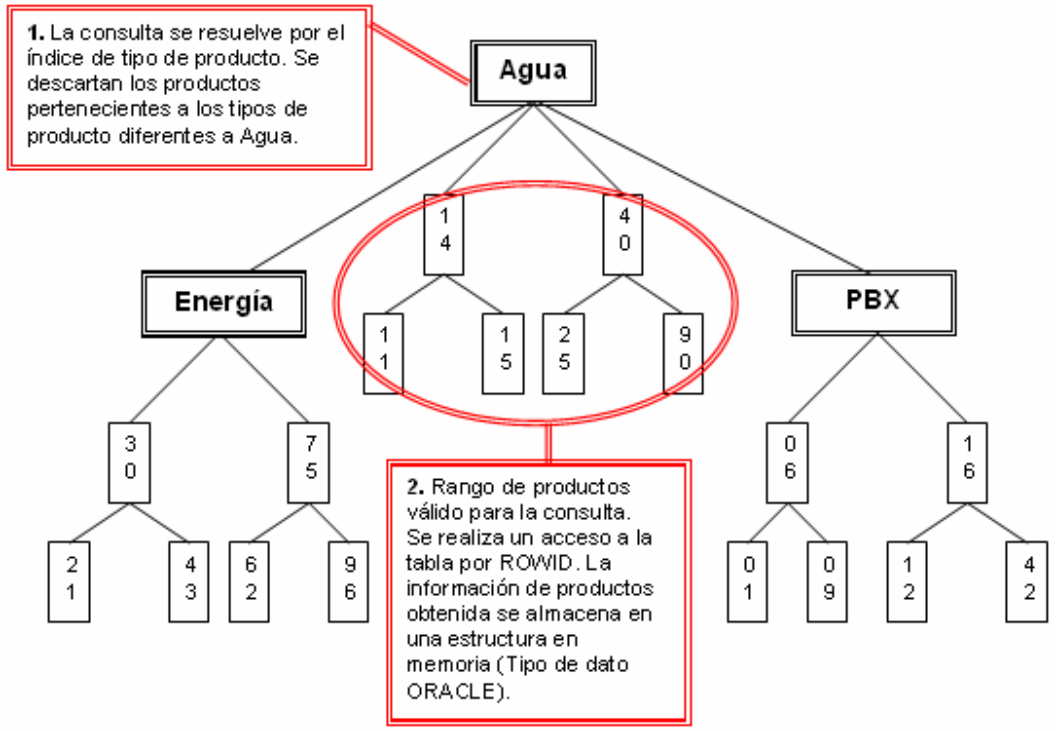
- INDEX RANGE SCAN IX_SESU_SERV: El motor de ORACLE respeta el HINT usado, así que resuelve la sentencia a través del índice IX_SESU_SERV, de esta manera se hace un acceso por rangos a los registros de productos (RANGE SCAN).

Al resolver una sentencia por rangos, se tiene la seguridad de no realizar un barrido de todos los registros de la tabla (FULL SCAN) puesto que de entrada se descartan los registros que no corresponden al rango, en el caso de la sentencia en evaluación el rango lo determina el tipo de Producto.

Suponiendo que el tipo de producto ingresado como parámetro es Agua, el motor de ORACLE accederá únicamente el rango de registros correspondiente al tipo de producto Agua.

- TABLE ACCESS BY INDEX ROWID SERVSUSC: Después de encontrar los productos correspondientes al rango, ORACLE obtiene el ROWID de cada registro (El ROWID está almacenado en el índice) y con este accede físicamente la tabla SERVSUSC para traer los campos solicitados en la sentencia.

Figura 39. Acceso a productos en la solución B



Cabe resaltar que las sentencias de la solución B no se realizan por iteraciones puesto que en esta solución no se granula el acceso a productos, por lo tanto, el plan de ejecución y la sentencia de selección de datos se realiza una sola vez durante todo el proceso.

Finalmente, los registros obtenidos se almacenan en una estructura en memoria para su posterior procesamiento.

15.3 SOLUCIÓN C

Parámetros de entrada al proceso:

- Numero del producto
- Método de calculo
- Tipos de consumo
- Fecha de proceso
- Id de seguimiento del proceso

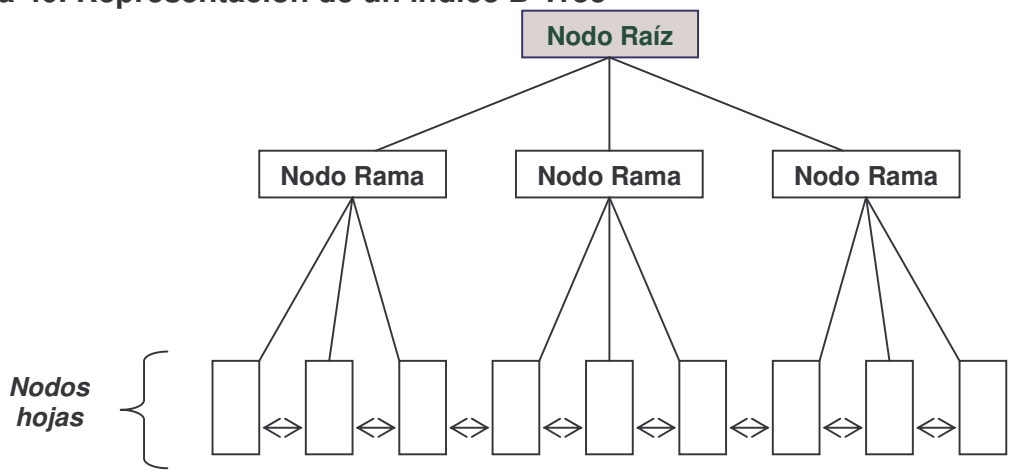
En esta solución se implementa la técnica de la granularidad similar a la de la solución A modificando la sentencia que selecciona los productos de un tipo de producto determinado, eliminando las sentencias de ordenación del SELECT y creando un índice compuesto por los campos producto y numero de producto.

En primera instancia se debe tener en cuenta que la lógica del proceso no sufre ninguna modificación, sin embargo, la eliminación de la cláusula ORDER BY y la creación del índice compuesto disminuye el tiempo de ejecución un 70% con un conjunto de datos de 30.000 registros.

De manera más detallada, un índice estándar en Oracle es un *B-Tree Index*, el cual se construye mediante una estructura de datos tipo árbol balanceado. Todo índice esta compuesto por un nodo raíz, del cual se desprenden las ramas y finalmente los nodos hoja, en los cuales se almacena la información del registro a procesar.

De manera grafica un B-Tree Index tiene la siguiente representación:

Figura 40. Representación de un índice B-Tree

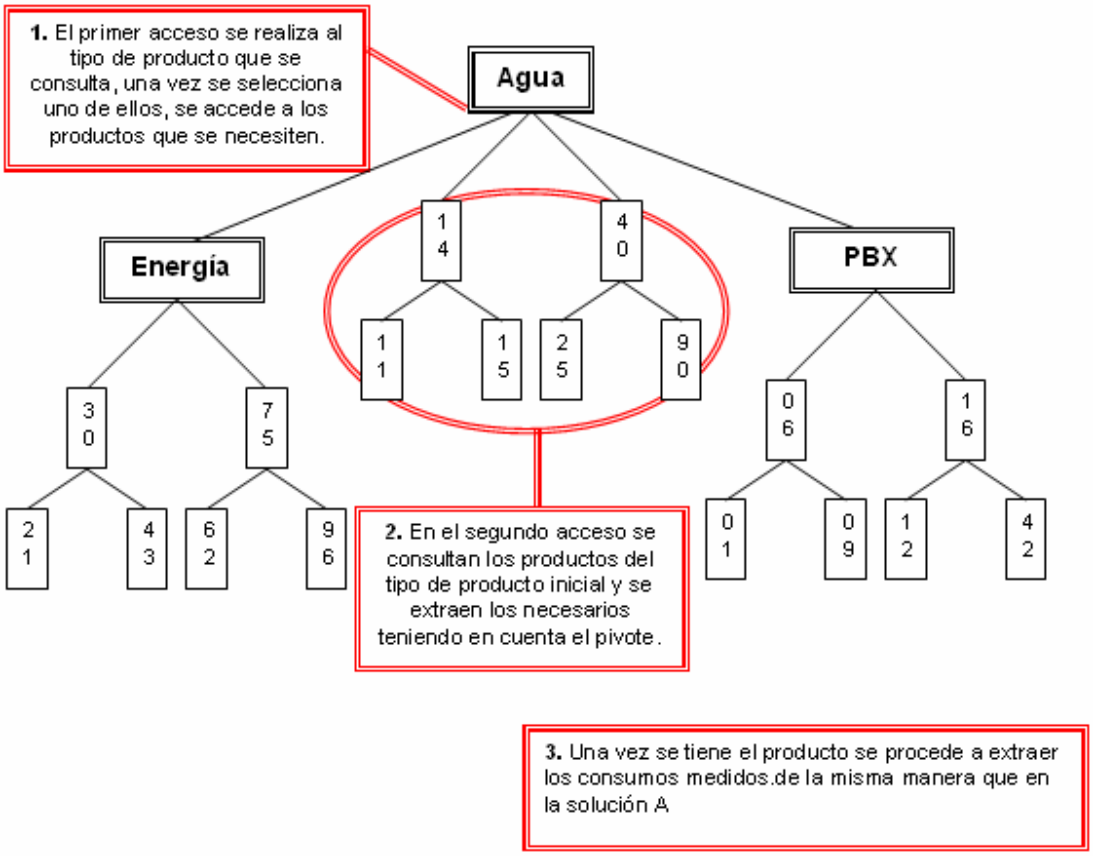


En esta estructura se accede mediante los ROWID de los segmentos de registros, en donde se almacena la dirección en disco de un registro determinado. A través de esta dirección se navega entre las hojas y no es necesario consultar nuevamente por el siguiente registro.

Por otro lado, teniendo en cuenta que es un árbol balanceado se almacenan los registros de manera ordenada y al hacer una consulta los datos se obtienen en orden ascendente, eliminando la necesidad de ordenarlos mediante cláusula ORDER BY.

En el proceso de extracción de productos la distribución grafica del acceso mediante índice se puede graficar de la siguiente manera:

Figura 41. Modelo de operación de la solución C



15.3.1 Plan de ejecución para el acceso a productos

```
SELECT --+ index(servsusc IX_SESU_SERV_NUSE)
*
FROM SERVSUSC
WHERE SESUNUSE > :B2
AND SESUSERV = :B1
```

Esta sentencia obtiene la información sobre productos (tabla SERVSUSC) pertenecientes al tipo de producto determinado por el parámetro de entrada del proceso (representado por la variable :B1).

El acceso a los productos en esta solución es granulado al igual que en la solución A, por lo cual en la condición se utiliza un pivote (variable :B2) que determina el código del último producto procesado, esto con el objetivo de restringir el grupo de datos a trabajar y llevar un control de los registros que ya se han analizado.

El tamaño del BULK usado durante el desarrollo de la solución C fue el mismo usado para la solución (100 registros).

El HINT usado en la sentencia le indica a ORACLE que debe resolver la sentencia utilizando el índice IX_SESU_SERV_NUSE sobre la tabla SERVSUSC; este índice está compuesto por dos campos de dicha tabla, Tipo de Producto: SESUSERV y Código de producto: SESUNUSE.

A diferencia de la sentencia de acceso a productos de la solución A, la solución C no presenta en su estructura la cláusula ORDER BY ya que no la necesita pues el ordenamiento de los productos lo cubre el índice utilizado.

El plan de ejecución de la sentencia C mostrado por la herramienta TKPROF es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
  30070  TABLE ACCESS BY INDEX ROWID SERVSUSC
  30070   INDEX RANGE SCAN IX_SESU_SERV_NUSE (object id 609355)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
  30070   TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF 'SERVSUSC'
  30070    INDEX (RANGE SCAN) OF 'IX_SESU_SERV_NUSE' (NON-UNIQUE)
```

Interpretación del plan de ejecución:

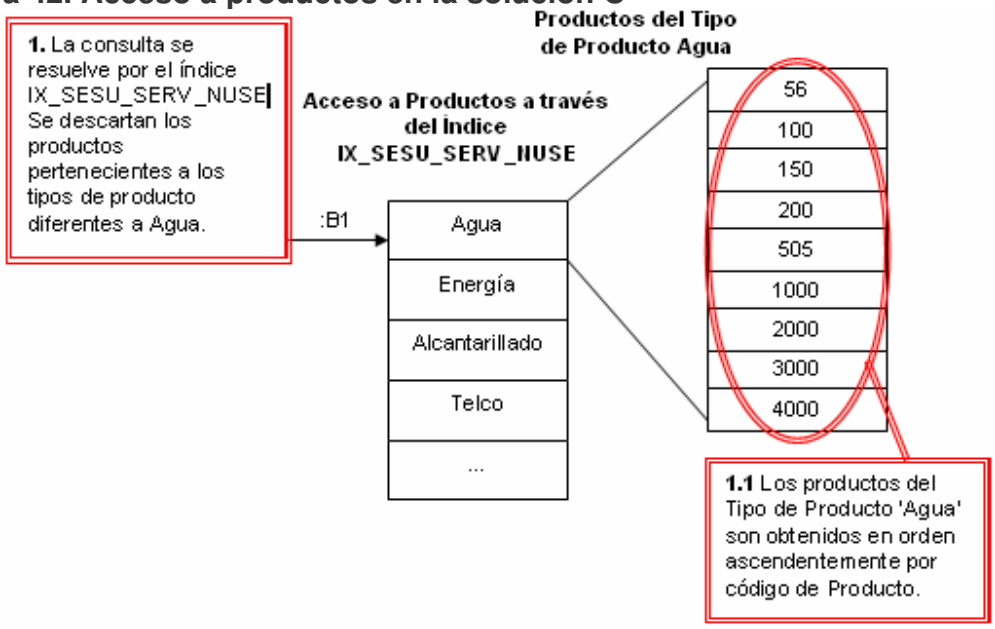
- INDEX RANGE SCAN IX_SESU_SERV_NUSE: El motor de ORACLE respeta el HINT usado, así que resuelve la sentencia a través del índice IX_SESU_SERV_NUSE, de esta manera se hace un acceso por rangos a los registros de productos (RANGE SCAN). Al resolver una sentencia por rangos, se tiene la seguridad de no realizar un barrido de todos los registros de la tabla (FULL SCAN) puesto que de entrada se descartan los registros que no corresponden al rango, en el caso de la sentencia en evaluación el rango lo determina el tipo de Producto.

El índice IX_SESU_SERV_NUSE usado en la solución C está compuesto en primera instancia por el Tipo de Producto, de modo que el acceso a los Productos se hace de la misma manera en la que se acceden los Productos en la solución A; la diferencia radica en que el índice IX_SESU_SERV_NUSE está compuesto también por el código de Producto por lo que se hace innecesario realizar un ordenamiento de los registros de Producto ya que el índice cubre esta necesidad.

Suponiendo que el tipo de producto ingresado como parámetro es Agua, el motor de ORACLE accederá únicamente el rango de registros correspondiente al tipo de producto Agua.

Nota: El índice asegura que los registros obtenidos se encuentren ordenados por código de producto, en este caso el índice se encuentra ordenado Ascendentemente.

Figura 42. Acceso a productos en la solución C



- TABLE ACCESS BY INDEX ROWID SERVSUSC: Después de encontrar los productos correspondientes al rango, ORACLE obtiene el ROWID de cada registro (El ROWID está almacenado en el índice) y con este accede físicamente la tabla SERVSUSC para traer los campos solicitados en la sentencia para los registros que cumplen la condición:

WHERE SESUNUSE > :B2

Es decir, ORACLE solo accederá a disco por la información de los registros de producto cuyo código sea mayor al pivote (:B2). Cada vez que se ejecute esta sentencia el grupo de registros a acceder será menor.

A continuación se muestra el mismo ejemplo de la solución A aplicado en la sentencia de la solución C:

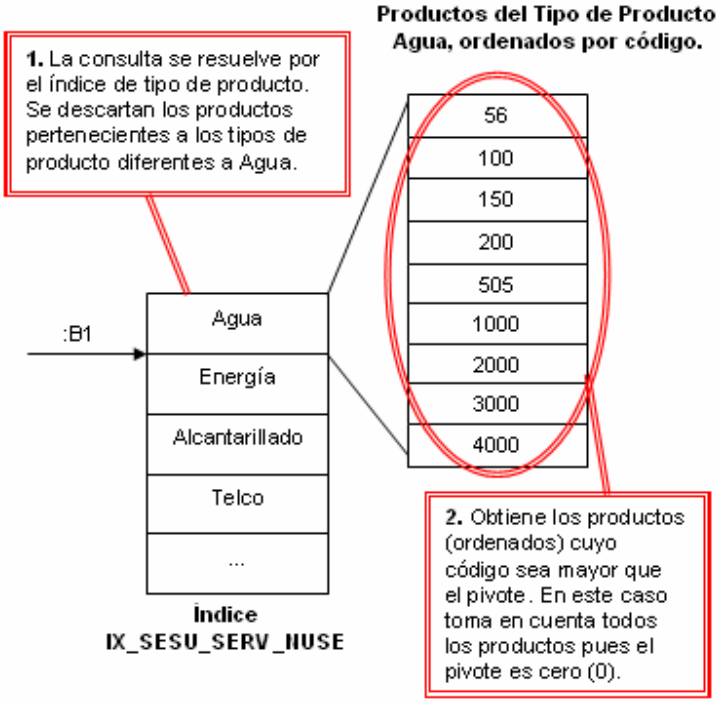
Ejemplo 1: Acceso a productos

- Situación inicial (t0): Primer acceso a los productos.

Tipo de producto (:B1) = Agua

Producto Pivote (:B2) = 0
BULK = 3

Figura 43. Operación de la solución C (primera iteración)



Análisis primera iteración t(0): Los registros de producto desplegados en la tabla del paso 2 se almacenan en un cursor para posteriormente obtener solo los N primeros registros de dicha estructura (la cantidad de registros a obtener la determina la constante BULK, 3 registros para efectos del ejemplo).

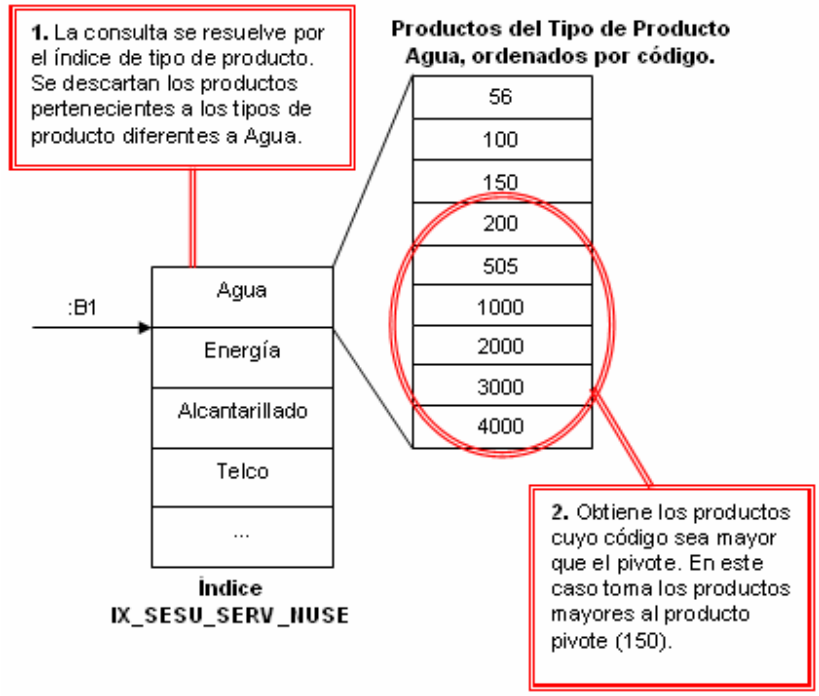
Los registros de producto obtenidos (56, 100 y 150) son almacenados en una estructura en memoria provista por ORACLE y para cada uno de ellos se obtienen los consumos asociados.

Al terminar de procesar los consumos de los tres registros, el último registro de producto presente en la estructura en memoria, se convierte en el producto pivote, es decir, el valor del producto pivote será 150.

- Segunda iteración (t1): Segundo acceso a los productos.

Tipo de producto (:B1) = Agua
Producto Pivote (:B2) = 150
BULK = 3

Figura 44. Operación de la solución C (segunda iteración)



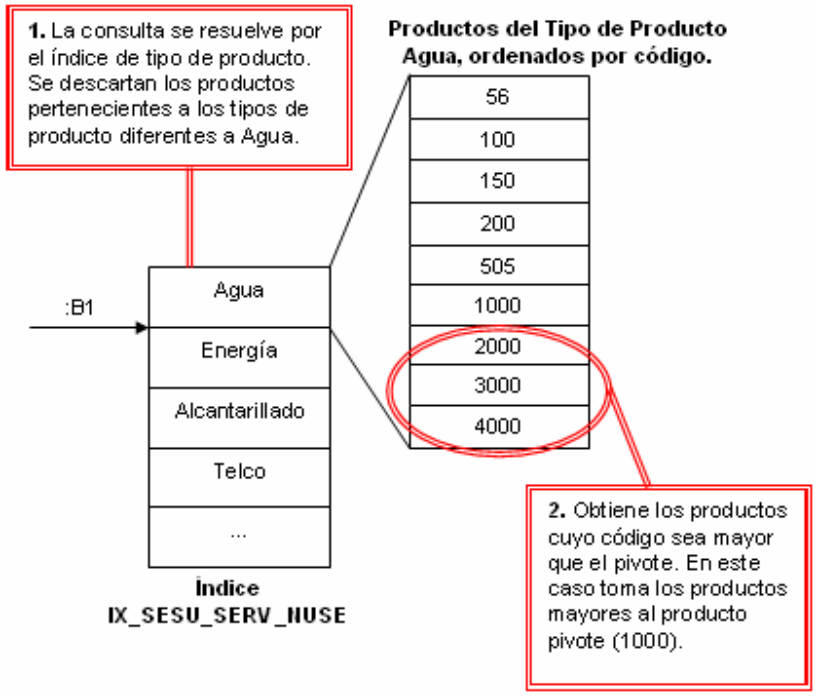
Análisis segunda iteración t(1): En esta iteración igualmente se obtienen del cursor los primeros tres registros de producto (200, 505 y 1000) y se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

Al terminar de procesar dichos consumos, el último registro de producto presente en la estructura en memoria, se convierte en el nuevo producto pivote, es decir, el valor del producto pivote será 1000.

- Tercera iteración (t2): Tercer acceso a los productos.

Tipo de producto (:B1) = Agua
Producto Pivote (:B2) = 1000
BULK = 3

Figura 45. Operación de la solución C (última iteración)



Análisis tercera iteración t(2): El resultado de la sentencia se almacena en el cursor, los primeros tres registros de producto presentes en el cursor (2000, 3000 y 4000) se almacenan en una estructura en memoria para obtener los consumos asociados a cada uno.

Finalizando el procesamiento de los consumos para los productos 2000, 3000 y 4000; se actualiza el valor del pivote a 4000 y termina el acceso a productos pues ya no quedan más por procesar. Fin de lógica del programa.

El ejemplo demuestra que la sentencia de la solución C es funcionalmente igual a la sentencia de la solución A, pues aunque usan diferentes maneras de obtener y de ordenar los datos, las dos soluciones procesan de manera correcta los mismos productos para un conjunto de datos de entrada.

15.4 PLAN DE EJECUCION PARA LAS SENTENCIAS COMUNES A LAS SOLUCIONES A, B Y C

El acceso a productos es la sentencia clave en el proceso de generación de consumos medidos ya que prácticamente determina el rendimiento del programa, sin embargo esta sentencia no es la única operación sobre la base de datos que se realiza en el proceso de generación de consumos. A continuación se presenta otro grupo de operaciones que se realizan durante el proceso, estas sentencias

son instrucciones que de alguna manera son similares para las soluciones A, B y C y están ordenadas descendentemente (primero las más costosas) por el costo que representan para el motor de ORACLE.

Sentencia 1:

```
SELECT SQ_EMC_CONSMEDI_COMDCODI.NEXTVAL, COSSMECC,
COSSTCON, COSSCOCA, PECSFECF, PECSFECI
FROM
CONSSESU, PERICOSE
WHERE COSSPECS = PECSCONS
AND COSSSESU = :B4
AND TRUNC (COSSFERE) = :B3
AND (INSTR (','||:B2, ',' || COSSTCON || ',') > 0)
AND (INSTR (','||:B1 , ',' || COSSMECC || ',') > 0)
```

Esta sentencia retorna para cada producto obtenido en la sentencia anterior (argumento :B4), los datos de consumos acordes con la fecha de registro (:B3), tipos de consumo (:B2) y métodos de cálculo de consumo (:B1) recibidos por el proceso como parámetros de entrada.
También se obtiene a través de un JOIN entre las tablas CONSSESU (Consumos) y PERICOSE (Periodo de consumo) la fecha inicial y la fecha final del periodo de consumo pues son datos requeridos por el proceso.

Nota: Cabe resaltar que la tabla CONSSESU almacena los consumos de todos los productos realizados durante mínimo los últimos seis meses, por lo cual es una tabla que maneja una gran cantidad de registros y tiende a crecer cada vez más.
Por otro lado la tabla PERICOSE almacena los diferentes periodos de consumo; estos son rangos de fechas en los que se agrupan los consumos asociados a los productos. Por lo general se definen periodos de consumo mensuales a través de los años, así pues, la tabla PERICOSE también tiende a crecer pero a un ritmo y volumen mucho menor que la tabla CONSSESU.

El plan de ejecución de la sentencia No.2 es el siguiente:

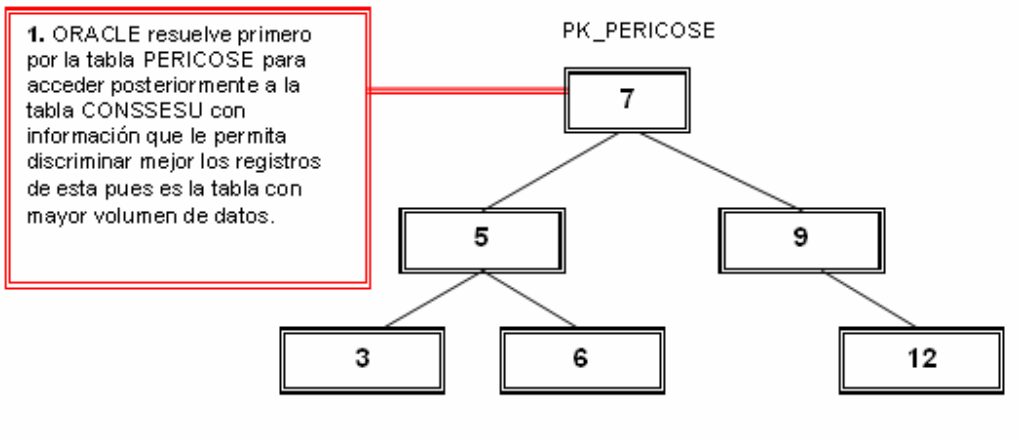
```
Rows     Row Source Operation
-------  ---------------------------------------------------
     13  SEQUENCE
     13   NESTED LOOPS
     13    TABLE ACCESS BY INDEX ROWID CONSSESU
  32879     INDEX RANGE SCAN IX_COSS_SESU_PEFA (object id 561065)
     13    TABLE ACCESS BY INDEX ROWID PERICOSE
     13     INDEX UNIQUE SCAN PK_PERICOSE (object id 566825)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
     13   SEQUENCE OF 'SQ_EMC_CONSMEDI_COMDCODI'
```

```
    13    NESTED LOOPS
    13     TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
               'CONSSESU'
 32879      INDEX   GOAL: ANALYZED (RANGE SCAN) OF 'IX_COSS_SESU_PEFA'
               (NON-UNIQUE)
    13     TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
               'PERICOSE'
    13      INDEX   GOAL: ANALYZED (UNIQUE SCAN) OF 'PK_PERICOSE'
               (UNIQUE)
```

Interpretación del plan de ejecución:

- INDEX UNIQUE SCAN PK_PERICOSE: El motor de ORACLE comienza a resolver el query a través del periodo de consumo (tabla PERICOSE). Para este fin accede uno de los registros de la tabla por medio de la llave primaria (Indice único PK_PERICOSE).

Figura 46. Sentencias comunes – Acceso a periodos de consumo

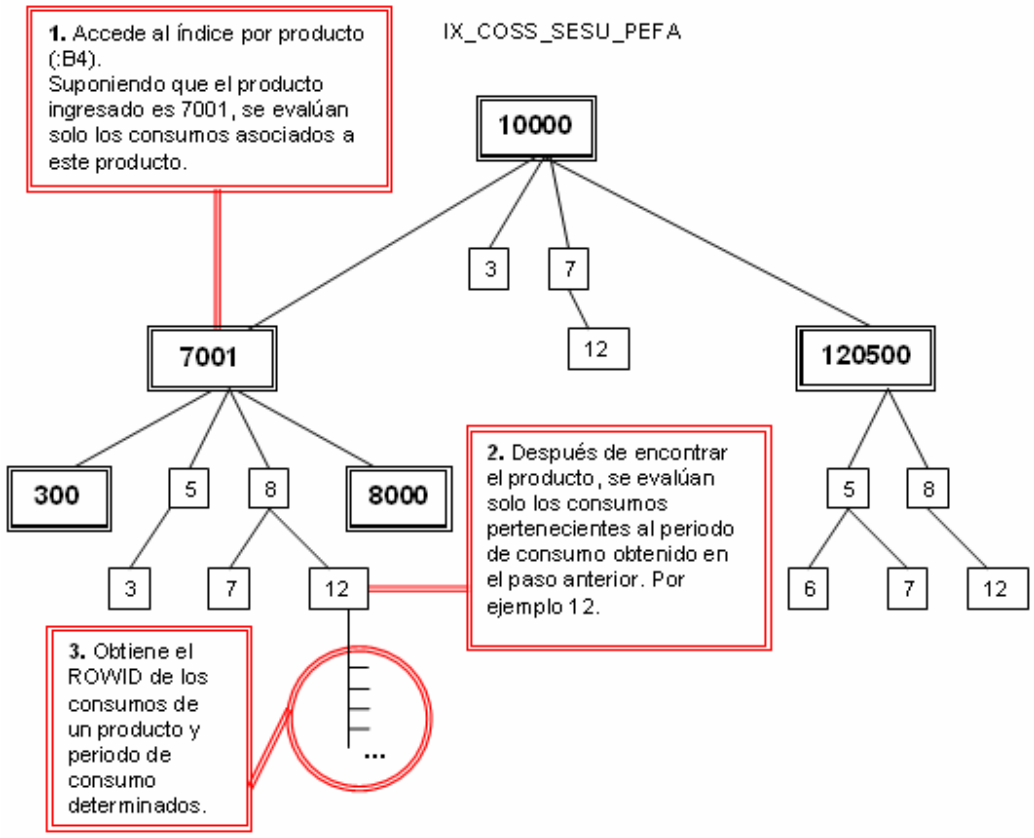


- TABLE ACCESS BY INDEX ROWID PERICOSE: Con el ROWID obtenido en el índice PK_ PERICOSE, ORACLE accede físicamente a la tabla PERICOSE para obtener los campos PECSFECF y PECSFECI requeridos en la sentencia SELECT.

- INDEX RANGE SCAN IX_COSS_SESU_PEFA: Obtiene los ROWID de los registros de consumos pertenecientes al producto ingresado como parámetro (:B4) y al periodo de consumo tomado en el paso anterior.

Este acceso por rangos (RANGE SCAN) lo realiza por medio del indice compuesto IX_COSS_SESU_PEFA el cual está ordenado por producto y en segunda instancia por periodo de consumo.

Figura 47. Sentencias comunes – Acceso a consumos por producto



- TABLE ACCESS BY INDEX ROWID CONSSESU: Con el ROWID obtenido en el índice IX_COSS_SESU_PEFA, ORACLE accede físicamente a la tabla CONSSESU para obtener los demás campos de la tabla requeridos en la sentencia SELECT.

- NESTED LOOPS: La herramienta TKPROF indica que ORACLE realiza un ciclo sobre el indice PK_PERICOSE y realiza los pasos 1, 2, 3 y 4 para cada periodo de consumo registrado en la tabla PERICOSE.

- SEQUENCE: Por ultimo obtiene el próximo número de la secuencia SQ_EMC_CONSMEDI_COMDCODI, este código se usa como llave primaria del registro a insertar en la entidad de consumos medidos. Se ejecuta una sola vez (Instrucción por fuera del NESTED LOOP).

Sentencia 2:

```
DELETE --+ INDEX(emc_consmedi, IX_EMC_CONSMEDI01) FROM
EMC_CONSMEDI
WHERE (COMDFEPR = :B2)
```

```
AND ROWNUM < :B1
```

Esta sentencia elimina de la entidad EMC_CONSMEDI los consumos registrados en una fecha (:B2) que ya se ha procesado con anterioridad (reproceso).
El borrado de los registros se hace de forma granulada, es decir, eliminando una cantidad de registros determinada por el argumento :B1 (constante con valor 100) y realizando commit de los 100 registros borrados para después eliminar los siguientes 100 registros. El acceso a los registros se hace por medio del índice IX_EMC_CONSMEDI01 el cual está compuesto por los campos fecha de proceso (COMDFEPR) y por el consecutivo de la tabla (llave primaria COMDCODI).

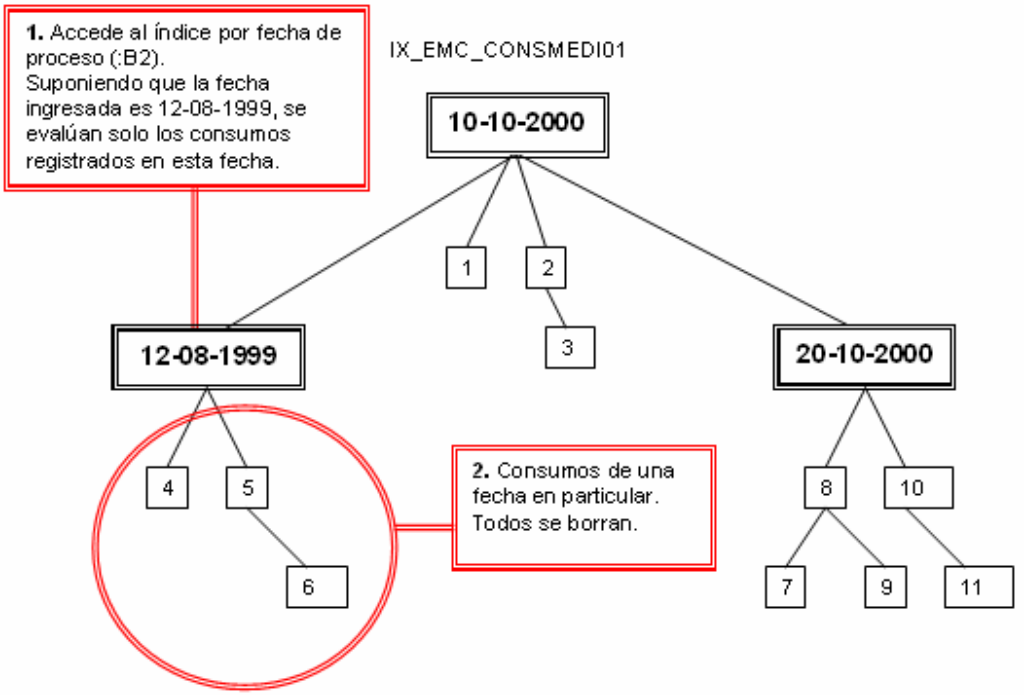
El plan de ejecución de la sentencia No.3 es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
      0  DELETE
     13   COUNT STOPKEY
     13    INDEX RANGE SCAN IX_EMC_CONSMEDI01 (object id 607754)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  DELETE STATEMENT   GOAL: CHOOSE
      0   DELETE OF 'EMC_CONSMEDI'
     13    COUNT (STOPKEY)
     13     INDEX (RANGE SCAN) OF 'IX_EMC_CONSMEDI01' (NON-UNIQUE)
```

Interpretación del plan de ejecución:

- INDEX RANGE SCAN OF IX_EMC_CONSMEDI01: El motor de ORACLE sigue el HINT indicado y resuelve la sentencia a través del índice IX_EMC_CONSMEDI01. El acceso a los registros de consumos es por rangos puesto que el índice descarta de entrada los consumos que no pertenecen a la fecha de entrada del proceso (:B2).

Figura 48. Sentencias comunes – Acceso a consumos medidos por fecha de generación



- COUNT STOPKEY: ORACLE determina la cantidad de registros que va a eliminar, esta cantidad la determina el número de filas (ROWNUM) acotado por el argumento constante :B1.

- DELETE: Borra el bloque de registros objetivo.

Sentencia 3:

```
UPDATE ESTAPROG
SET ESPRPORC = :B6 , ESPRMESG = :B5 ,
ESPRSUPR = NVL (:B4 , ESPRSUPR), ESPRFEFI = SYSDATE, ESPRTAPR
= NVL (:B3 , ESPRTAPR), ESPRSUFA = NVL (:B2, ESPRSUFA)
WHERE ESPRPROG = UPPER (:B1)
```

La tabla ESTAPROG (Estado del programa) se utiliza en el producto OpenSmartFlex para realizar un seguimiento de los procesos que el usuario ejecuta, dicho seguimiento se realiza sobre un registro de la entidad ESTAPROG, sobre este se puede consultar sí el proceso sigue en ejecución o sí ya terminó, igualmente indica sí el programa terminó correctamente o sí terminó con errores.
La sentencia No. 4 actualiza el registro de ESTAPROG que identifica el proceso de generación de consumos indicando la cantidad de productos procesados hasta el momento y la última fecha de actualización.

El plan de ejecución de la sentencia No.4 es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
    302  UPDATE
    302   INDEX UNIQUE SCAN PK_ESTAPROG (object id 561550)
```

El acceso al registro de ESTAPROG se realiza por medio de la llave primaria PK_ESTAPROG puesto que en el WHERE de la sentencia se indica el código primario de la entidad (ESPRPROG).

Sentencia 4:

```
SELECT --+ index(servsusc IX_SESU_SERV)
COUNT(*)
FROM SERVSUSC
WHERE (SESUSERV = :B1)
```

Esta sentencia contabiliza los registros de producto (tabla SERVSUSC) correspondientes al tipo de producto (SESUSERV) ingresado como parámetro de entrada del proceso (argumento :B1). El objetivo de esta sentencia es obtener la cantidad total de productos por tipo de producto para permitir contrastar la cantidad de productos procesados contra la cantidad de productos a procesar, es decir, permite realizar el seguimiento del proceso a través de la entidad ESTAPROG.

El plan de ejecución de la sentencia No.5 es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE
  30070   INDEX RANGE SCAN IX_SESU_SERV (object id 567541)
```

La sentencia se resuelve a través del índice IX_SESU_SERV el cual se compone del campo SERVSUSC:SESUSERV y se limita a contabilizar la cantidad de registros existentes por tipo de producto.

Sentencia 5:

```
INSERT INTO EMC_CONSMEDI
(COMDCODI, COMDFEPR, COMDSESU, COMDTICO,
COMDMECC, COMDUNCO, COMDFEIC, COMDFEFC)
VALUES
( :B8 , :B7 , :B6 , :B5 , :B4 , :B3 , :B2 , :B1 )
```

La sentencia No. 6 realiza la inserción del registro procesado sobre la tabla de consumos medidos.

Los valores ingresados son:

:B8 Código obtenido de la secuencia SQ_EMC_CONSMEDI_COMDCODI.
:B7 Fecha de registro del consumo original
:B6 Código del producto
:B5 Código del tipo de consumo
:B4 Método de cálculo de consumo
:B3 Cantidad de unidades de consumo
:B2 Fecha inicial de consumo
:B1 Fecha final de consumo

Sentencia 6:

```
SELECT --+ INDEX(emc_consmedi, IX_EMC_CONSMEDI01)
COUNT(*)
FROM EMC_CONSMEDI
WHERE (COMDFEPR = :B1)
```

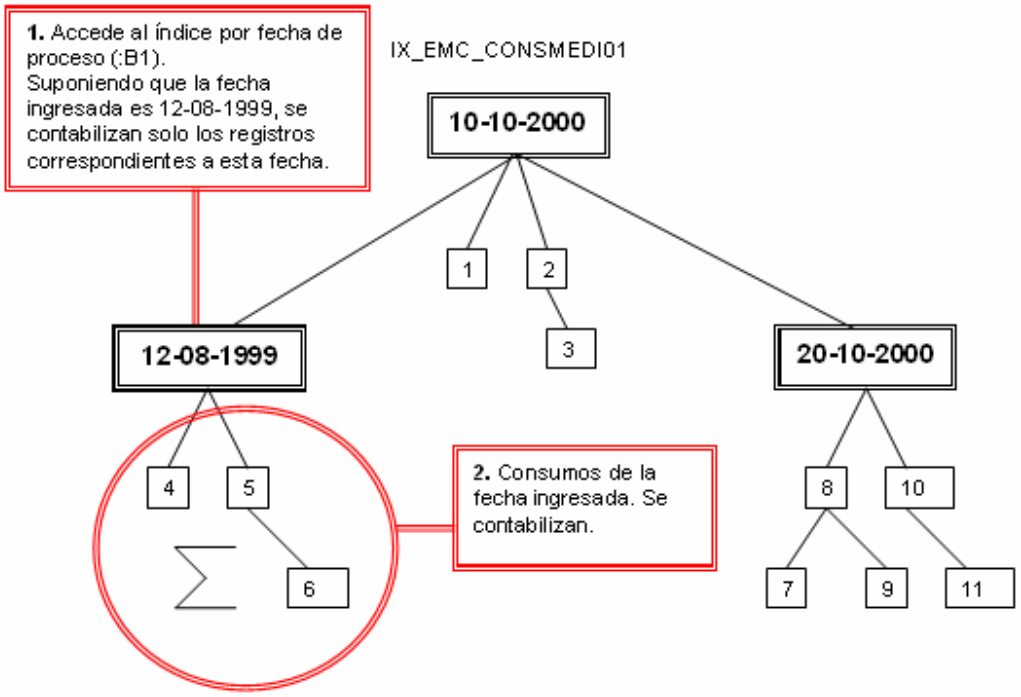
Esta sentencia selecciona la cantidad de registros existentes en la entidad de consumos medidos para la fecha ingresada como parámetro inicial del proceso. La sentencia se ejecuta al iniciar el proceso con el fin de determinar si ya se han generado consumos para la fecha dada (reproceso).

El plan de ejecución de la sentencia No.7 es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE
     13  INDEX RANGE SCAN IX_EMC_CONSMEDI01 (object id 607754)
```

La sentencia se resuelve a través del índice IX_EMC_CONSMEDI01 el cual está compuesto por el campo COMDFEPR (fecha de proceso) y su función es contabilizar la cantidad de registros existentes para la fecha ingresada.

Figura 49. Sentencias comunes – Acceso a consumos medidos por fecha de proceso



Sentencia 7:

```
INSERT INTO ESTAPROG
(ESPRPROG, ESPRPORC, ESPRMESG, ESPRFEIN,
ESPRFEFI, ESPRTAPR, ESPRSUPR, ESPRSUFA)
VALUES
( :B8 , :B7 , :B6 , :B5 , :B4 , :B3 , :B2 , :B1 )
```

La sentencia No. 8 inserta el registro en la tabla de seguimiento del proceso, ESTAPROG.

Los valores ingresados son:
:B8 Identificador del programa (HGCM<consecutivo>)
:B7 Registros procesados representado porcentualmente (inicia en 0%)
:B6 Descripción del estado del proceso.
:B5 Fecha de inicio de proceso (Fecha de inserción del registro en Estaprog)
:B4 Fecha final del proceso (Fecha de ultima actualización en Estaprog)
:B3 Total de registros a procesar (expresado en cantidad de productos del tipo de producto ingresado como parámetro del proceso).
:B2 Registros procesados (Cantidad de productos procesados hasta la última actualización del registro).
:B1 campo utilizado en otro proceso ajeno a la generación de consumos.

Sentencia 8:

```
SELECT *
FROM
SERVICIO
WHERE SERVCODI = :B1
```

Esta sentencia selecciona toda la información de un producto específico.

El plan de ejecución de la sentencia No.9 es el siguiente:

```
Rows     Row Source Operation
-------  ---------------------------------------------------
      2  TABLE ACCESS BY INDEX ROWID SERVICIO
      2  INDEX UNIQUE SCAN PK_SERVICIO (object id 567525)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
      2  TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF 'SERVICIO'
      2  INDEX   GOAL: ANALYZED (UNIQUE SCAN) OF 'PK_SERVICIO' (UNIQUE)
```

Debido a que la sentencia recibe el código del producto (llave primaria de la tabla) como parte de la cláusula WHERE, ORACLE accede directamente al ROWID del registro solicitado por medio del índice PK_SERVICIO y con el ROWID accede físicamente a la tabla para recolectar los otros campos pedidos.

Sentencia 9:

```
SELECT SQESPRPROG.NEXTVAL
FROM
DUAL
```

Selecciona el próximo valor de la secuencia SQESPRPROG. Con este valor y el nombre del programa (HGCM) el programa arma la llave primaria del registro a insertar en la tabla ESTAPROG al iniciar el proceso.

16. COMPARACIÓN DE TIEMPOS EN LAS SOLUCIONES

Al realizar el proceso de análisis de traza mediante el paquete Oracle SQL Trace se obtienen diferentes tiempos de ejecución y de operaciones internas (como parseo de sentencias, consultas a disco, consultas a memoria y obtención de datos) los cuales sirven para establecer medidas de rendimiento en las aplicaciones.

Estos tiempos están dados por las operaciones[7]:

- Parse (Parseo): En esta fase el motor Oracle interpreta la sentencia ingresada, analizándola sintácticamente, estableciendo las variables que requiere usar y preparándola para la ejecución.
- Execute (Ejecución): Una vez se parsea la sentencia Oracle la ejecuta. Aunque el proceso puede parecer muy simple, sí la sentencia es compleja puede consumir cantidades considerables de tiempo en ejecución.
- Fetch (Obtención de datos): Esta fase esta dada por los accesos a disco o memoria que el motor deba realizar para obtener los datos que requiera para el proceso. Esta operación se ejecuta únicamente en sentencias SELECT.

Cada una de las anteriores operaciones tiene los siguientes análisis realizados por SQL Trace:

- Count: Se refiere al conteo del número de veces que la operación se realiza durante todo el proceso:
- CPU: Mide el tiempo que la operación consume en procesador.
- Elapsed: Tiempo que trascurre mientras que la operación se finaliza.
- Disk: Se refiere a la cantidad de registros que la operación consulta en disco.
- Query: Se refiere al número de registros obtenidos de manera secuencial en las operaciones anteriores.
- Current: Se refiere al número de registros obtenidos actualmente en las operaciones.
- Rows: Se refiere al número de filas procesadas por la sentencia para cumplir el objetivo requerido.

De acuerdo a lo anterior, se procede a comparar los tiempos consumidos en segundos por las sentencias explicadas en el punto 15.

Solución A:

```
SELECT --+ INDEX(servsusc IX_SESU_SERV)
```

[7] ORACLE. En: Oracle9i, Database Performance Tuning Guide and Reference, Release 2 (9.2). (Mar. 2002); p xxxiv.

```
SESUNUSE, SESUFEIN, SESUFERE FROM SERVSUSC WHERE SESUSERV =
:B2 AND SESUNUSE > :B1 ORDER BY SESUNUSE
```

Tabla 31. Solución A

CALL	COUNT	CPU	ELAPSED	DISK	QUERY	CURRENT	ROWS
PARSE	1	0.02	0.01	0	0	0	0
EXECUTE	589	0.03	1.01	0	0	0	0
FETCH	301	52.06	136.99	1653325	1657306	1835	30072
TOTAL	891	52.11	138.02	1653325	1657306	1835	30072

Solución B:

```
SELECT --+ index(servsusc IX_SESU_SERV)
 * FROM SERVSUSC WHERE SESUSERV = :B1
```

Tabla 32. Solución B

CALL	COUNT	CPU	ELAPSED	DISK	QUERY	CURRENT	ROWS
PARSE	1	0.00	0.00	0	0	0	0
EXECUTE	1	0.01	0.00	0	0	0	0
FETCH	1	5.03	6.18	5198	5506	0	30072
TOTAL	3	5.04	6.19	5198	5506	0	30072

Solución C:

```
SELECT --+ index(servsusc IX_SESU_SERV_NUSE)
 * FROM SERVSUSC WHERE SESUNUSE > :B2 AND SESUSERV = :B1
```

Tabla 33. Solución C

CALL	COUNT	CPU	ELAPSED	DISK	QUERY	CURRENT	ROWS
PARSE	1	0.00	0.00	0	0	0	0
EXECUTE	301	0.03	0.02	0	0	0	0
FETCH	301	2.16	2.13	5257	14357	0	30072
TOTAL	603	2.19	2.15	5257	14357	0	30072

Los tiempos de las tres soluciones se han organizado del mayor a menor consumo de tiempo en CPU, el cual es que busca optimizarse.

En primera instancia, la solución A denota el mayor tiempo registrado de consumo en las operaciones, con un total de 52.11 segundos, la solución B consume un total de 5.04 segundos mientras que la solución C tiene un consumo de 2.19 segundos. De manera general, se puede notar que en todas las soluciones obtuvieron la misma cantidad registros a procesar (un total de 30072, en cada tabla se muestra en la última fila de la columna Rows), lo que garantiza que se efectuaron los cálculos sobre el mismo conjunto de datos en cada uno de los procesos.

Por otra parte, el tiempo consumido por la operación Fetch en las soluciones que utilizan técnica de granulación (soluciones A y C) es considerablemente diferente, aunque se ejecute la misma cantidad de veces.

La diferencia entre las soluciones A y C radica en la necesidad de la solución A de ordenar los datos de producto obtenidos para garantizar que no se omitan registros durante el proceso, ya que por cada conjunto de datos que se obtiene se ordena y luego se procesa. Esta ordenación de cada iteración perjudica notablemente el tiempo que consume el proceso en CPU y en tiempo de espera (Elapsed).

La solución B no utiliza granulación, esta solución obtiene en una sola consulta todos los registros a procesar y los almacena en memoria para su posterior procesamiento, esta característica se nota claramente en la operación Count del llamado Fetch ya que solamente se realiza 1 ejecución para traer la totalidad de los datos, mientras que en las opciones A y C la operación Fetch se realiza una vez por cada iteración del programa (granulación).

En esta opción no se requiere contar con los datos ordenados y no se presentan situaciones de bloqueo, ya que cada cierta cantidad de datos se liberan los mismos al realizar commit sobre la transacción.

Otro aspecto a destacar de la solución B es que presenta tiempos de respuesta buenos comparados con los tiempos de la solución A, sin embargo sus tiempos no superan en rendimiento los tiempos de la solución C, pues un programa granulado correctamente siempre presentará un comportamiento igual o mejor que el de un programa que obtiene los datos de una sola vez y los almacena en memoria.

17. PARTICIPANTES

Estudiantes:
David Arias Villalobos.
Julio Cesar Ledesma Román.
Andrés Fabián Sánchez Cifuentes.

Universidad Autónoma de Occidente.

Director Académico:
Ricardo Llano Valencia.
Ingeniero de Sistemas, Magister en Dirección Universitaria, Especialista en Auditoría de Sistemas, Especialista en Investigación en Contextos de Docencia Universitaria.

Director Región Andina - JPVAL

Coordinador en la empresa:
José Ricardo Calixto Porras.
Ingeniero de Sistemas, Diplomado Desarrollo Software Orientado a Objetos, Oracle Master - Oracle Corporation University.

Arquitecto de Producto - Open Internacional Systems Ltda.

18. CRONOGRAMA

Tabla 34. Cronograma

FASE/SEMANA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Redacción y aprobación de anteproyecto																		
Conocimiento de la lógica del negocio																		
Recopilación y análisis de información																		
Coaching / Autoformación																		
Diseño de la solución																		
Implementación																		
Pruebas funcionales de la aplicación																		
Análisis y comparación de las soluciones																		
Elaboración de documento final																		

19. CONCLUSIONES DE DESARROLLO

Al analizar los procesos, planes de ejecución y tiempos consumidos de las tres soluciones planteadas, se puede concluir que la combinación de prácticas de afinamiento de sentencias SQL, la utilización de recursos de la Base de Datos como los índices y las técnicas de desarrollo usadas establecen la eficiencia en la implementación de un producto de software.

En primera instancia se puede mencionar el caso de la solución A, en el cual se utilizó la técnica de granularidad haciendo uso de un índice para acceder por rangos los registros de productos, es decir, segmentando el conjunto de datos a procesar por tipo de producto. Sin embargo, dicho índice no contaba con el campo pivote dentro de su estructura, incumpliendo el principio de la granularidad, en el cual se requiere ordenar por el pivote los datos obtenidos. Debido a que los datos no se obtenían ordenados a través del índice se recurrió al uso de la cláusula ORDER BY en la sentencia de selección de productos, lo cual desmejoró notablemente el tiempo de ejecución del proceso con respecto a las otras soluciones pues la sentencia ordena en cada iteración de la granulación el conjunto de datos lo cual resulta muy costoso para el motor de ORACLE puesto que aumentan las operaciones a realizar por parte de éste, dando como resultado un incremento en el uso del procesador y en los tiempos de espera del proceso.

De la solución A se concluye que:

- A pesar de que se implementó la técnica de granular el acceso a los datos con el fin de mejorar los tiempos de ejecución del proceso y de optimizar los recursos de la base de datos y del servidor, granular en esta ocasión fue contraproducente pues los tiempos obtenidos fueron peores que los de otras soluciones que incluso no granulaban.

- La razón por la cual no fue un éxito el uso de la granularidad en la solución A fue el hecho de sobrecargar el trabajo del motor de la base de datos con la tarea de ordenar el conjunto de registros obtenido en cada iteración del programa.

- Aunque los tiempos obtenidos fueron pésimos comparados con los tiempos de las otras dos soluciones, en la solución A se cumplieron otras ventajas consecuencia del uso de la granularidad como:
  - Liberación de recursos de la base de datos: Al ejecutar la instrucción commit (persistencia de los cambios realizados en la base de datos) después de cada iteración se liberan los registros de productos obtenidos, es decir, al realizar commit los registros involucrados quedan desbloqueados y pueden ser utilizados por otro proceso concurrente al proceso de generación de consumos.

- Liberación de memoria SGA: La solución A en cada iteración obtiene n cantidad de registros de productos, los sube en memoria SGA usando una colección, procesa los registros de la colección y finalmente inicializa la colección borrándola de la memoria para utilizarla nuevamente en la siguiente iteración. Es así como la solución A es eficiente en cuanto a utilización de memoria SGA pues siempre reserva la misma cantidad pequeña en lugar de acumular reserva de memoria por cada iteración.

Por otra parte, en la solución B no se implementó la técnica de granularidad buscando agilizar la construcción del desarrollo; en su lugar se utilizó una estructura en memoria o colección para almacenar temporalmente todos los registros de productos del tipo de producto ingresado como parámetro.

Esta técnica se conoce como caché de segundo nivel y consiste en subir en memoria todos los registros a procesar, con el objetivo de reducir el número de accesos a la base de datos. Esto mejora considerablemente los tiempos de respuesta del programa puesto que es mucho más rápido acceder los datos en memoria que accederlos en disco duro.

Se recomienda usar caché de segundo nivel cuando la cantidad de registros a subir en memoria no es muy grande (entre 1000 y 10000), pues una cantidad de registros demasiado elevada implica que se reserve mucha memoria para almacenar la colección con los registros.

Aunque la solución B utilizó el mismo índice que la solución A para acceder los productos, en ésta no fue necesario ordenar los productos por código a través del índice ni a través del uso de la instrucción ORDER BY puesto que al almacenar todos los productos objetivo en una colección, se garantizaba que todos los registros serían procesados sin importar el orden en el que estuvieran en la colección.

De la solución B se concluye que:

- La principal ventaja de la solución B radica en que no exige obtener los registros de productos ordenados por código, por lo que el motor realiza menos operaciones y consultas consumiendo menos tiempo y menos procesador del servidor.

- Igualmente se destaca de esta solución la característica de hacer un solo acceso a disco por productos durante toda la ejecución del programa (caché de segundo nivel) con el fin de trabajar con los datos en memoria.

- Aunque la solución B no trabaja por iteraciones como la solución A, esta también ejecuta la instrucción commit cada n registros procesados; por lo tanto

también cuenta con la ventaja de liberar recursos de la base de datos pues no bloquea los registros de la tabla de productos.

- Otra ventaja de esta solución es que al obtener a la misma vez todos los productos (sin iteraciones), disminuye la complejidad del acceso a los datos mejorando el tiempo promedio de construcción, simplificando la lógica del desarrollo para el programador y facilitando las posteriores pruebas de escritorio del programa.

- La desventaja de la solución B consiste en no liberar memoria SGA. Ésta solución reserva una cantidad de memoria proporcional a la cantidad de registros de producto del tipo de producto ingresado como parámetro, almacena en una colección dichos registros, los procesa secuencialmente y sólo hasta el final del programa elimina la colección y libera la memoria consumida. A diferencia de la solución A cuya memoria reservada durante todo el programa consistía en el grupo de n registros obtenido en cada iteración.

Finalmente, en la construcción de la solución C se utilizó también la técnica de granularidad y adicionalmente se construyó un índice compuesto para acceder por rangos los registros de productos, dicho índice está ordenado por el tipo de producto y en segunda instancia por el código de producto (pivote). Puesto que el índice garantiza obtener en orden los productos no fue necesario incluir la cláusula ORDER BY dentro de la sentencia de acceso a productos como se hizo en la solución A.

De la solución C se concluye que:

- Cuenta con las ventajas procedentes de la implementación de la técnica de granularidad:
  - Liberación de recursos de la base de datos: Al ejecutar la instrucción commit (persistencia de los cambios realizados en la base de datos) después de cada iteración se liberan los registros de productos obtenidos, es decir, al realizar commit los registros involucrados quedan desbloqueados y pueden ser utilizados por otro proceso concurrente al proceso de generación de consumos.
  - Liberación de memoria SGA: La solución C es eficiente en cuanto a utilización de memoria SGA pues únicamente reserva la cantidad de memoria necesaria para almacenar los n registros de producto por iteración, en lugar de almacenar en memoria todos los productos a procesar durante todo el tiempo de ejecución del proceso tal como lo hace la solución B.

- Es la solución más completa porque además de las ventajas derivadas de la granulación obtenidas en la solución A, también cuenta con los mejores tiempos de ejecución de los tres obtenidos.

- El éxito de la implementación de la solución C radica en que al igual que en la solución A se cumplieron todos los principios de granularidad; sin embargo, la exigencia de trabajar con los registros de productos ordenados por pivote fue suplida por el índice y no por el motor de la base de datos, lo cual se ve reflejado en los tiempos de ejecución totales del proceso y en la cantidad de operaciones que ORACLE tuvo que realizar para obtener los datos.

- Los tiempos de ejecución del proceso son buenos en la solución C puesto que la información de productos se encuentra ordenada de tal manera en el índice compuesto que ya está lista para ser procesada. El tiempo de ordenamiento que en la solución A era cargado al proceso de generación de consumos, en la solución C es cargado al mantenimiento del índice en el momento en que se ejecuten operaciones de inserción, actualización y/o borrado sobre la tabla de productos.

- Fue la solución más compleja de construir pues involucra conceptos de granularidad y fue necesario realizar un análisis extra para evaluar y construir el índice que cubriera las necesidades del programa.

Finalmente el plan de ejecución de las soluciones A, B y C se aprecia que las sentencias de los tres programas son similares; sin embargo, la sentencia de acceso a los productos marca la diferencia entre cada una de ellas pues es el punto critico del programa donde se exigen al máximo los recursos de la maquina y de la base de datos, es por esto que la solución escogida como la mejor solución es la solución que presentó en este punto un mejor comportamiento en comparación a las otras dos soluciones.

La mejor solución es la solución C pues de acuerdo a los análisis de las trazas de rendimiento presenta los mejores tiempos de respuesta en el acceso a los registros de producto, lo cual hace de esta solución una pieza de software construida acorde con las estrategias trazadas por Open Systems donde el rendimiento de los procesos es un factor primario para determinar el éxito del producto de software.

20. CONCLUSIONES PERSONALES

El objetivo de estas conclusiones es contextualizar al lector acerca de las experiencias, percepciones y conocimientos adquiridos por los estudiantes de la Universidad Autónoma de Occidente durante la práctica laboral realizada en la empresa Open Systems.

Con respecto a la formación ofrecida por la Universidad Autónoma de Occidente en el programa académico de Ingeniería Informática, se puede afirmar que brindó las herramientas básicas necesarias para asimilar los conceptos de sistemas y aplicarlos al producto de Open Systems, logrando así un buen desempeño laboral. De igual manera la responsabilidad moral y ética inculcada en las diferentes áreas de conocimiento hacen presencia en los diferentes aspectos empresariales en donde se deben tomar decisiones que pueden poner en tela de juicio nuestro buen nombre o capacidad de conciencia del buen nombre del ingeniero.

Sin embargo, la experiencia a nivel de empresa es considerablemente diferente al académico, pues aunque se tienen claros los conceptos teóricos de la universidad, existen factores no comunes a los acostumbrados en un nivel educativo; como la presión del cliente o experiencias empíricas de otras personas que inducen a cambiar la perspectiva del problema y en ocasiones incluso llegar a *quebrar* reglas que hasta antes de la práctica empresarial parecían absurdas, como por ejemplo la creación de una tabla desnormalizada.

La fortaleza que mas se encontró en cada uno de los pasantes fue la habilidad para diseñar y manipular esquemas de bases de datos relacionales, mas aún cuando el sistema gestor de bases de datos Oracle utilizado en Open Systems y muchas otras empresas, es el mismo motor con el que se realizaron las prácticas en la universidad. De igual manera, las buenas prácticas de desarrollo como la documentación y estandarización de los desarrollos realizados en la universidad, facilitaron el acoplamiento al modelo de trabajo en la empresa. También se facilitó el desarrollo de sistemas mediante procesos unificados (UML y RUP) debido a que los conocimientos adquiridos en cuanto a las herramientas como casos de uso, diagramas de secuencia, diagramas de colaboración y pruebas de software fueron inculcados y fuertemente desarrollados por las prácticas académicas.

Por otra parte, refiriéndose al perfil que tiene el profesional egresado o en proceso de graduarse de la Universidad Autónoma de Occidente frente al de otras instituciones de la región o el país, no puede definirse de una manera sólida y precisa, debido en gran parte al poco tiempo de creación del programa y al poco ingreso al mercado laboral de sus graduandos. Teniendo en cuenta lo anterior y lo vivenciado por la pasantía, puede decirse que el ingeniero de informática de la

Universidad Autónoma de Occidente puede destacarse por su formación integral y fácil integración en el mercado laboral; sin dejar a un lado su capacidad de conocimiento técnico y administrativo, al realizar funciones de programador, diseñador de modelos de sistemas o líder de proyectos informáticos.

Hablando de las debilidades que se encontró durante el transcurso de la pasantía en la empresa Open Systems, se debe hacer referencia a la mayor falencia que se encontró por los pasantes en la institución, la falta de desenvolvimiento o inseguridad al expresar sus ideas o pensamientos a otros ingenieros. Fue esta situación la que en mayor medida dificultó la adaptación al proceso laboral, pero que se corrigió durante el transcurso del tiempo, logrando así proyectar mayor seguridad y pro-actividad en las labores desempeñadas. Es importante que en la universidad a través de las diferentes temáticas analizadas en los cursos se integre el desarrollo de habilidades de exposición de conceptos o ideas a nivel grupal (tipo foro), para proyectar mayor seguridad y se enfatice la importancia de analizar los diferentes aspectos de los proyectos y poder encontrar sus beneficios, logrando así vender las ideas más fácilmente.

En relación al acoplamiento con ingenieros y compañeros de trabajo en la empresa no se encontraron dificultades considerables, ya que el ambiente laboral en la empresa siempre da lugar a resolución de dudas, consultas sobre procesos internos de la empresa (refiriéndose al nicho del negocio el cual es el sector de servicios públicos, donde las reglas del negocio son particulares y con muchas variables) y acompañamiento durante todo el proceso de análisis del proyecto. Sin lugar a duda, la mayor ganancia que tuvimos de la interacción con las personas de la empresa fue la riqueza en cuanto a adquisición de conocimientos técnicos, ya que el nivel de técnicas empleadas en los desarrollos es muy enriquecedor para alguien en proceso de formación.

En última instancia, puede concluirse que la pasantía en la empresa Open Systems estuvo caracterizada por el trabajo en equipo y el aumento del conocimiento técnico en los pasantes, ya que son pocas las empresas en donde a los nuevos integrantes se les abre un espacio en su ciclo productivo para el aprendizaje. Siempre se mantuvo la disposición de adquisición de conocimiento y de integración con las demás personas, aumentando siempre el espectro de ingenieros con los que se relacionaba, logrando así mayor desenvolvimiento interpersonal.

21. CONSIDERACIONES ESPECIALES

El objetivo de estas consideraciones es servir como medio para exponer ideas y conocimientos adquiridos durante la realización de la pasantía, que puedan ser aprovechados como retroalimentación por estudiantes que inician un proyecto de desarrollo de software de misión critica como opción de grado. La idea es explicar temas puntuales que surgieron durante la implementación de la aplicación cuyo conocimiento previo hubiese facilitado el avance del presente proyecto de grado o tal vez puedan servir como apoyo para el desarrollo de futuros proyectos.

En primer lugar es indispensable investigar sobre software de misión crítica ya que a nivel universitario no se profundiza en esta temática, es importante adquirir conciencia de la seriedad que implica desarrollar este tipo de software pues son aplicaciones que deben estar disponibles todos los días las 24 horas. En este orden de ideas, se espera que la ejecución de cualquier proceso en este tipo de aplicaciones consuma el menor tiempo posible haciendo uso efectivo de los recursos de máquina (procesador, memoria), evitando así, congestionar el acceso a la aplicación. Es en este punto donde el rendimiento o performance se vuelve un factor dominante para determinar el éxito de una aplicación de misión crítica; por lo tanto, también es necesario instruirse acerca de técnicas de rendimiento como afinamiento de sentencias SQL (statement tunning) y manejo de índices para las consultas en la base de datos.

Otra de las consideraciones descubierta durante el desarrollo del proyecto también tiene que ver con el software de misión crítica, pues dado que en estas aplicaciones se busca un desempeño óptimo en tiempos de ejecución y en consumo de recursos, en ocasiones se recurrió a incumplir reglas teóricas (Normalización de bases de datos, creación de llaves foráneas en las tablas, entre otras) como un sacrificio en función del performance del software; fue necesario entonces para el grupo de trabajo superar el impacto que significó quebrantar algunas de las teorías enseñadas en la universidad, para pasar a realizar un balance entre lo aprendido teóricamente en el ámbito académico con las técnicas prácticas que pueden llegar a aprenderse en el ámbito empresarial.

BIBLIOGRAFÍA

BRITO G., Alexandra Del Valle. Historia de Oracle. [en línea]. Mexico: Monografías, 1997. [Consultado 20 Octubre, 2006]. Disponible en Internet: http://www.monografias.com/trabajos25/oracle/oracle.shtml

COMUNICACIONES DE TELEFÓNICA I+D. El mundo de los OSS y BSS ante las nuevas exigencias de negocio. [en línea]. España: Telefónica, 2004. [Consultado 12 Septiembre 2006]. Disponible en Internet: http://www.tid.es/presencia/publicaciones

GURRY, Mark; CORRIGAN, Peter. Oracle Performance Tuning. 2 ed. New York: O'Reilly Media, 1996. 960 p.

HOLDSWORTH, Andrew. Oracle9i Database Performance Planning, Release 2 (9.2). California: Oracle, 2002. 66 p.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. The Unified Software Development Process. Boston: Addison-Wesley Professional, 1999. 463 p.

JAMEN, LISA M. Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack, Release 9.0.1. California: Oracle, 2001. 282 p.

LAKSHMAN, Bulusu. Oracle Developer Forms Techniques. Washington: Sams, 2000. 250 p.

LULUSHI, Albert. Oracle Forms Developer's Handbook. New York: Pearson Education, 2000. 1008 p.

LUSCHER, Lenore. Oracle9i Database Performance Tuning Guide and Reference, Release 2 (9.2). California: Oracle, 2002. 818 p.

MAGEE, BILL. An Introduction to Oracle SQL Statement Tuning. [en línea]. Reino Unido: Bill Magee, 2003. [Consultado 10 Abril 2006]. Disponible en Internet: http://www.billmagee.co.uk/oracle/sqltune/

MILLSAP, Cary; HOLT, Jeff. Optimizing Oracle Performance. New York: O'Reilly Media, 2003. 388 p.

OPEN SYSTEMS. Información de la Compañía. [en línea]. Colombia: Open Systems, 2000. [Consultado 20 Octubre 2006]. Disponible en Internet: http://openinternational.com/spanish/company/index.htm

ORACLE. Oracle DataBase Server Documentation. [en línea]. Miami: Oracle Corporation, 2000. [Consultado 2 Junio 2006]. Disponible en Internet: http://www.oracle.com/technology/products/database/oracle10g/index.html

ORACLE. Oracle Forms Documentation. [en línea]. Miami: Oracle Corporation, 2000. [Consultado 13 Mayo 2006]. Disponible en Internet: http://www.oracle.com/technology/products/forms/index.html

PACK, Charles. OCP Oracle9i Database: Performance Tuning Exam Guide. Londres: McGraw-Hill Osborne Media, 2000. 470 p.

UNIVERSIDAD DE JAÉN. Programación de Bases de Datos Oracle. [en línea]. España: Universidad de Jaén, 2000. [Consultado 1 Julio 2006]. Disponible en Internet: http://wwwdi.ujaen.es/~molina/bd1/apuntes_prac.pdf

UNTERSCHÜTZ, THOMAS. Operations Support Systems for NGN. [en línea]. Dinamarca: Eurescom, 2005. [Consultado 18 Septiembre 2006]. Disponible en Internet: http://www.eurescom.de/message/messageDec2004/Operation_Support_Systems_for_NGN.asp

ANEXO A. PAPER DE PASANTÍA

DESARROLLO DE INTERFAZ OPENSMARTFLEX – ENERGIS: MODULOS DE PETICIONES Y CONSUMOS

DAVID ARIAS VILLALOBOS
JULIO CESAR LEDESMA ROMÁN
ANDRÉS FABIÁN SÁNCHEZ CIFUENTES

Universidad Autónoma de Occidente
Campus Valle del Lili KM 2 Vía Jamundi
buzon@uao.edu.com
Cali - Colombia

Resumen: Un proyecto de desarrollo de software de misión crítica para sistemas robustos donde su operación nunca se detiene, requiere de un alto conocimiento de las herramientas de desarrollo de software que se manejen y los sistemas operativos donde éste funcione. Para llevar a cabo un desarrollo de esta envergadura, es necesario una excelente planeación de cada una de las etapas del proyecto: análisis, diseño y construcción, las cuales son cruciales para este tipo de proyectos.

Adicionalmente a estos pasos se debe poseer un know how propio desarrollado a través del tiempo, la investigación en sistemas críticos y retroalimentación recibida por los clientes. Tal es el caso de la empresa en la que se desarrollo esta pasantía, Open Systems, que es una casa de software dedicada al desarrollo de software para empresas de servicios públicos. Esta empresa con su producto OpenSmartFlex ha alcanzando una solución de software de misión crítica con calidad, que cubre los aspectos necesarios para el funcionamiento de una empresa dedicada al sector de servicios públicos. Sin embargo, no todas las áreas de una empresa pueden cubrirse mediante un solo sistema, ya sea por la existencia de sistemas que han perdurado en el tiempo o por asuntos externos a la implantación del nuevo software; tal es el caso del requerimiento cubierto por esta pasantía, en donde se busca comunicar la aplicación OpenSmartFlex con el sistema de gestión existente en EMCALI E.I.C.E. Energís. Mediante esta integración se garantiza la unicidad de fuente de datos y la sincronía de datos que alimentan los posteriores procesos de gestión que se vienen realizando mediante el sistema externo.

Este artículo plasma importantes experiencias vividas durante este proyecto, con el objetivo que estudiantes de informática y carreras a fines que decidan dedicarse al desarrollo de software de misión crítica, basados en bases de datos Oracle, conozcan algunos tips que les ayuden en su vida profesional.

INTRODUCCIÓN

La empresa de servicios públicos de EMCALI, maneja un gran volumen de información en cuanto a la administración de sus procesos comerciales y de operación, razón por la cual debe contar con sistemas de información robustos que faciliten la gestión de dichas tareas. Open Systems es una empresa que provee soluciones de software especializado para los sectores de las telecomunicaciones y servicios públicos domiciliarios (Telco & Utilities), por lo cual Emcali ha adquirido su producto OpenSmartFlex con el fin de suplir sus necesidades.

Aparte de las soluciones integradas que ofrece el producto OpenSmartFlex, también se ha requerido una solución para una necesidad particular. Este tipo de requerimientos se demoninan customizaciones, cuyo desarrollo se considera de misión crítica para el negocio. Esta customización consiste en la integración del sistema adquirido con otro ya existente, para el caso de estudio de esta pasantia Energis, utilizado actualmente para gestionar labores de campo, el cual requiere de la publicación de los consumos medidos y trámites de los servicios suscritos de los clientes de EMCALI.

La función de este proyecto fue desarrollar de esta customización bajo los estándares de Open Systems, para lo cual fue necesario un entrenamiento en cuanto a sus herramientas de desarrollo, estándares de calidad, arquitectura de producto y optimización de recursos.

DESARROLLO DEL PROYECTO

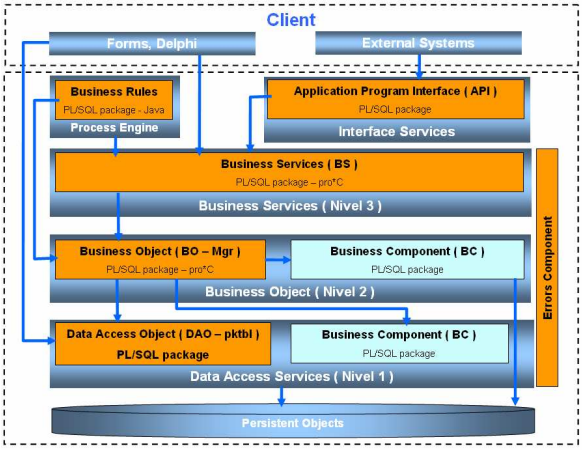
El proyecto inicia con el levantamiento de requerimientos realizado por los ingenieros de servicios profesionales situados en el cliente EMCALI. Este requerimiento es estructurado y documentado mediante el marco de estándares de Open; a través del ciclo productivo dicho documento se va madurando y especificando, al pasar por las etapas de análisis, diseño y desarrollo; obteniendo así una documentación consolidada del desarrollo a realizar que podrá ser utilizada para futuros proyectos o modificaciones que lo requieran.

El primer paso fue el conocimiento de la arquitectura del producto OpenSmartFlex, la cual se tuvo en cuenta para el diseño de la solución.

Esta es una arquitectura de tres capas donde la primera se encarga de los aspectos relacionados a accesos en la base de datos. La

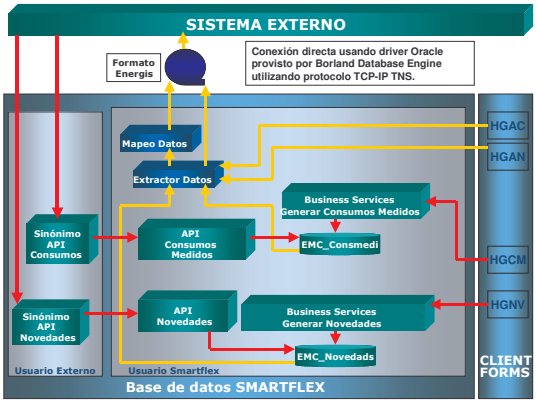
segunda capa contiene componentes relacionados con la lógica del negocio y en la tercera se encuentran los servicios publicados a los procesos. Esta arquitectura también cuenta con componentes para el manejo de errores, interfaces con sistemas externos y configuración de las reglas de negocio.

Figura 1. Arquitectura OpenSmartFlex



Siendo consecuentes con la arquitectura de producto se plantea la el modelo de la solución de la integración para el cliente, donde se especifican los puntos de contacto con el sistema externo.

Figura 2. Arquitectura de la solución.



En etapa de análisis se detallan los casos de uso que se obtuvieron del requerimiento inicial, los cuales sirven como herramienta para identificar procesos involucrados en la publicación de los consumos medidos y trámites de los servicios suscritos.

Durante la etapa de diseño se definen los objetos nuevos, sus relaciones y las posibles integraciones con los objetos existentes. De igual manera se diseñan las Interfaces Graficas de Usuario. Todo este diseño esta encapsulado dentro de los estándares de desarrollo definidos por la empresa Open Systems.

Cabe resaltar el énfasis académico impulsado en la universidad con respecto a los componentes necesarios para el modelamiento de sistemas, como son el Modelo de Desarrollo Unificado (UML) y todas sus herramientas, en donde se pueden nombrar Casos de Uso y Diagramas de Secuencia. De igual manera se recalca el modelamiento en bases de datos para obtener una fuente de datos consistente y eficiente.

Después de un desarrollo inicial se impulso la mejora de la solución, respecto a tiempos de respuesta. Para lograr este objetivo se hizo uso de las herramientas EXPLAIN PLAN y TKPROF que ofrece Oracle, como también del know how de la empresa desarrolladora de software. La primera herramienta determina el plan de ejecución de la sentencia SQL, es decir, indica el camino que debe seguir la sentencia involucrada.

La segunda potente herramienta es mas completa que la primera porque ofrece estadísticas de cómo se ejecutó la sentencia; de igual manera permite obtener información de las sentencias ejecutadas como: Tiempo de consumo en la CPU, tiempo de consumo en disco, consultas que realiza, filas que afecta, filas que retorna, cantidad de parseos que se hacen en la sentencia, fetchs que se hacen con la sentencia.

La combinación de estas dos herramientas más el concepto de GRANULARIDAD propio de Open Systems, el cual se puede explicar mediante una simple analogía: “Usar los recursos que se necesiten, sin necesidad de ocupar los demás registros”, y otras más técnicas de rendimiento lograron que se obtuviera un software de misión critica con calidad.

CONCLUSIONES

Se pudo notar como una metodología para el desarrollo de software facilita el proceso en cada una de las etapas del mismo.

Mediante el uso de estándares, know how de Open Systems y conocimientos específicos en Oracle y sistemas operativos se logró obtener un software de misión crítica con calidad.

El entrenamiento, conocimiento y experiencia adquirida en la empresa desarrolladora fue muy enriquecedor para las vidas de los pasantes en el ámbito profesional y personal, ya que cambiaron y reestructuraron los modelos teóricos analizados en la universidad.

REFERENCIAS

BRITO G., Alexandra Del Valle. Historia de Oracle. [en línea]. Mexico: Monografías, 1997. [Consultado 20 Octubre, 2006]. Disponible en Internet: http://www.monografias.com/trabajos25/oracle/oracle.shtml

COMUNICACIONES DE TELEFÓNICA I+D. El mundo de los OSS y BSS ante las nuevas exigencias de negocio. [en línea]. España: Telefónica, 2004. [Consultado 12 Septiembre 2006]. Disponible en Internet: http://www.tid.es/presencia/publicaciones

GURRY, Mark; CORRIGAN, Peter. Oracle Performance Tuning. 2 ed. New York: O'Reilly Media, 1996. 960 p.

HOLDSWORTH, Andrew. Oracle9i Database Performance Planning, Release 2 (9.2). California: Oracle, 2002. 66 p.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. The Unified Software Development Process. Boston: Addison-Wesley Professional, 1999. 463 p.

JAMEN, LISA M. Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack, Release 9.0.1. California: Oracle, 2001. 282 p.

LAKSHMAN, Bulusu. Oracle Developer Forms Techniques. Washington: Sams, 2000. 250 p.

LULUSHI, Albert. Oracle Forms Developer's Handbook. New York: Pearson Education, 2000. 1008 p.

LUSCHER, Lenore. Oracle9i Database Performance Tuning Guide and Reference, Release 2 (9.2). California: Oracle, 2002. 818 p.

MAGEE, BILL. An Introduction to Oracle SQL Statement Tuning. [en línea]. Reino Unido: Bill Magee, 2003. [Consultado 10 Abril 2006]. Disponible en Internet: http://www.billmagee.co.uk/oracle/sqltune/

MILLSAP, Cary; HOLT, Jeff. Optimizing Oracle Performance. New York: O'Reilly Media, 2003. 388 p.

OPEN SYSTEMS. Información de la Compañía. [en línea]. Colombia: Open Systems, 2000. [Consultado 20 Octubre 2006]. Disponible en Internet: http://openinternational.com/spanish/company/index.htm

ORACLE. Oracle DataBase Server Documentation. [en línea]. Miami: Oracle Corporation, 2000. [Consultado 2 Junio 2006]. Disponible en Internet: http://www.oracle.com/technology/products/database/oracle10g/index.html

ORACLE. Oracle Forms Documentation. [en línea]. Miami: Oracle Corporation, 2000. [Consultado 13 Mayo 2006]. Disponible en Internet: http://www.oracle.com/technology/products/forms/index.html

PACK, Charles. OCP Oracle9i Database: Performance Tuning Exam Guide. Londres: McGraw-Hill Osborne Media, 2000. 470 p.

UNIVERSIDAD DE JAÉN. Programación de Bases de Datos Oracle. [en línea]. España: Universidad de Jaén, 2000. [Consultado 1 Julio 2006]. Disponible en Internet: http://wwwdi.ujaen.es/~molina/bd1/apuntes_prac.pdf

UNTERSCHÜTZ, THOMAS. Operations Support Systems for NGN. [en línea]. Dinamarca: Eurescom, 2005. [Consultado 18 Septiembre 2006]. Disponible en Internet: http://www.eurescom.de/message/messageDec2004/Operation_Support_Systems_for_NGN.asp