UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

SISTEMAS DE CONCILIACION FINANCIERA

Autora: Ana Cerro Campón

Tutor: Manuel Velasco de Diego

A mi madre y a mis tías (Susana y Trini) por haberme
inculcado la ilusión por aprender,
y por haberme educado como persona

AGRADECIMIENTOS.

Dar las gracias a las personas que han contribuido que finalice una de los logros de mi vida, que ha sido finalizar la carrera y convertirme en Ingeniera Informática.

En primer lugar, quiero dar mi más sincero agradecimiento a mi madre que ella sola ha luchado por brindarme la oportunidad de estudiar, por todo su cariño y por la confianza en mi. A mi tía Susana, por su apoyo y sus consejos, y a mi tía Trini haya donde quiera que este.

A mi hermana María, por aguantarme a lo largo de los periodos de exámenes, por su paciencia y su cariño en todos los momentos de mi vida. A Dani, por esperar este momento tanto como yo, y a su familia por apoyarme.

Gracias a Manuel Velasco, mi tutor, por ayudarme hacer fácil algo difícil, por sus sugerencias y por su colaboración. Gracias por la paciencia durante todo este año.

Gracias a todos mis amigos y compañeros de universidad y de la empresa, con los que he compartido momentos inolvidables en estos años de carrera.

A todo ellos GRACIAS ¡¡

ÍNDICE

Marco del Proyecto:

El sistema de conciliaciones financieras, es un sistema que permite conciliar los datos extraídos en ficheros pertenecientes a las distintas aplicaciones donde se genera la mayor parte de la información perteneciente a la operativa bancaria, tales como certificación de precios, cambios de divisa, etc.

La información a conciliar pertenece a extracciones de aplicaciones distintas y que habitualmente deben de certificar la misma información, la necesidad de realizar la conciliación se basa en comprobar que ambas aplicaciones, aunque con características y funciones distintas, no pierden o duplican información.

La necesidad de certificar la información para las aplicaciones dentro de unos horarios y los grandes volúmenes en algunos de los productos hacen de los sistemas de conciliación una necesidad para el buen funcionamientos de la información de los bancos, y un control de certificación de operativa bancaria de precios.

La definición de un adecuado proceso de Conciliación contribuye a Mejorar el entendimiento de riesgos asociados a controles generales y establecimiento de acciones para eliminarlos.

- Establecer los procesos que generen evidencias de auditoría.
- Definir controles básicos, procesos de autorización y de distribución de responsabilidades.
- Establecer una estructura fiable de control interno y procedimientos automatizados para el reporte de información financiera.

A lo largo de la vida financiera siempre han existido varios tipos de herramientas de conciliación, estas herramientas han ido evolucionando. En este caso vamos a tratar la última herramienta en el mercado, es la herramienta de sistemas de conciliación financiera TLM.

En este sistema conviven una serie de procesos y aplicaciones que van ejecutando los distintos pasos de la conciliación. Este sistema consta de varias aplicaciones funcionales:

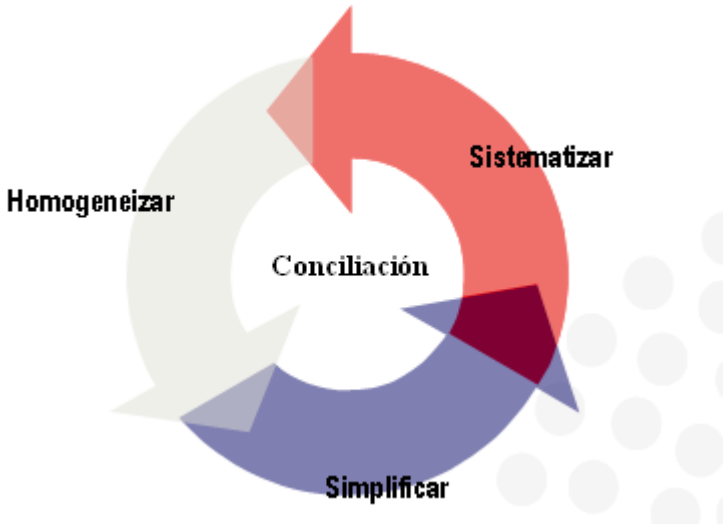
- SmartSchema: Esta aplicación nos permite diseñar el esquema de carga para extracción de una manera específica, según la información y estructuración de los datos recibidos.

- TLM (cliente Pesado):

  Esta aplicación nos permite crear las reglas de cruce definidas por analistas de banca. Además podremos crear los informes que mostraran el resultado de la conciliación.

- TLM (Cliente ligero):

  Es la misma aplicación que la anterior con la única diferencia de que se accederá a los datos mediante vía WEB, y únicamente se podrá consultar los informes finales a asociados al usuario.



Capitulo I

OBJETIVOS

1.1. OBJETIVO:

- Generación del esquema de carga para los ficheros en la base de datos mediante el smartschema. Creación de los nodos y definición de signos de los nodos. Los signos asociados a los nodos se definirán según criterio de carga y cruce para los ficheros de partida y contrapartida.

- Creación de los atributos necesarios para la carga y para las reglas de cruce, asociados a los nodos generados a primer nivel o a subniveles. Selección de tipo de campos para los atributos y de su tamaño, asociación de propiedades para su utilización en la herramienta de conciliación, TLM. Y definición de decodes para alguno de los tipos de campos.

- Generación del esquema lógico de carga en la base de datos según los nodos definidos sus signos y sus atributos sobre la base del esquema definido en el smartschema, KB.

- Adaptación del KB a las cabeceras del fichero y a otros elementos como entrada de constantes, o modificación a un lenguaje más descriptivo.

- Creación de cuenta de cargas en TLM asociada a los nodos creados previamente en el SmartSchema.

- Creación de entrada en el Host utilizado por los procesos denominados Fetchit, en el Fetchit llamamos a los procesos GEMS. Los GEMS son demonios que esperan a ser llamados desde el Fetchit para realizar la carga en la base de datos desde el nodo indicado.

- Creación de ficheros. trg que seleccionan el patrón de fichero que vamos a llamar en el Fetchit.

- Creación de la entrada en el Host, definiendo el nombre para su ejecución, la ruta donde se encuentran los ficheros, el patrón a seleccionar o ficheros .trg , la ruta donde se debe dejar el fichero tras su carga , el nodo donde debe cargar el fichero , y por ultimo la base de datos.

- En caso de ser necesario el reformateo del fichero, porque la extracción no sea del todo correcta, queremos manipular algún campo, etc. Necesitamos hacer un pequeño programa con lenguaje AWK, que deberemos de indicar en el Host. Este reformateo se ejecutara cuando llamemos al Fetchit.

- Una vez comprobada que la carga de los ficheros es correcta definimos las reglas de cruce en TLM.

- Una vez definidas las reglas de cruce, creamos la unidad principal que llama a todas la lista de reglas creadas, que son las iniciaciones.

- Creamos los informes para mostrar los resultados de la conciliación, en algunos casos se generan dashboards, los dashboards son informes creados en TLM. Estos informes pueden ser vistos desde el cliente ligero o acceso WEB. En algunos casos el usuario prefiere ver los informes directamente enviados en correos con datos adjuntos. Para ello creamos vistas para modelar el formato de la información que el cliente desea obtener, directamente desde la base de datos.

- Creamos un script Bourne shell. En el script comprobamos los procesos, los nombres de los ficheros, la fecha de estos, comprobamos la carga correcta de los ficheros en base de datos, insertamos en base de datos la iniciación del cruce, insertamos en la tabla estadística algunos datos de la conciliación y por ultimo mandamos el informe, o informamos de que la conciliación ha sido correcta.

1.2. Arquitectura del sistema:

La arquitectura de TLM, es una arquitectura SOA (Arquitectura Orientada a Servicios).

Los beneficios desde el punto de vista tecnológico son claros:

- Favorece la reutilización y la reducción de tiempos.
- Aumenta el grado de reutilización al desacoplar las capas de una aplicación.
- Permite reutilizar las aplicaciones existentes mediante la encapsulación en servicios.
- Permite la utilización de servicios de terceros.
- Permite reaprovechar las plataformas existentes.

Aumenta la flexibilidad:

- Simplifica la adaptación de los sistemas existentes.
- Evita el desarrollo de interfaces punto a punto entre los sistemas.
- Aumenta la interoperabilidad entre sistemas, permitiendo tanto la externalización como la prestación de servicios.

Mejora la productividad de los procesos:

- Aumenta el nivel de automatización de los procesos, reduciendo el número de actividades manuales.
- Permite monitorizar la actividad del negocio (cuadros de mando).
- Permite realizar un análisis estadístico de los flujos de negocio reales basándose en indicadores clave de negocio, permitiendo la identificación de puntos de mejora a optimizar.
- Permite evaluar el impacto y beneficio de variantes en los procesos mediante simulación.

Mejora el proceso de construcción de software:

- Favorece la industrialización.
- Mejora la especificación de los requerimientos de negocio.
- Proporciona una filosofía de desarrollo común a todos los negocios y canales.
- Mejora la calidad.
- Desacopla el desarrollo de servicios y de procesos. Mejora el mantenimiento (procesos autodocumentados).

Mejora la usabilidad de las aplicaciones:

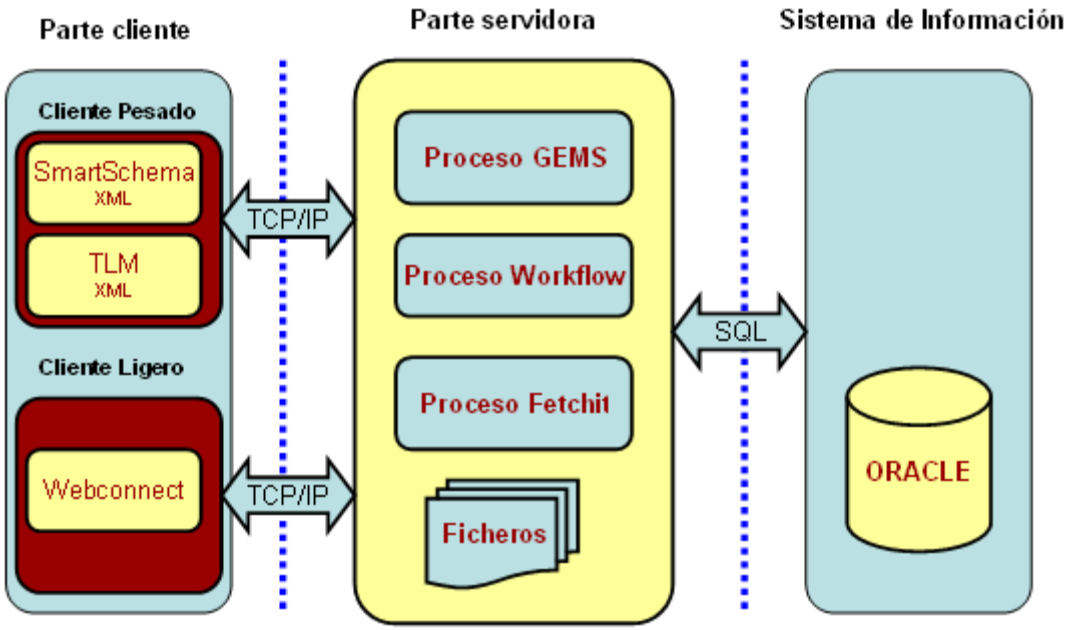
- Permite presentar al usuario la información dispersa en distintos sistemas y de forma integrada.

- Permite alcanzar un mayor nivel de automatismo en las aplicaciones en procesos complejos de workflow.

  Son aplicaciones cooperativas entre si. Un workflow es la representación computarizada de un proceso de negocio, el cual especifica las distintas actividades que forman dicho proceso, el orden en que estas deben ejecutarse, el flujo de datos entre ellas y los múltiples agentes que colaboran para llevar a cabo el proceso.

- Permite utilizar tecnologías de presentación avanzadas como Web 2 .0.

Esta arquitectura SOA utiliza de modo eficiente el siguiente Webservice.

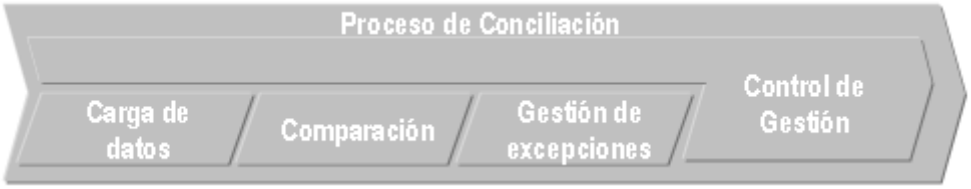


Capitulo II

ESTADO EN CUESTION

2.1 Aplicaciones Propias del Sistema de Conciliación Financiera:

La reingeniería del Proceso de Conciliación se detendrá en los distintos pasos del proceso. Los sistemas de conciliación financiera



Para cada uno de estos pasos del proceso de conciliación necesitaremos utilizar las distintas aplicaciones que componen el sistema de conciliaciones financieras. El sistema de conciliación se basa en dos aplicaciones. Ambas aplicaciones se expondrán a continuación.

Las dos aplicaciones que se exponen se encargan de modelar la carga del fichero en la base de datos, y programar las reglas que se aplicaran sobre los datos cargados.

Carga de Datos:

Mediante el SmartSchema podremos modelizar y asociar el diseño del fichero a la carga en la base de datos.

- Recibir datos para comparar
- Resolver incidencias relativas a dicha recepción
- Convertir a formato de conciliación y cargar datos en el sistema

Comparación:

En la comparación utilizaremos la aplicación de modelizado de reglas a aplicar sobre los datos cargados previamente.

2.2 ¿Qué es SmartSchema?

El SmartSchema, es el interfaz grafico para la modelización personalizada de una estructura acoplada a las necesidades de carga de las extracciones recibidas y la estructura existente en la base de datos.
Los esquemas principales en la base de datos se dividen en universos en el SmartSchema, estos esquemas se asocian a tablas o vistas.

Los principales universos TLM son los siguientes:

- Message Type
- Message Header
- Item
- Case Summary

Estos universos son tablas asociadas entre sí y de una gran importancia para la carga de las extracciones.

Los demás universos creados suelen ser vistas, la creación de estos universos surge con la necesidad de mostrar los datos de una manera determinada.

2.2.1 ¿Cómo se estructura el SmartSchema?

Dentro de los universos, podemos crear nodos. En la creación de estos nodos hay que parametrizar una serie de propiedades en función de las extracciones de los ficheros recibidas y de las futuras reglas de cruce. Todas las extracciones deberán de contener unos campos obligatorios que decidirán la parametrizacion de estos campos, como son el lado, la cuenta, y el signo.

En los nodos podemos crear los atributos necesarios tanto paral la carga de las extracciones como para utilizarlos en TLM. En el SmartSchema se definirá el nombre, las propiedades dentro de la herramienta TLM que rigen su comportamiento tanto como atributo como dentro de cada uno de los objetos de TLM. Podremos definir el tamaño del atributo, el tipo, etc. El valor que se puede asignar a un atributo está determinada por sus datos tipo y longitud.

Los tipos de datos en el SmartSchema son:

- Tipo Date, formato de fechas.

- Tipo String, es el tipo indicado para las cadenas de texto.

  Donde podremos definir el tamaño, nunca superando el máximo establecido en el modelo de la base de datos.

- Tipo Flag, es el tipo indicado para los enteros.

  Este tipo de valor se puede decodear. El decode no afecta a la carga en la base de datos, sino a su utilización en TLM. Si se definen dos posibles valores para un tipo de campo flag y se decodean, al ser utilizado este atributo en TLM para la creación de reglas no mostrará sus posibles valores numéricos, pero si sus decodes.

  Teniendo en cuenta que los informe se muestran a través de la herramienta TLM, su uso es bastante practico evitando mostrar valores numéricos y su valor lógico.

- Tipo amount, se utiliza para decimales.

Los universos en TLM son jerárquicos y a su vez para un nodo podemos crear subnodos. La necesidad de crear subnodos depende de la información a cargar y las necesidades de cruce. Las propiedades de los nodos padre, así como sus atributos, son heredados para los subnodos.

Cuando creamos un nodo en el universo item necesitamos relacionarlo con Message Type y Message Header. Una vez relacionados podemos extraer el KB.

Las propiedades que se pueden asignar se pueden dividir en dos:

Las propiedades para el nodo: Las propiedades para los nodos deben de ser definidas antes de ser utilizadas en TLM, y de generar el KB.
Los nodos deben de ser definidos para cada nodo, y debe de tener un valor de:

- **LADO:**

  - L (Ledger)
  - S (Statement)
  - B (Broker)

El lado indica el tipo de elemento que puede ser cargado en cada nodo, los ficheros que se carguen en un nodo con lado L, deberán de contener el atributo Lado con el valor L .

Para determinar el lado en las extracciones deberemos de tener en cuenta si queremos que el fichero A cruce con el fichero B, cada uno deberá de tener asignado un lado distinto.

- **SIGNO:**

El signo junto con la definición del nodo, indica el comportamiento de los elementos que contengan valores en TLM.
Los signos se definen en:

- C (Credit)
- D (Debit)
- U (Undefined)

- **Acción:**

Esta definición es necesaria cuando utilicemos funciones de TLM tales como NET, NETPERCENT, SUM, ect. Indican suma o resta de un total general. Se definen con:

- +
- –

Las propiedades para los atributos:

Las propiedades para los atributos asignados a un nodo regirán el comportamiento de estos atributos dentro de la herramienta TLM. Se divide en varias propiedades.

- **Propiedades de rules:**

Se dividen según los objetos rules de TLM, e indicaran si el atributo puede ser utilizado en las distintas rules, como Pass_quality, scope, population, etc.

- **Propiedades de visualización:**

Indican si el atributo es privado, únicamente siendo visible para el desarrollador, o también público siendo visible en informe consultados por el cliente.

- **Propiedades de rango:**

Esta propiedad se utiliza con los datos de tipo amount, pudiendo seleccionar la cantidad de decimales a mostrar.

- **Propiedades de Eventos:**

Al igual que las propiedades de las Rule, si son seleccionados los atributos serán visibles en los eventos de TLM.

Asignación de ID:

Cuando creamos un atributo en el SmartSchema, este automáticamente le asigna un ID, que será único para cada atributo. Este atributo será utililizado TLM para reconocer todas propiedades tanto físicas como lógicas del atributo. Por ello solamente deben crearse atributos en el entorno de desarrollo, ya que de no ser así SmartSchema podría asignar un ID que esta siendo utilizado en desarrollo y que supondrá un conflicto a la hora de importar el esquema de entorno.

Para exportar el esquema SmartSchema cuenta con una opción para genera un archivo xml con toda la información de todos los universos. Para importar debemos de seleccionar el archivo xml previamente importado y seleccionar los universos y nodos a importar.

2.3 ¿Qué es el KB?

El KB es un archivo que contiene la relación entre el diseño con el esquema en el SmartSchema, los campos del fichero y la base de datos. Este mapa de carga indica una relación entre los atributos de los ficheros, mediante las cabeceras, separadores y los atributos donde debe de cargar en la base de datos. Además indica las propiedades del nodo donde vamos a realizar la carga mediante los signos de cruce.

El KB se subdivide en varias partes, que van ejecutándose de modo secuencial, se subdivide en:

Feed:

Personalizamos el nombre del Feed, que posteriormente utilizaremos en el Host para indicar en que lugar queremos cargar los ficheros. Y que relacionara los demás elementos que se Irán llamando.

```
feed FD_AC_BS
        description "Columnar 10060"
        eoh  ""
        sof  ":"
        delimiter  ";"
        type  columnar
        messages M_AC_BS
```

- EOH: El carácter que marca el final de la cabecera del fichero.
- SOF : el carácter que marca el comienzo del feed
- Delimiter: el carácter que indica la separación entre campo en el KB.
- Messages: se define posteriormente en el KB, e indica la asociación con la message_header, y de los ID perteneciente a él.

Message:

El campo Message de la definición de Feed redirecciona a esta parametrización, donde indicamos el nodo donde queremos cargar, y los signos asignados en función del campo MLSideX, ILSideXSignX, donde X dependiendo del número indicara el tipo de signo, previamente definido en el SmartSchema. Además MLSide1 contiene un número el cual asocia el universo ítem con el universo message_header y Message_type.

```
message M_AC_BS
        description "Flat File AC_BS"
        tag "AC_BS"
        MLSidel 10060
        ILSidelSignl 10072
        type columnar
        category C_GENERAL
        endfn Gms_SubmitMessage
        blocks B_AC_BS_A, B_AC_BS_B
```

El campo blocks redirecciona a la parte de definición de bloques.

Blocks:

Los blocks son el listado de atributos definidos para este feed. Se dividen en dos partes, la parte de los campos obligatorios, los cuales deben de serlo también para los ficheros enviados al sistema de conciliación. Y la parte opcional donde indicamos los campos que contiene el fichero a cargar. Podemos observar la diferencia en mandatory en el bloque A, y optional en el bloque B.

Los fields redireccionan el KB a la parte de asociación del nombre del atributo en la cabecera del fichero, con el nombre asociado en el SmartSchema y el nombre físico en la base de datos.

```
block B_AC_BS_A
        description "MTAC Block A"
        mandatory
        endfn Gms_NewMessage
        fields F_AC_BS_A_HEADER_SIDE, F_AC_BS_A_HEADER_MESSAGE_TYPE, F_AC_BS_A_HEADER_FEED_ID,
               F_AC_BS_A_HEADER_STATEMENT_DATE, F_AC_BS_A_HEADER_STATEMENT_NUMBER,
               F_AC_BS_A_HEADER_STATEMENT_PAGE

block B_AC_BS_B
        description "MTAC_BS Block B"
        repeat
        optional
        endfn Gms_AddItem
        fields  F_AC_BS_B_FCODE, F_AC_BS_B_FNAME, F_AC_BS_B_EDATE,
                F_AC_BS_B_EVALUE, F_AC_BS_B_ITEM_SIGN
```

Fields Opcionales:

Se indica el nombre asociado al nodo, la B indicando el bloque donde se encuentra, y el nombre asignado.

```
field F_AC_BS_B_EDATE
        description "EDate"
        tag ":EDATE:"
        optional
        category C_GENERAL
        split into
                item.date_5 as "%[^\n]"
```

En desciption, mostrará el nombre asignado en el SmartSchema. El Tag debe ser exactamente igual a la cabecera del fichero. Y por ultimo la asociación con el nombre en la base de datos se definirá en el Split into.

Fields Obligatorios:

A diferencia de los anteriores estos atributos se heredan del universo item, no se definen para nuestro nodo, y deben de estar contenidos en todos los KB, y en los ficheros. Indican en el nombre su pertenencia al bloque A, y en este caso pasan de contener la propiedad optional a mandatory.

```
field F_AC_BS_A_HEADER_STATEMENT_DATE
        description "Header Statement Date"
        tag ":FECHA:"
        mandatory
        category C_GENERAL
        split into
                msghdr.stmt_date as "%[^\n]"
```

2.4 ¿Qué es TLM?

TLM es la aplicación donde creamos las reglas de cruce para el modelo previamente diseñado en el SmartSchema.

Tanto el modelado del esquema, como las propiedades seleccionadas en el SmartSchema regirán el comportamiento de los atributos asignados al nodo dentro de TLM, así como de los propios nodos.

Todas las reglas tienen que definirse dentro de un universo, y a su vez el nivel dentro del universo donde queremos que se ejecuten estas reglas. Una regla creada en el ámbito de un subnodo no podrá seleccionarse en un nivel superior, sin embargo, si esta creada en un nivel superior si podrá ser utilizada en un nivel inferior.

Dentro de TLM diferenciamos el cliente pesado, donde se parametriza la conciliación y el cliente ligero, de uso exclusivo para el cliente, donde podrá consultar los resultados obtenidos, en informe que previamente se desarrollaran en el cliente pesado.

2.4.1 ¿Cómo funciona TLM?

Permite asociar mediante cuentas la información de los ficheros recibidos y cargados en la base de datos, con la secuencia de reglas de TLM.

Estas cuentas deben de ser creadas tanto en TLM, como venir informada en alguno de los atributos de los ficheros.

Creación de cuentas:

En primer lugar deberemos de tener creada una categoría en TLM. En general suele haber una categoría por proyecto, simplemente sirve para agrupar todas las cuentas necesarias para un producto.

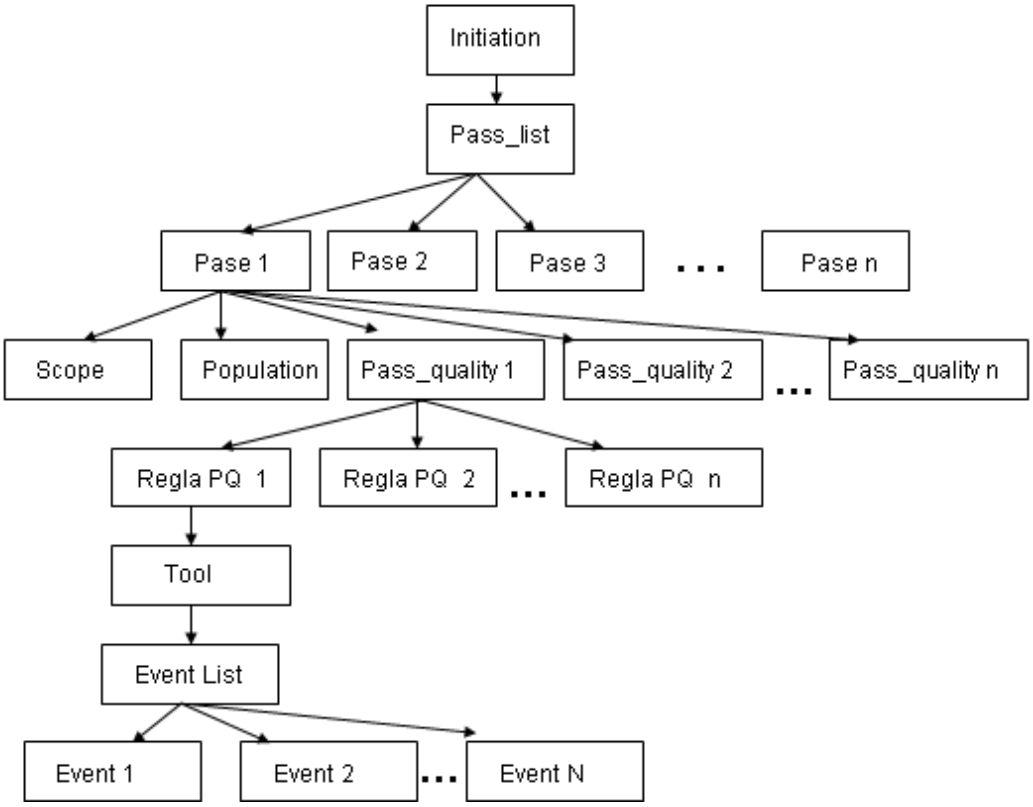
La definición tanto de la categoría como de la cuenta quedará recogida en la tabla Bank, el nombre de la categoría será el agent_code, y el nombre de las cuentas asociadas será el local_acc_no. Posteriormente asignaremos los messages feed a los nombres de las cuentas, que realmente es donde indicamos mediante

el feed del fichero, el nodo o ls_type en que nodo queremos cargar la información que vamos a utilizar.

La cuenta junto con el gin, que es un id único por registro cargado, hacen que cada uno de los registros sea único en la base de datos. Las cuentas diferencian al producto, y son claves primarias e índices. Por lo que su utilizan para facilitar el acceso a los datos. Si estos datos no están correctamente relacionados en el SmartSchema, KB y base de datos (en base de datos se crean desde TLM), la carga no podrá asociarse a TLM.

Una vez asociada la información de la base de datos con TLM, se procede a parametrizar las reglas de cruce con TLM.

Existen distintas jerarquías de objetos que engloban otros objetos, la unidad principal de este conjunto de reglas se denomina Initiation:



La Initiation esta asociada a otros objetos. Estos objetos van llamando a los objetos integrados en estos a un nivel inferior. Las initiation redireccionan la ejecución llamando al siguiente objeto con el que esta enlazado que es el Pass List. El Pass List contiene una lista de pases asociados. Los pases son la unidad principal donde se engloba los objetos que definen las reglas de cruce.

En los pases se selecciona la información, se agrupa, se realizan comprobaciones según la agrupación y las reglas definidas. Si las agrupaciones cumplen las reglas se desenlaza una serie de eventos y operaciones que modifican o marcan el resultado de alguno de los campos de los registros.
Las reglas se definen todas del mismo modo, aunque algunas dependiendo del objeto y el nivel de jerarquía donde se encuentren tiene restringidas algunos operadores o funciones.

2.4.1.1 Rule:

Existe una gran cantidad de rules, en general todas siguen el mismo procedimiento de creación y ejecución.

En todas las reglas vamos a tener una parte izquierda o LHS y una parte derecha RHS que no será obligatoria siempre y que dependerá del objeto en cuestión.

La parte izquierda o LHS siempre será un atributo que puede ir acompañado o no, de una función de agregación, dependiendo del objeto podremos seleccionar distintas posibilidades en la función de agregación.

La parte derecha o RHS podrá ser un valor constante, otro atributo o simplemente no existir. En los casos de los objetos de Rule Pass Quality, esta parte derecha podrá obviarse y elegir un operador del tipo "are all the same".

Entre la parte izquierda y la parte derecha tendremos un operador, que dependiendo del objeto ofrecerá distintas posibilidades.
Por objeto no crearemos una única comprobación, podremos encadenar varias comprobaciones mediante los operadores lógicos and, or y where.

Si una rule contiene más de una regla podemos hacer que alguna de estas reglas se conviertan en subreglas, a un nivel inferior de la primera.

Las reglas al igual que todos los objetos deben especificar el nivel de creación dentro del universo. La selección del nivel depende del nivel donde se hayan creado los atributos para el nodo. Un atributo especificado al nivel más bajo deberá de tener en TLM las reglas definidas a este nivel.

Algunos atributos estarán delimitados por las propiedades seleccionadas en el SmartSchema, para su comportamiento en TLM. Por ejemplo el tamaño seleccionado hará comprobaciones únicamente de

ese tamaño y los atributos únicamente serán visibles en los distintos objetos si tienen permiso en el SmartSchema para ese objeto.

Los objetos pueden contener más de una regla, moduladas en distintas en función de la necesidad. Si existen más de dos reglas juntas, estas se comportaran como una "y" lógica.
La siguiente tabla muestra los distintos operadores según el tipo de objeto.

Operador	SQL Equivalencia	Rule Type
Equal To	=	All
Not Equal To	!=	All
Greater Than	>	All
Less Than	<	All
Greater Than or Equal To	>=	All
Less Than or Equal To	<=	All
Starts Whit	like 'String%'	All
Ends Whit	like '%String'	All
Contains	like '%string%'	All
Does not Starts Whit	not like 'String%'	All
Does not Ends Whit	not like '%String'	All
Does not Contains	not like '%string%'	All
Plus	+	All
Minus	-	All
Are all the same	son todos iguales	Pass Quality
Are not all the same	no son todos iguales	Pass Quality
Have at least one equal to	Al menos existe uno en RHS (deberá ser un atributo)	Pass Quality
Have none equal to	No existe ninguno en RHS (deberá ser un atributo)	Pass Quality

Las funciones de agregación serán opcionales sobre los RHS, se comentan en la siguiente tabla:

Function	Significado	Tipo de dato
COUNT	El número de registros seleccionados que cumplan la cláusula where	All
SUM	La suma de valores del atributo indicado para los registros que cumplan la cláusula where	Amounts
NET	La suma de valores sea 0 , teniendo en cuenta el signo del lado	Amounts
RANGE	Los registros que cumplan el rango de valores sea igual al especificado	Amounts , date
NETPERCENT	La suma de valores sea 0 , teniendo en cuenta el signo del lado, en tanto %	Amounts
PRINCIPAL	El valor del registro principal dentro de la agrupación	All
AVG	El valor medio de los registros agrupados	Amounts

MIN	El mínimo valor del grupo	Amounts , date
MAX	El máximo valor del grupo	Amounts , date

Las funciones de agregación NET y NETPERCENT se rigen según se haya definido los signos y lados en el SmartSchema. La definición de estos signos cobra aquí su gran importancia. Si definidos en el mismo lado y distinto signo funcionarán, si definidas en distinto lado y no tiene signo también.

2.4.1.2 Eventos:

Los eventos, son objetos donde asignamos un valor a un atributo.
El valor asignado puede ser un valor constante, puede ser otro atributo o puede proceder de la función de agregación del valor del atributo.

Para crear el evento debemos de seleccionar el nivel dentro de la jerarquía de nodos, al igual que todos los objetos de TLM.

Una vez seleccionado el nivel podremos seleccionar el atributo. Para este atributo podemos filtrar en función de una regla, reglas de filtro. De este modo si tenemos una agrupación podremos seleccionar únicamente los registros que cumplan la condición.

Por ultimo seleccionamos el valor, que puede ser constante o puede encontrarse en otro campo. En caso de ser otro atributo también existe la posibilidad de asignar un filtro para el atributo que da el valor.
Estos objetos ofrecen la posibilidad de utilizar operadores y agregaciones.

Operador 1	Operación Permitida	Operador 2	Resultado
Amount	+ , - , X	Amount	Amount
Amount	/	Amount	Amount o Integer
Amount	X , /	Amount o Integer	Amount
Amount o Integer	+ , - , X , /	Amount o Integer	Amount o Integer
Amount o Integer	X , /	Amount	Amount

Tipo De Dato Atributo	Operador Permitido	Funcion de Agregación Permitida
Integer	+ , - , X , /	Max , Count , Min , Net , Sum
Amount	+ , - , X , /	Max , Count , Min , Net , Sum
String	Concatenación , +	
Date	No Permitido	Max , Min

Trabajando con currency:

En TLM existe la posibilidad de utilizar monedas bancarias asociadas a cantidades durante reglas y eventos, para realizar los cálculos.

El establecimiento de USE_BANK_CURRENCY controla:

- Si está en USE_BANK_CURRENCY, y hay una moneda asociada con el conjunto, la moneda bancaria se utiliza para los cálculos.

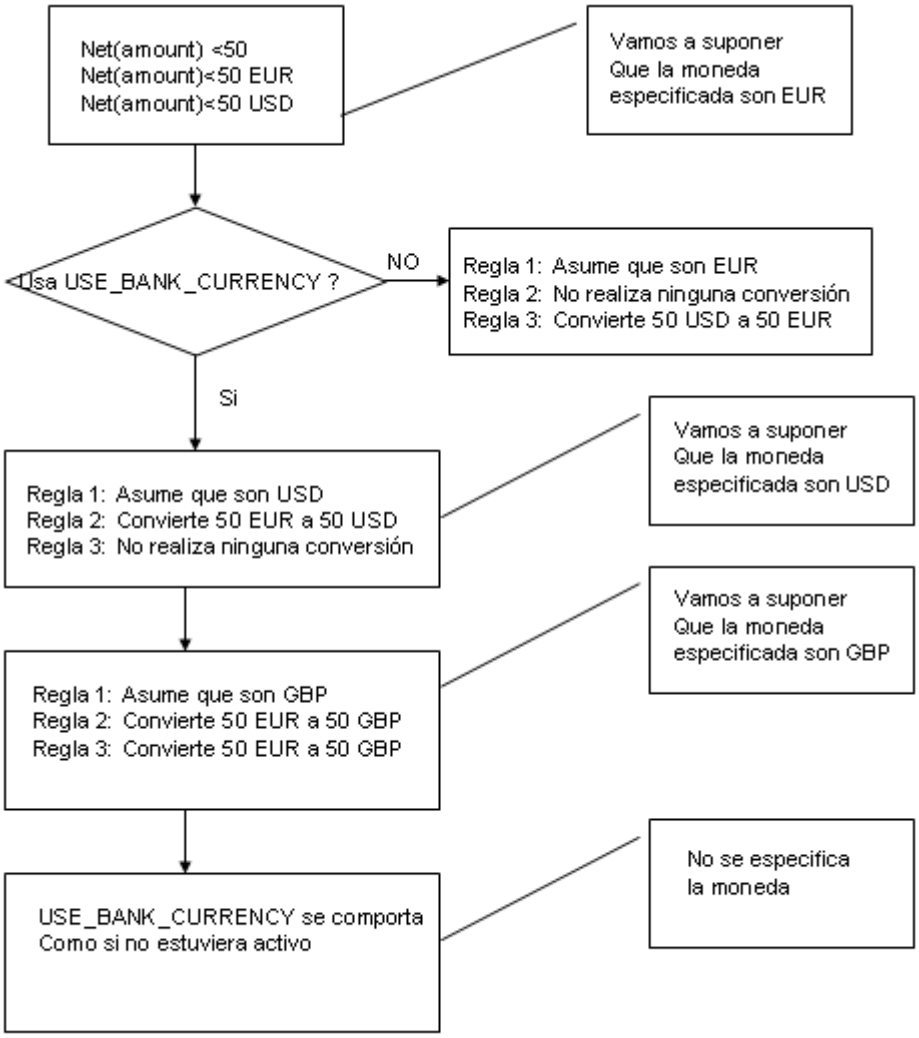
- Si USE_BANK_CURRENCY está apagado, o está encendida pero no hay moneda asociada al conjunto, la base de moneda se utiliza.

Esta configuración afecta a las siguientes áreas:

- Agregaciones en las reglas.
- Population rules (incluida la agregación y la tolerancia de las columnas).
- Workflow Rules
- Pass Quality Rules
- Agrupación de funciones tales como NET, SUM, MOD.

A la hora de crear reglas que usen cantidades específicas de monedas se asociará en la parte RHS la cantidad y la moneda que cumplen la regla.

Podemos observar el diagrama de flujo de interpretación del workflow cunado en una regla asociamos una moneda.



2.4.1.3 Componentes de los pases:

Scope:

Los Scope son rules que sirven para seleccionar la información que queremos recuperar de la base de datos, para realizar las posteriores comprobaciones. En función del nivel de creación dentro del universo de la rules Scope, podremos acceder a una u otra información.

En los pases se pueden asociar varios Scope. Si se seleccionan varios scope funcionan como un "y" lógico entre ellos.

Los Scope no son obligatorios, en caso de no seleccionar ninguno se leerán todos los registros existentes en la base de datos. Por ello no tiene sentido cuando únicamente tengamos un pase. En el caso de haber más de uno, los registros no modificados en ninguno de los eventos, quedaran en memoria para el siguiente pase.

Es importante para el rendimiento de la aplicación crear estas reglas con el mayor número posible de atributos índices.
Los scope tienen limitación de memoria y no son capaces de reprocesar volúmenes exagerados por lo que en ocasiones debemos de restringir al máximo la selección de los registros.

Una de las limitaciones de los scope es el uso de operadores como "Greater than" o "Less than", para el tipo de datos fechas.

Population:

Las Population podemos describirlas como una agrupación de registros según los campos seleccionados.

Para crear una population debemos de seleccionar en primer lugar el nivel de la categoría donde queremos agrupar los atributos, en caso de ser registros de nodos distintos debemos de crear dos entradas en la population, una por cada nodo. Si los atributos están en un nivel superior por categoría basta con incluir una única entrada.

Tras seleccionar las entradas necesarias indicaremos los atributos por los cuales queremos agrupar los registros. Si tenemos mas de una entrada deberán de seleccionar los mismo atributos, o en su defecto el mismo tipo de dato. Debemos intentar evitar los nulos, ya que provocan un error en el sistema de conciliación. Podemos asemejarlos al Group by en SQL.

Hay distintos tipos de population. El tipo de population agregate se utiliza cuando necesitamos agrupar registros pertenecientes a distintos nodos, un caso 1:N. Esto significa que ira sumando las cantidades en los tipos numéricos, y en el caso de la suma de los registros de N sea igual al otro registro del otro nodo, los registros quedarán agrupados. El tipo de population match indica, que la agrupación se esta realizando comprueba si los atributos seleccionados son iguales.

En el caso de que sea un registro perteneciente al tipo amount que es un campo numérico decimal, podremos además, poner una tolerancia, para los casos de registros con un redondeo despreciable. De este modo si las cantidades numéricas están dentro del redondeo, se considera la misma cantidad y los registros quedarán agrupados.

Dentro de las population podemos incluir reglas de esta manera si no se cumplen esa regla no se agruparan los registros.

Pass Quality:

Los Pass Quality son una secuencia de reglas que van realizando una serie de comprobaciones sobre los grupos generados tras la Population. Son una combinación de reglas y Tools. Las Tools son objetos que combinan una serie de eventos y una operación final sobre ellos. La finalidad es realizar una modificación en alguno de los campos de los registros a través de la Tool sobre los registros agrupados y que cumplen la regla de la rule pass quality.

Las Pass qualitys pueden tener varias líneas de reglas que se van ejecutando de modo secuencial. Cuando una agrupación cumple alguna de las reglas y los registros son modificados por un evento, el grupo de registros queda eliminado de memoria temporal y no continúan con la secuencia de reglas restante.

Las pass quality se compone de una o varias rules pass quality y una tool con uno o varios event list.

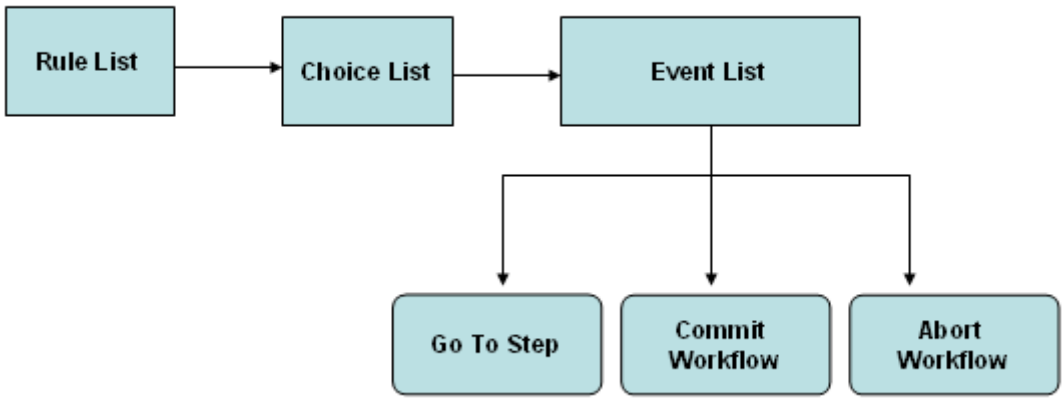
- Rule Pass Quality:

  La Rule pass quality, son condiciones sobre los registros que los campos deben de cumplir.

  Puede haber más de una rule Pass Quality por líneas de la Pass Quality. Si existe más de una serán ejecutadas como si un "y" lógico estuviera entre medias de las dos.

- Tool **:**

  Las Tools son objetos que son composiciones de pasos, estos pasos son opcionales, pero si se presentan deben de estar en el siguiente orden:



Una vez definido el nombre de la Tool, seleccionamos los pasos que queremos que de nuestro flujo de trabajo.
Podemos elegir entre los siguientes:

- Añadir una Rule List:

  Evalúa cuando se cumplen una serie de reglas, estas reglas son definidas para los workflow rules. Una vez tratados indicamos el evento a ejecutar en el caso de que se hayan cumplido con éxito o en fracaso. Posteriormente indicamos la operación a realizar, dentro de estas operaciones encontramos commit, abort workflow y go to step.

- Choice List:

  Permite hacer una serie de preguntas o mensajes al usuario y determinar la acción a seguir. Estas posibles acciones son commit, abort workflow y go to step.

- Event List:

  Una serie de eventos son ejecutados de forma automática sin realizar ninguna comprobación, únicamente las incluidas en los filtros de los eventos.

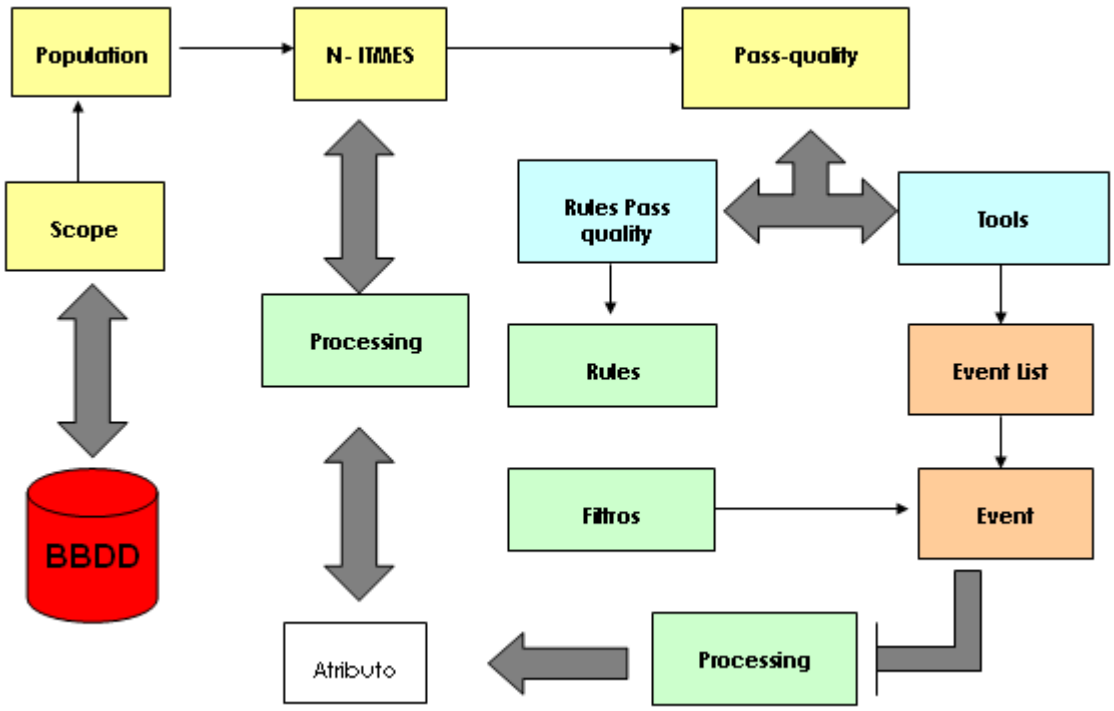
- **Event List:**

  Los Event List constan de varios Eventos, se ejecutarán de modo secuencial según su orden en la lista.

  Los Event List, modificaran los valores de los atributos seleccionados, según la ejecución de los eventos.

  Para crear un Event List hemos de haber creado previamente los eventos que queremos incluir.

Diagrama de flujo de un PASS (pase):



La información se selecciona de la base de datos mediante los scope, pasa a quedarse en memoria temporal. Posteriormente pasa a agruparse mediante la population según los atributos definidos y las posibilidades de agrupación de las population. Una vez agrupados los registros comienzan las comprobaciones en las Pass Quality. Las comprobaciones están formadas por rule pass quality y Tools. Tras evaluarse la pass quality en caso de ser verdadera la comprobación, la Tool indica los pasos que siguen el flujo de trabajo. Una vez asignado este flujo los registros son modificado mediante una lista de eventos. Por último se indica el procesamiento final de los registros (commit, siguiente paso, Abort.)
Una vez tratado el registro o grupo de registros, dejan de estar en memoria temporal.

2.4.2 ¿Cómo muestra la información TLM?

En TLM, se pueden diseñar la interfaz de usuario para satisfacer las necesidades operacionales. TLM permite al diseñador crear ventanas a medida, conocida como cuadros de mando o dashboards, que contiene una variedad de controles, para la presentación, actualización y tratamiento de datos de flujo de trabajo.

Los Dashboards se componen de varios elementos, un diseño de interfaz donde podremos visualizar el resultado, un diseño de vista donde seleccionaremos los campos que queremos mostrar. Y reglas Search, que sirven para seleccionar los registros a mostrar. Esta interfaz gráfica permite interactuar con la búsqueda introduciendo valores por el usuario.

Rule Search:

Son rules con la propiedades de las rule especificadas con el añadido de que algunos de sus campos se mostraran vacíos para que el usuario pueda introducir valores. También existe la posibilidad de que no tenga que meter valores sino seleccionarlo de una lista.

Para crear una vista desde TLM podemos realizarlo de dos modos distintos, desde el universo item (que es el habitual), para ello deberemos crear la vista desde el objeto dashboard. Desde otro universo, para ello crearemos la vista en base de datos y la asociaremos a un nuevo universo desde SmartSchema.

Para crear la vista desde los dashboards tendremos que seleccionar el nivel del nodo desde el que queremos mostrar los datos. Por ultimo asociaremos la vista creada al Dashboards.

Los Dashboards tienen un gran número de opciones gráficas para mostrar los registros, se puede ordenar, agrupar de una manera determinada e incluso se puede interactuar con los registros. Se puede interactuar con ellos gracias a las workflow rule creadas en la tool y a las Choice list que esperan una respuesta del usuario.

Estos dashboards se diseñan en el cliente pesado pero lo normal es que el usuario lo haga mediante el cliente ligero. La versión WEB de TLM es el Webconnect, desde esta herramienta creada con menús superiores sencillos, se puede seleccionar el informe y tratar o actualizar la información.

2.4.3 ¿Qué es Oracle?

Es un manejador de base de datos relacional [DE MIGUEL, 1999] que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información.
Es el conjunto de datos que proporciona la capacidad de almacenar y acude a estos de forma consecuente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas.

Es el mayor y más usado Sistema Manejador de Base de Dato Relacional (RDBMS) [DE MIGUEL, 1999] en el mundo. La Corporación Oracle ofrece este RDBMS como un producto incorporado a la línea de producción. Además incluye cuatro generaciones de desarrollo de aplicación, herramientas de reportes y utilitarios.

Oracle corre en computadoras personales (PC), microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo. Además corre automáticamente en más de 80 arquitecturas de hardware y software distinto sin tener la necesidad de cambiar una sola línea de código. Esto es porque más el 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos.

2.4.4 ¿Qué es PL/SQL?

El lenguaje PL/SQL es la extensión procedural del lenguaje de consulta estructurado SQL en Oracle [WEB Oracle]. El propósito de PL/SQL es combinar el lenguaje de SQL de base de datos y lenguaje de programación mediante procedimientos.

PL/SQL es un lenguaje estructurado en bloques. Cada bloque puede contener a su vez otro sub-bloques y así sucesivamente. Cada parte de un bloque o sub-bloque resuelve, normalmente, un problema o sub-problema. PL/SQL utiliza la táctica del “divide y vencerás” para la resolución de problemas más complejos.

Un bloque o sub-bloque relaciona lógicamente declaraciones y comandos. Las declaraciones son propias de ese bloque y dejan de existir cuando el bloque se finaliza.

2.4.4.1 Bloques PL/SQL

Un bloque PL/SQL esta compuesto por tres partes bien diferenciadas:

- **La parte declarativa:**

  En ella se declaran las variables, constantes, cursores, registros, etc. Que se utilizará en la parte de la ejecución.

- **La parte de ejecución:**

  En ella se especifican todos los comandos que se realizarán para solucionar el problema o sub-problema. También alberga las estructuras de control. Es la única parte que es obligatoria.

- **La parte de excepciones:**

  En ella se tratan los errores que se hayan podido producir en la parte de ejecución.

El orden es lógico puesto que primero se declara con qué se va a trabajar; en la segunda se trabaja con ello y en la tercera se comprueba si ha habido algún error y se obra en consecuencia.

El lenguaje PL/SQL ofrece todas las ventajas de los lenguajes de programación como la encapsulación de datos, definición de objetos, manejo de excepciones y ocultación de la información sensible. Al estar integrado en el núcleo Oracle ofrece también el acceso ala información mediante comandos SQL, portabilidad y seguridad.

En SQL las filas se tratan en grupos y no una por una. PL/SQL aporta las características de un lenguaje de 4ª generación para poder tratar cada fila de una manera independiente.

En el lenguaje PL/SQL se pueden destacar las siguientes ventajas:

- **Soporte al lenguaje SQL:**

  SQL se ha convertido en el lenguaje estándar de las bases de datos por su flexibilidad, potencia y facilidad de uso y aprendizaje. PL/SQL permite utilizar todas las funciones, operadores, pseudo columnas y tipos de datos de SQL.

- **Soporte a la programación orientada a objetos:**

  PL/SQL permite la encapsulación de operaciones y datos con lo que se pueden crear componentes que sean modulares, de fácil mantenimiento y reutilizables.

- **Mejor rendimiento:**

  PL/SQL puede enviar un bloque de comandos al servidor Oracle reduciendo de esta manera el tráfico de red.
  Los procedimientos almacenados son compilados una sola vez y guardados en la base de datos en forma compilada; además una vez que son invocados, estos se guardan en la caché y son compartidos por todos los usuarios. De esta manera PL/SQL reduce el tráfico de red, los requerimientos de memoria y el tiempo de invocación.

- **Total portabilidad:**

  PL/SQL es compatible con cualquier sistema operativo o plataforma donde se este ejecutando un servidor Oracle.

- **Integración con SQL:**

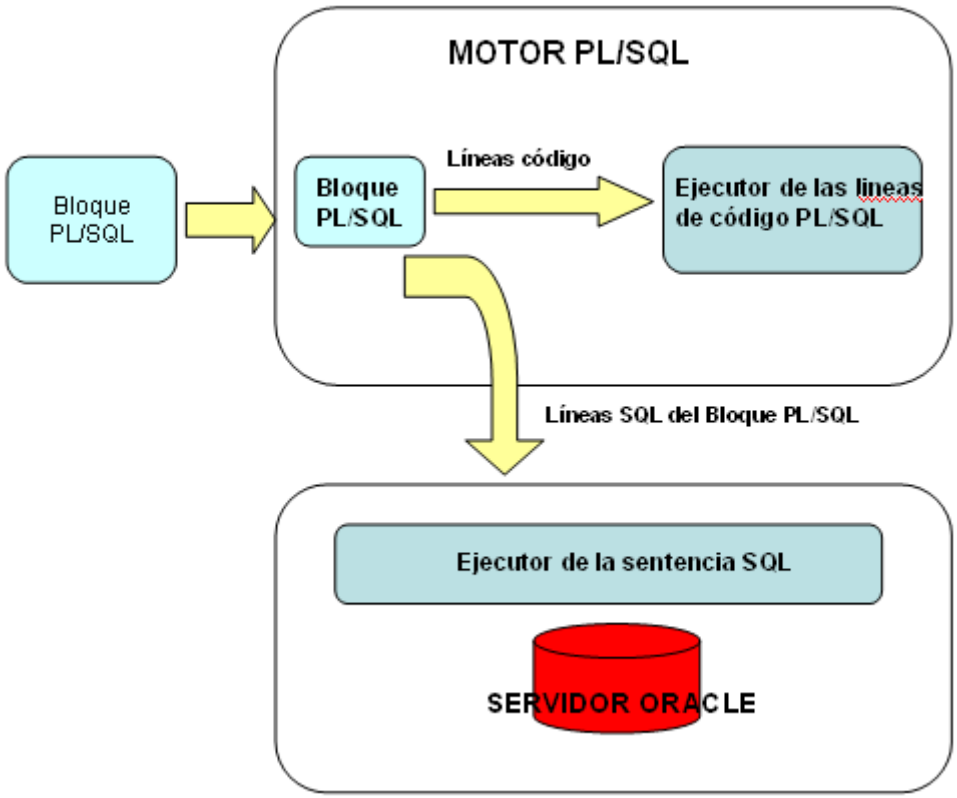
  PL/SQL esta totalmente integrado con SQL. Soporta todos los tipos de datos de SQL así como el valor nulo.
  Los atributos %TYPE y %ROWTYPE permiten definir variables basándose en las columnas de las tablas Oracle. Si variara el tipo de columna, el procedimiento PL/SQL utilizaría la nueva definición sin tener que ser modificado.

- **Gran seguridad:**

  Los procedimientos almacenados permiten dividir la aplicación entre la parte cliente y la parte servidor. De esta manera la parte cliente no puede manipular información sensible. Los disparadores dan la posibilidad de auditoria.
  Permite limitar el acceso a la información de tal manera que sólo sea manipulable a través de procedimientos.

2.4.4.2 Arquitectura del lenguaje PL/SQL:

Cuando se compila un bloque PL/SQL, las líneas de código procedural las ejecuta el motor PL/SQL ya sea dentro de la herramienta en la parte del cliente o del servidor. La sentencia SQL las ejecuta el servidor Oracle y enviará la información obtenida al motor PL/SQL.



- **Bloques anónimos:**

  En tiempo de ejecución, los bloques anónimos, que pueden estar definidos en un programa que se precompile, son enviados al servidor Oracle el cual compilará y ejecutará. Estos bloques no se guardan en el servidor Oracle.

- **Disparadores de Bases de Datos:**

  Los disparadores (Triggers) son procedimientos PL/SQL asociados a eventos que se pueden producir en tablas, vistas o esquemas de la base de datos.

- **Procedimientos Almacenados:**

  Los procedimientos se compilan y guardan dentro del servidor Oracle listo para ser utilizados en cualquier momento por uno o varios usuarios, Los procedimientos se guardan dentro del diccionario y pasan a formar parte del usuario que lo ha creado.

  Los procedimientos pueden formar parte de un paquete (Package) o pueden definirse como independientes (Standalone). Dentro de un paquete puede ser procedimientos públicos (que pueden ser invocados desde cualquier parte) o privados (No son accesibles fuera del paquete y son necesarios para la ejecución del paquete).

  Pueden ser invocados desde cualquier aplicación.

- **Subprogramas:**

  Los subprogramas son bloques PL/SQL con un nombre a los que se les pueden pasar parámetros y pueden ser invocados.

  PL/SQL tiene dos tipos de subprogramas: Los procedimientos que se utilizan para ejecutar una acción y las funciones que siempre retornan un valor.

  Los subprogramas tienen:

  - Las especificaciones: donde se define el tipo de subprograma (procedure o function), el nombre del mismo y los parámetros de entrada y/o salida. Los parámetros son opcionales.
  - Una parte declarativa: variables, constantes, cursores, tipos de datos, subprogramas, etc.
  - Una parte de ejecución: que comienza con BEGIN donde se realizan todas las acciones, sentencias de control y sentencia SQL.
  - Una parte de excepciones, opcional, para el control de errores.

- **Paquetes:**

El paquete es un objeto del esquema que agrupa lógicamente variables, constantes, tipos de datos y subprogramas PL/SQL. Los

Paquetes se dividen en:

- Especificaciones:

Es la zona de declaración de las variables, tipos, constantes, excepciones, cursores y subprogramas disponibles para ser usados.

- Cuerpo:

Zona en la que se implantan el código de los cursores y subprogramas definidos en la especificación, también puede contener otras declaraciones y otros subprogramas que no estén definidos en la especificación.

2.4.5 ¿Qué es Shell Bourne?

Shell es el intérprete de comandos, proporciona comunicación directa entre el usuario y el sistema operativo.

Un Shell Script es Normalmente, usamos el término Shell Script para referirnos a programas escritos para la shell de UNIX/LINUX.
La programación en shell-script es muy útil para resolver tareas repetitivas, típicas de los Administradores. Son ficheros de texto que contienen comandos y son directamente ejecutables por el sistema.

Bourne Shell es un programa informático cuya función consiste en interpretar órdenes. Incorpora características tales como control de procesos, redirección de entrada/salida, listado y lectura de ficheros, protección, comunicaciones y un lenguaje de órdenes para escribir programas por lotes o (scripts o guiones). Fue el intérprete usado en las primeras versiones de Unix.

Mediante los scripts el sistema de conciliaciones financiera automatiza cada uno de los circuitos. Mediante estas shell dirige todos los procesos que se ejecutan y que se encadenan en una conciliación financiera, utilizando este lenguaje, además agrega programas en awk. Hace llamadas a Oracle con bloques de PL/SQL, genera informes y manda correos con los resultados de las conciliaciones.

2.4.6 ¿Qué es AWK?

Dentro de las herramientas del sistema UNIX awk es útil para modificar archivos, buscar y transformar bases de datos, generar informes simples y otras muchas cosas. También se puede utilizar para realizar el tipo de funciones que proporcionan muchas de las otras herramientas del sistema UNIX. Pero puesto que también es un lenguaje de programación, resulta más potente y flexible que cualquiera de ellos.

Awk está especialmente diseñado para trabajar con archivos estructurados y patrones de texto. Dispone de características internas para descomponer líneas de entrada en campos y comparar estos campos con patrones que se especifiquen.
Sin embargo, awk es un lenguaje de programación con estructuras de control, funciones, y variables. Así, si se aprenden órdenes awk adicionales, pueden escribirse programas más complejos.

El nombre de awk se debe a las iniciales de sus diseñadores: Alfred V. Aho, Peter J. Weinberger y Brian W. Kernighan. La versión original de awk fue escrita en 1977 en los Laboratorios de AT&T. En 1985 una nueva

Versión hizo al lenguaje de programación más potente, introduciendo funciones definidas por el usuario, múltiples streams de entrada y evaluación de expresiones regulares.

Awk tiene una gran utilidad en nuestro sistema de conciliación para evitar el uso de disparadores befote insert, pudiendo formatear los campos necesarios, o añadiendo otros nuevos a partir de cierta información.

2.4.7 Planificaciones en Control-M:

Una vez que se han automatizado los circuitos las distintas conciliaciones son planificadas en control-M.

Se trata de una herramienta que permite gestionar la integración de los procesos del negocio, así como la gestión de los servicios del mismo, abstrayéndose de la plataforma tecnológica y de las diferentes aplicaciones que ésta pudiera contener, ayudando a mantener un entorno de flujo de trabajo organizado.

Hace posible la automatización de la carga de trabajo de sistemas, en la ejecución, seguimiento y control de los procesos operativos del centro de cómputos.

Capitulo III

METODOLOGIA DEL DESARROLLO

3.1 Metodología Detallada Para el Desarrollo del Proyecto:

Fase I:

➲ Análisis de la Situación Actual

➲ Reingeniería del Proceso de Conciliación

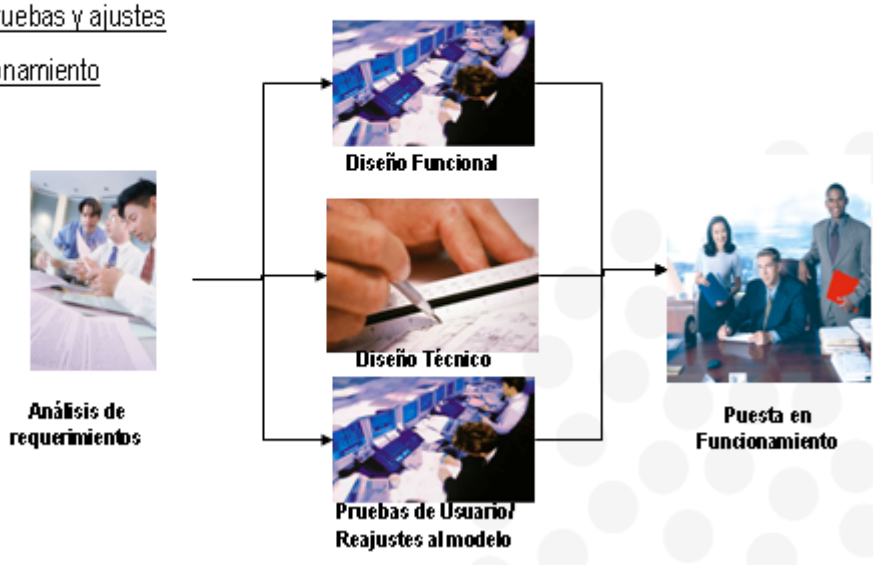
➲ Implantación de la herramienta de Conciliación

➲ Análisis de la Situación Actual: las **necesidades que se ponen de manifiesto** son

- escalabilidad de soluciones
- flexibilidad y adaptación a cualquier tipo de necesidad de conciliación y a cualquier tipo de arquitectura
- control del ciclo de vida completo de las transacciones
- repositorio central de información
- homogeneidad en la definición del proceso y en la toma de decisiones
- organización de la actividad

➲ Para la **Implantación de la herramienta de Conciliación** se realizan las siguientes actividades:

- Análisis de Requerimientos
- Diseño funcional y técnico de la solución
- Validaciones, pruebas y ajustes
- Puesta en funcionamiento



Fase II:

➲ Centralización de la Actividad de Conciliación

porque se trata de una actividad

- **generadora de costes**, no de ingresos
- **imprescindible pero distrae recursos de los objetivos "core"** de las áreas de negocio
- **rutinaria, repetitiva y cada vez más voluminosa**
- **"fácilmente" segregable** (commodity)
- **automatizable** (gracias a los avances en tecnología)
- las necesidades de interacción de múltiples participantes en la resolución de incidencias y excepciones la hacen un **candidato óptimo para la implantación de un workflow**.

Ventajas que aporta la Centralización de la Actividad de Conciliación

Aumento de la Calidad	• Uso de recursos especializados en gestión y operación del servicio y posibilidad de compartir mejores prácticas y know-how
	• Utilización de tecnologías más apropiadas:
	  • Generales para la gestión (cuadros de mando, workflow, etc.)
	  • Específicas de los servicios prestados
	• Los procesos para prestar los servicios acordados son misión-crítica y, por tanto:
	  • La optimización de los procesos y la mejora continua está garantizada (no hay ningún objetivo más prioritario)
	  • Se asegura la adopción de nuevas tendencias tecnológicas (p.ej., STP, gestión de reglas, etc.) con rapidez

Reducción de Costes	• Generación de economías de escala y sinergias

Mejoras Organizativas	• Optimiza la dedicación de recursos
	  •Recursos de las áreas de negocio quedan liberados de actividades "secundarias" para el área lo que contribuye además a una mayor satisfacción y motivación por no tener que realizar labores percibidas como de escaso valor
	  •Recursos especialistas en la actividad centralizada para la operación de cada proceso

Gestión más eficiente y controlada	• Gestión y control operacional en base a cuadros de mando (KPIs) y compromisos de calidad (SLAs). • Mayor facilidad para la implantación de programas, políticas rigurosas de calidad y la toma de decisiones para la mejora continua

3.2 Arquitectura Detallada

Arquitectura detallada aplicada a nuestro sistema:
Como hemos podido observar existen varias aplicaciones, procesos y archivos que dependiendo de su arquitectura consiguen un mejor funcionamiento del sistema.

- La arquitectura del sistema consta de un servidor UNIX, donde tenemos la base de datos, Oracle, donde cargamos los datos de los ficheros.

- Un filesystem en UNIX donde se encuentran todo los archivos de configuración de los procesos, KB, archivos .trg , host, archivos awk y directorios específicos para la recepción de ficheros .

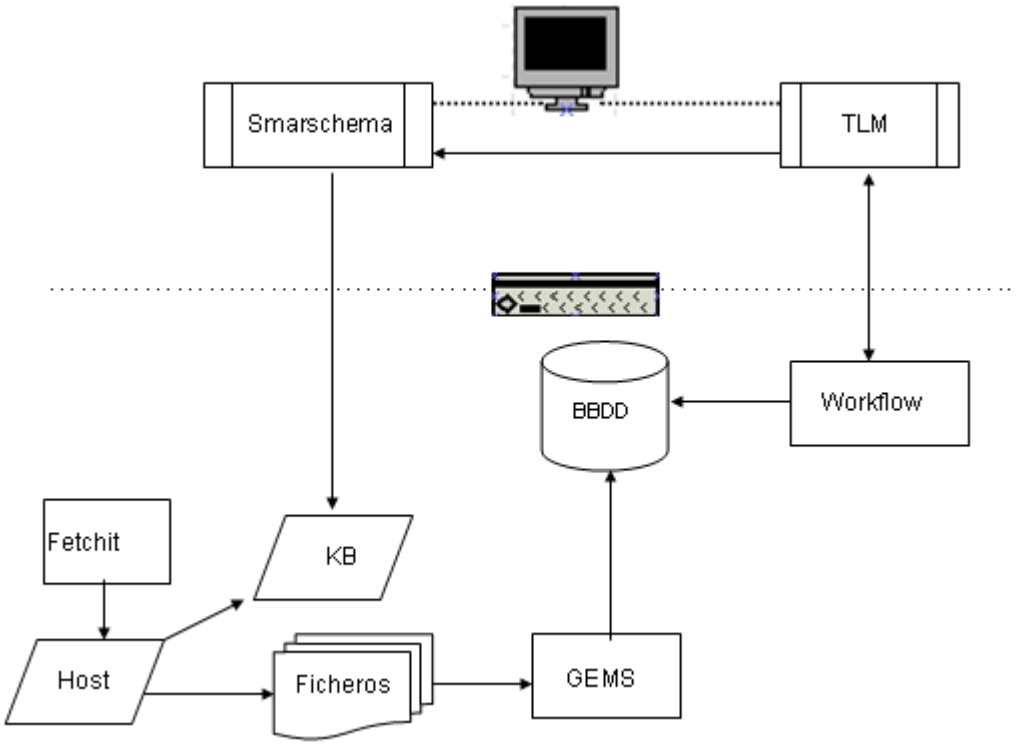
- Varios procesos, entre ello el proceso Fetchit.

- Varios procesos demonios denominados GEMS, que se encargan de realizar la carga.

- Varios procesos demonios denominados workflows que se encargan de ejecutar las reglas de cruce en TLM.

- Dos aplicaciones, SmarchtSchema y TLM.

- El cliente ligero, de uso del cliente, mediante vía WEB.



3.3 Explotación del sistema

Dentro del sistema de conciliación hemos visto que participan una gran cantidad de procesos que interactúan con la base de datos, junto con las aplicaciones.

Una vez configurado el SmartSchema, creado el KB, parametrizadas las reglas de cruce en TLM. Habrá que configurar otros archivos para la ejecución del proceso.

3.3.1 Ficheros .TRG:

Son archivos que indican el patrón del nombre del fichero.

3.3.2 El HOST:

El archivo Host sirve para indicar al proceso fetchit donde debe de buscar la correspondencia dentro de nuestro filesystem de los argumentos que necesita.

Las líneas de Host estarán identificadas con un nombre el cual se ejecutara el proceso fetchit (. /fetchit nombre _ asignado). La ruta de los ficheros de entrada, los archivos .trg , la ruta donde depositara los ficheros , la invocación al proceso de carga GEMS , el nombre del FEED correspondiente al KB (esquema de carga) y la base de datos.

En algunas ocasiones se invocaran programas awk que nos permitirán reformatear el fichero antes de la carga.

Una vez encontrados los ficheros, el fetchit recoge los ficheros de la ruta especificada en el host, inserta una entrada en la tabla vque, y espera que el proceso demonio GEMS actúe.

3.3.3 Proceso GEMS:

El proceso GEMS, es un proceso demonio que esta pendiente de las entradas en la tabla vque para realizar la cargar de los ficheros en la base de datos. Se encarga de cargar en la base de datos los ficheros según lo especificado en el KB.

Suele haber más de un GEMS en la carga, para cargar en paralelo más de un fichero.

El problema surge cuando se quiere cargar más de un fichero el mismo nodo. Cuando más de un proceso GEMS acude al mismo nodo en la base de datos se producen bloqueos. Estos bloqueos suponen la

perdida de tiempo en las comprobaciones que desbloquean en la base de datos.
Una vez comprobado que la carga ha sido correcta, se inserta en la base de datos en la tabla workfow_queue, una entrada con los argumentos necesarios para que el demonio workflow comience su trabajo.

3.3.4 Proceso Workflow

El proceso workflow comienza a ejecutar todas las reglas de conciliación parametrizadas e integradas en la Initiation. El workflow se asocia a una cuenta, para ejecutar únicamente la información que nos interese.
Existen más de dos workflows en paralelo, y únicamente pueden cruzar información que se encuentre en la misma cuenta.

3.3.5 Programas de automatización de procesos:

Para automatizar los procesos de carga, cruce y reporte de resultados, desarrollamos programas Shell en la máquina AIX. Normalmente tendremos un programa por cada conciliación, teniendo en cuenta si se trata una conciliación de ficheros “fijos” o bien de ficheros individuales como los mensajes Swift.

Un script, en general, podrá englobar las siguientes tareas:

1. Comprobación de la existencia de los ficheros adecuados para la completa carga de ficheros
2. Comprobación de fechas en ficheros, a través del nombre de los mismos.
3. Posible chequeo de punto de partida correcto (por ejemplo, si no existen registros pendientes de días anteriores) ó marcado de registros anteriores a un estado concreto.
4. Carga a través del fetchit correspondiente (entrada presente en fichero Host de TLM y ficheros trg con los patrones de nombres).
5. Comprobación de la carga completa: especialmente importante en carga de varios ficheros de un mismo FEED: hay que esperar a que todos los registros terminen con una cuenta positiva válida en BANK.corr_acc_no.
6. Lanzamiento del cruce: mediante inserción en tabla WORKFLOW_QUEUE
7. Comprobación de finalización.
8. En algunos casos, envío del resultado en fichero plano de columnas con separador.

Para estas tareas el script realizará llamadas a la base de datos.

Los scripts estarán programados en el planificador Ctrl-M, según dos posibles esquemas:

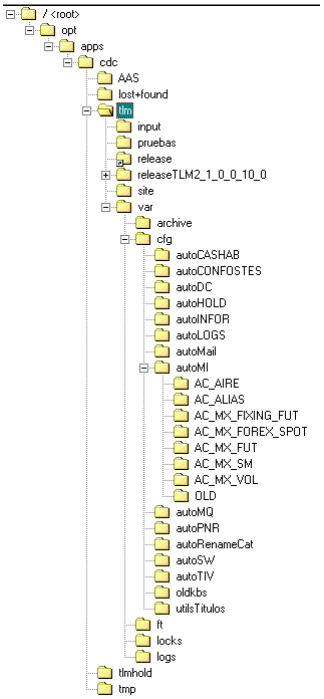
- Planificación a horas concretas:

  El script, en caso de ficheros fijos, tiene un tiempo personalizado de espera de los ficheros desde la hora de lanzamiento.

- Planificación condicionada a la ejecución en Ctrl-M de otros procesos:

  Así podemos condicionar nuestro proceso al correcto ftp previo de ficheros que también se encuentren planificados en Ctrl-M.

Tendremos una estructura de directorios en la ruta $WINACFG de la máquina AIX, con una nomenclatura auto___

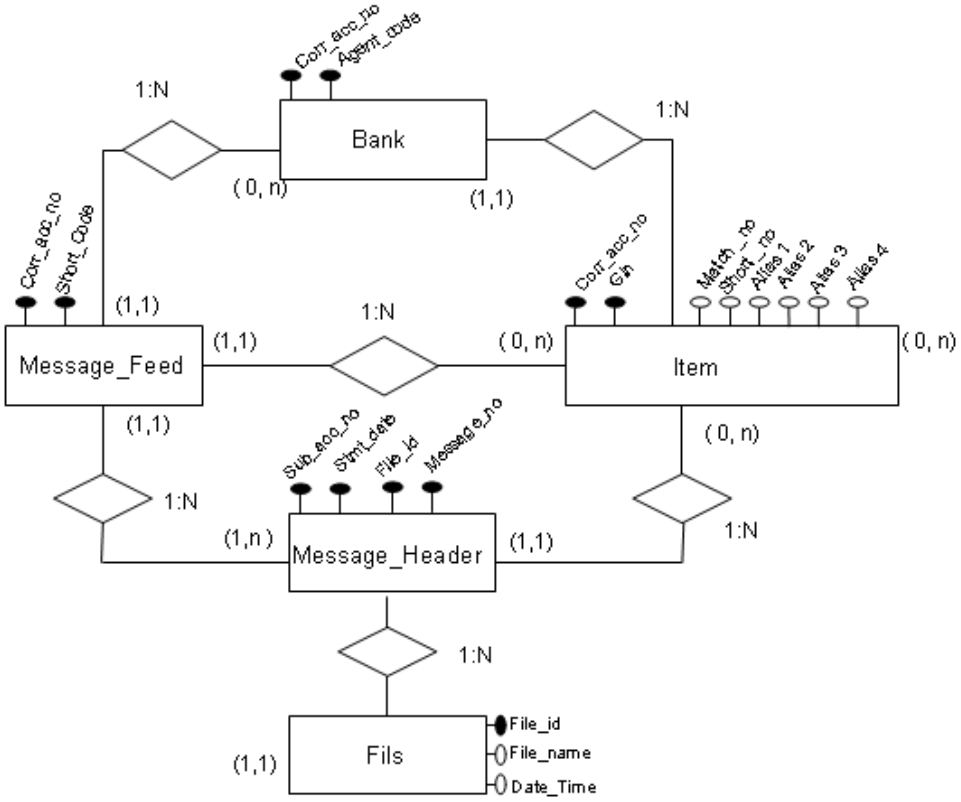


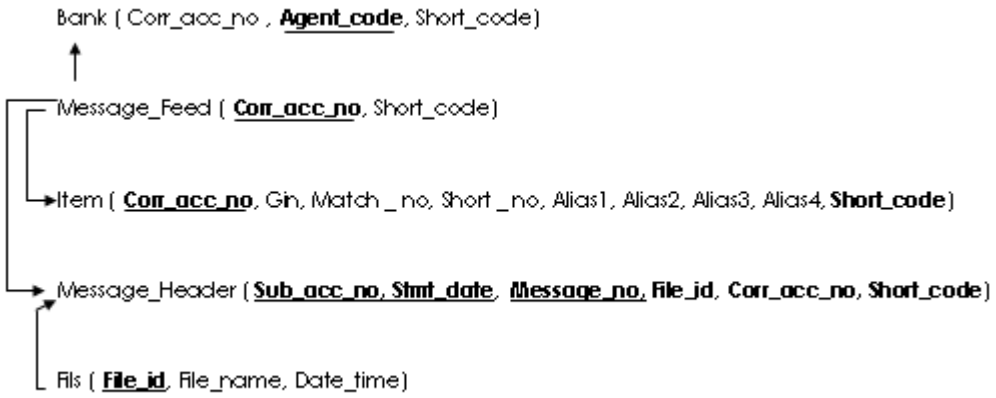
3.4 Modelo Entidad Relación

Una base de datos relacional [DE MIGUEL, 1999] simplifica y definida como un modelo de información es estrictamente visualizable por los usuarios mediante tablas. Una tabla esta compuesta por una matriz bidimensional de filas y columnas. En cualquier ocasión la información es cambiada en una base de datos relacional, cualquier información es el resultado de una consulta presentad por el usuario en el formato filas/columnas. Un modelo relacional posee tres grandes aspectos:

- Estructuras: Definición de objetos que contengan datos accesibles a los usuarios.
- Operaciones: Definir acciones que manipulen datos u objetos.
- Reglas: Leyes para gobernar la información, como y quien manipular.

3.4.1 Modelo de datos lógico asociado a la configuración SmartSchema-TLM:





3.4.1.1 Asociaciones importantes entre las tablas 3.4.1:

- Tabla Bank y item

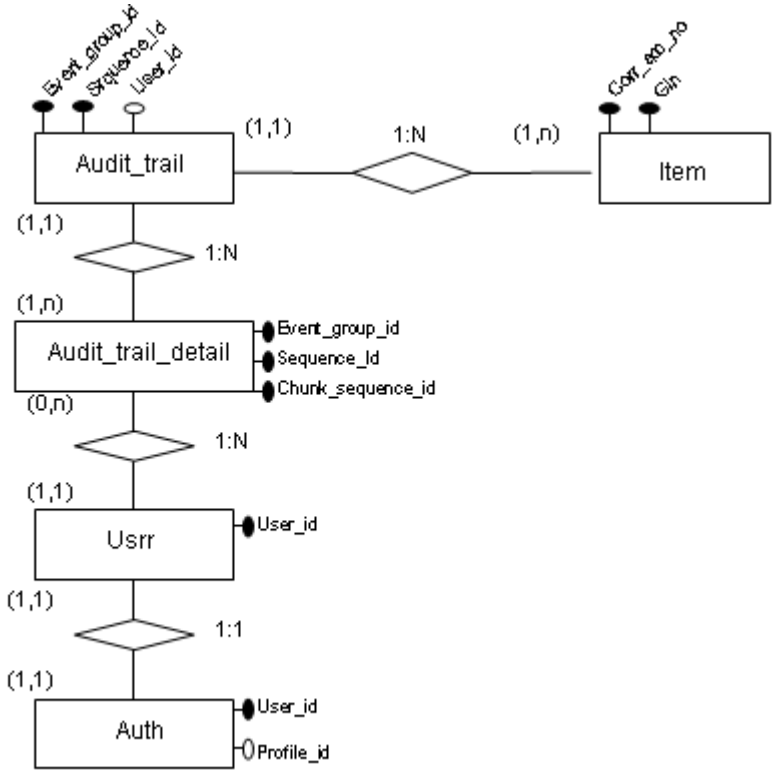
  Para comprobar que los registros han cargado correctamente se hace un JOIN con los números de cuentas entre las tablas item y bank.
  Es importante realizar esta comprobación puesto que los registros pueden haberse cargado en item con cuenta negativa. Esto significa que no se esta relacionando la tabla bank con la tabla item y TLM no localizará los registros en la base de datos.

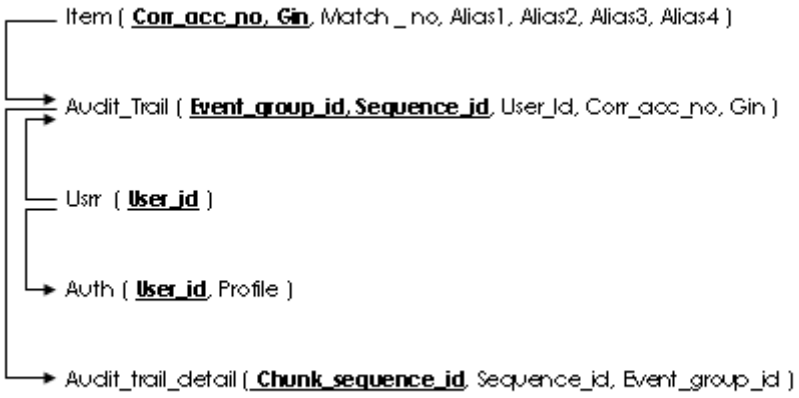
- Tabla Fils, Message_Header y Item:

  Cada vez que se realiza una carga, el proceso GEMS crea una entrada en esta tabla, donde se recoge el nombre del fichero cargado en el atributo file_name y el file_id que es un valor numérico que se asigna para identificar el fichero cargado.

  El file_ id tiene un uso bastante importante ya que asocia las tablas message_header y item. El campo file_id es clave ajena en message_header, que a su vez es clave ajena mediante el campos message_no =alias_4 de la tabla item. Lo que posibilita poder distinguir los registros pertenecientes a una carga en concreto.
  Tiene una gran utilidad puesto que de esta forma se pueden realizar borrados lógicos de los datos, basta con poner el alias_4 a negativo de los registros que no queramos mostrar en item.

3.4.2 Modelo de datos lógico asociado a los eventos (update de registros en item):





3.4.2.1 Asociaciones Importantes entre las tablas del modelo:

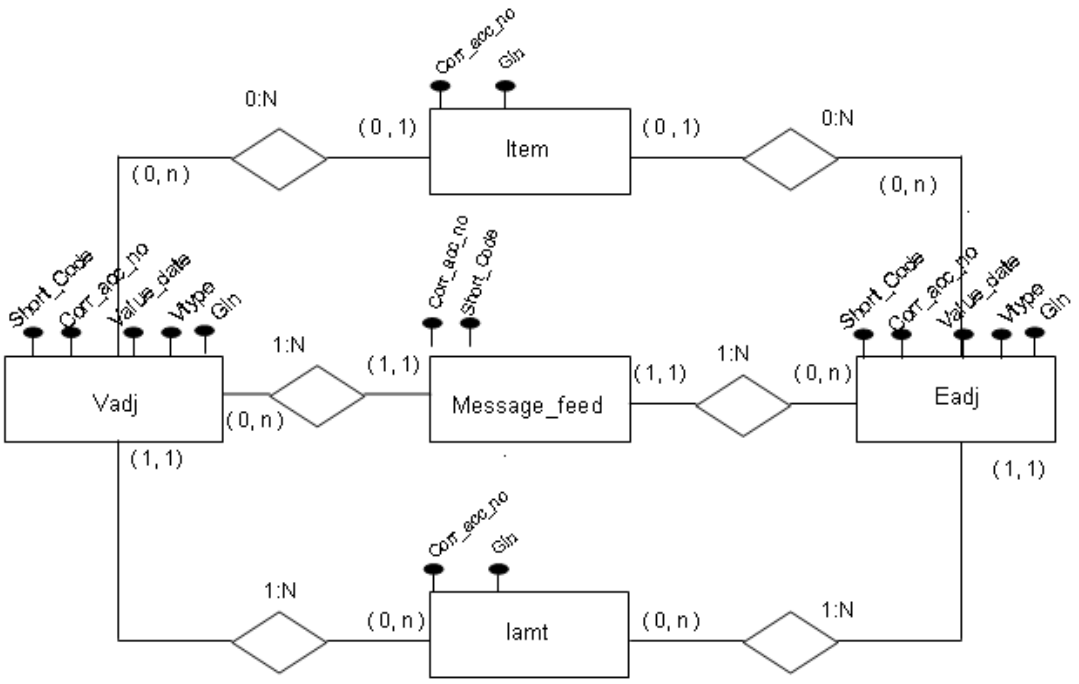
La auditoria del sistema es una forma de comprobar los sucesivos eventos que han afectado a una partida concreta. Para ello, el sistema almacena en la tabla AUDIT_TRAIL qué eventos han actuado sobre atributos concretos de cada registro en ITEM. Esta consulta, accesible para los usuarios a través de la opción Launch Audit Trail en WebConnect, puede usarse para reconstruir estados.

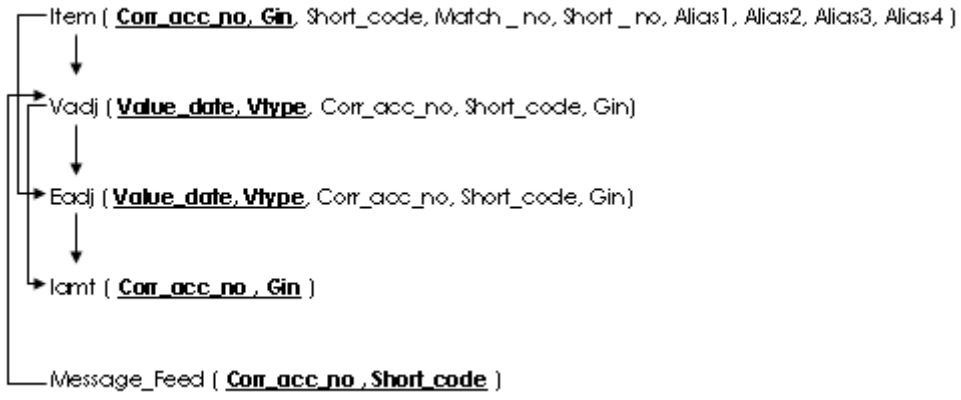
La tabla AUDIT TRAIL puede darnos información de una partida como los eventos que están modificando un valor en concreto y el valor antiguo del atributo.

Para ello la tabla AUDIT TRAIL se relaciona con la tabla item mediante las claves ajena corr_acc_no y gin. De esta forma podemos saber mediante el campo Event_id, que a su vez se relaciona con la tabla EVENT_DETAIL, que eventos están actuando sobre un grupo de registros y que valores tenían y están tomando.

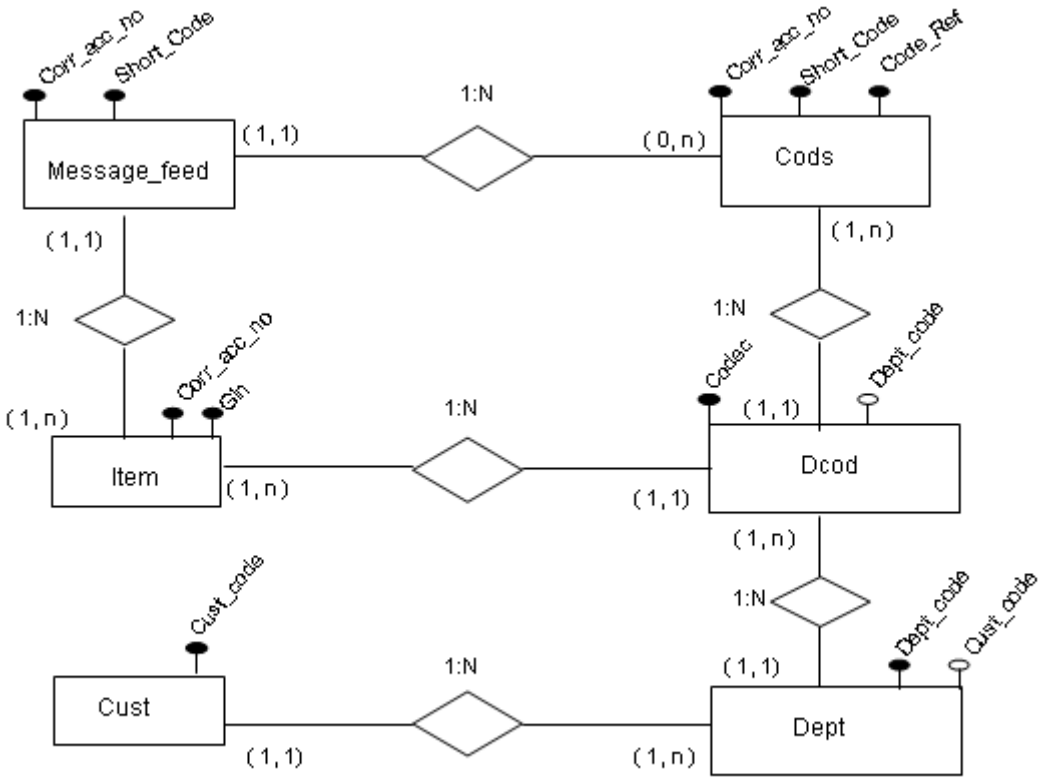
Además puesto que el propio usuario puede modificar los resultados se tiene un control si la modificación ha sido automática o realizada por algún usuario, mediante el campo User_id y la tabla User.

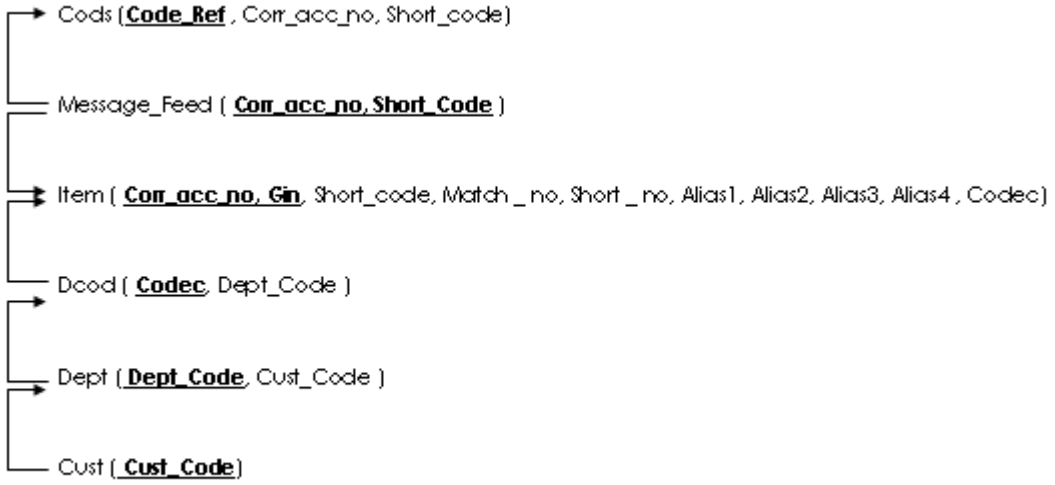
3.4.3 Modelo de datos lógico asociado a la creación de reglas:





3.4.4 Modelo de datos lógico asociado a la creación de cuentas y su asociación con la cuenta de la sucursal correspondiente:





3.4.4.1 Asociación importantes entre las tablas 3.4.4 :

El siguiente diagrama de ER muestra la relación entre la creación de cuentas a través de TLM con los datos de las cuentas correspondientes a un banco.

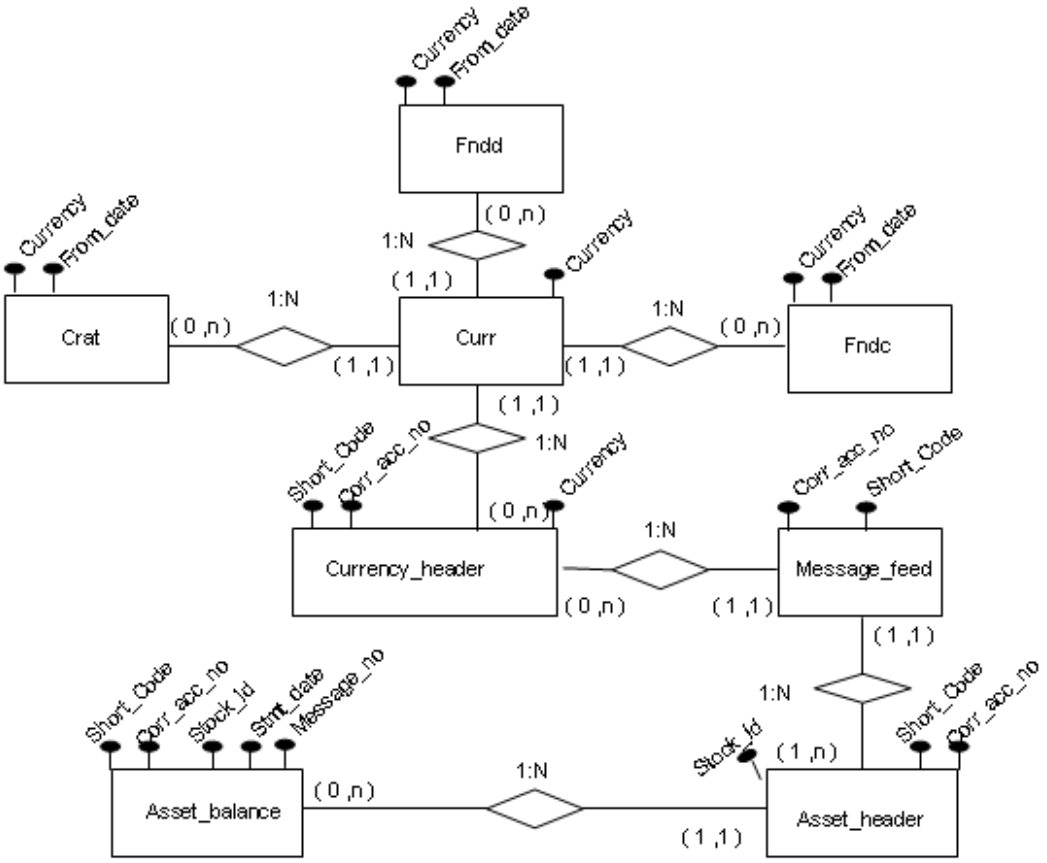
Por ejemplo, la creación de la cuenta 4B, para el departamento de BANESTO, asociado al banco Santander. Se asociaran datos como correo del cliente, teléfono, fax, etc.

La asociación con los registros de item se realizará a través, de la siguiente relación:

Cods.corr_acc_no=item.corr_acc_no=Message_feed.corr_acc_no

No son tablas de consultas de datos masivos por lo que no contienen índices importantes, salvo los correspondientes a las tablas de datos auxiliares con item.

3.4.5 Modelo de datos lógico asociado a los currency:



3.5 Diseño Físico

3.5.1 Índices

Los índices son esquemas que hacen que Oracle acelere las operaciones de consulta y ordenación sobre los campos a los que el índice hace referencia.

Se almacenan aparte de la tabla a la que hace referencia, lo que permite crearles y borrarles en cualquier momento.

Lo que realizan es una lista ordenada por la que Oracle puede acceder para facilitar la búsqueda de los datos. Cada vez que se añade un nuevo registro, los índices involucrados se actualizan a fin de que su información esté al día. De ahí que cuantos más índices haya, más le cueste a Oracle añadir registros, pero más rápidas se realizan las instrucciones de consulta.

La mayoría de los índices se crean de manera implícita, como consecuencia de las restricciones PRIMARY KEY (que obliga a crear un índice único sobre los campos clave), UNIQUE (crea también un índice único) y FOREIGN KEY (crea un índice con posibilidad de repetir valores, índice con duplicados). Estos son índices obligatorios, por los que les crea el propio Oracle.

Índices en TLM:

Los índices más importantes en TLM, se encuentran en la tabla item, que es la tabla donde se crean los registros con los datos de las extracciones y sobre la que se hacen el mayor número de consultas.

Índices sobre Item:

POR DEFECTO, EN LA APLICACION

INDEX_NAME	COLUMN_POSITION	COLUMN_NAME
ITEM_IND_KEY	1	CORR_ACC_NO
	2	GIN
ITEMIXA	1	CORR_ACC_NO
	2	FLAG_2
	3	AMOUNT
	4	VALUE_DATE
	5	GIN
ITEMIXB	1	CORR_ACC_NO
	2	MATCH_NO
	3	GIN
ITEMIXC	1	CORR_ACC_NO
	2	SHORT_NO
	3	SENT_DATE
	4	AMOUNT
	5	STOCK_ID
	6	GIN
ITEMIXD	1	ENTRY_DATE
	2	CORR_ACC_NO
	3	FLAG_2
	4	STOCK_ID
	5	GIN
ITEMIXE	1	VALUE_DATE
	2	CORR_ACC_NO
	3	FLAG_2
	4	STOCK_ID
	5	GIN
ITEMIXF	1	STOCK_ID
	2	CORR_ACC_NO
	3	FLAG_2
	4	AMOUNT
	5	GIN
ITEMIXG	1	ALIAS_4

A MEDIDA [Eliminado]

INDEX_NAME	COLUMN_POSITION	COLUMN_NAME
ITEMCUSTIXA	1	CORR_ACC_NO
	2	STRING_15
	3	LS_TYPE
	4	DATE_3
	5	FLAG_2
	6	AMOUNT
	7	GIN
ITEMCUSTIXB	1	LS_TYPE
	2	FLAG_2
	3	CORR_ACC_NO
	4	STRING_15
ITEMCUSTIXC	1	CORR_ACC_NO
	2	FLAG_2
	3	LS_TYPE
	4	STRING_10
	5	STRING_30
	6	SFIELD_8
	7	FLAG_7
	8	SFIELD_9
	9	STRING_17
	10	STRING_12
	11	STRING_14
	12	FLAG_8
	13	GIN
ITEMCUSTIXD	1	CORR_ACC_NO
	2	LS_TYPE
	3	SFIELD_8
	4	FLAG_7
	5	SFIELD_9
	6	STRING_17
	7	STRING_12
	8	STRING_14
	9	FLAG_8
ITEMCUSTIXE Eliminado 2009ENE28	CORR_ACC_NO	
	2	LS_TYPE
	3	FLAG_2
	4	STRING_10
	5	STRING_30
	6	FLAG_42
ITEMCUSTIXF	1	CORR_ACC_NO
	2	LS_TYPE
	3	FLAG_50
	4	FLAG_2
	5	FLAG_42
ITEMCUSTIXG	1	CORR_ACC_NO
	2	LS_TYPE
	3	FLAG_43
	4	FLAG_2
	5	FLAG_42
ITEMCUSTIXH	1	CORR_ACC_NO
	2	LS_TYPE
	3	FLAG_42
	4	FLAG_2
ITEMCUSTIXI	1	CORR_ACC_NO
	2	LS_TYPE
	3	SENT_DATE
	4	FLAG_2
	5	FLAG_42

- Ls_type:

  Identifica el nodo donde vamos a cargar los datos. Se asigna en la tabla Message_Feed cuando creamos el nodo en el SmartSchema de forma automática. y es relacionado con item en el propio SmartSchema.

- Amount:

  Es un índice creado sobre un dato de tipo decimal. En caso de tener que asignar cantidades, es importante usarlo. Los informes de consultas de los usuarios realizan búsquedas de valores introducidos por el usuario, lo que supone una mayor rapidez en caso de introducir la cantidad.

- Entry_date:

  En ese campo se suele guardar la fecha del sistema en el momento de la carga, tiene una gran validez, para restringir los registros únicamente cargados en el día.

- Flag_2:

  Es un dato de tipo entero, que suele usarse para indicar el resultado del cruce. Es importante a la hora de realizar varios pases, ya que en muchos scope se pretende restringir de la búsqueda los registros ya cruzados.

- Corr_acc_no:

  Es el número de cuenta donde se han cargado los registros, es el campo más importante, ya que diferencia los distintos productos cargados en la base de datos. Los asocia a otras muchas tablas, y en caso de realizarse una carga errónea se habrán cargado cargados con un número negativo.

  Utilizar el corr_acc_no en las reglas de TLM no es posible puesto que este valor cambia en función del entorno.

  Las cuentas tienen que ser creadas a mano desde TLM, se insertan directamente en BANK que mediante la relación con item, la asocia como clave ajena.

- Gin:

El gin es un valor que asigna el proceso GEMS de modo automático una vez insertado en la base de datos, es único junto con el corr_acc_no.

Índices sobre Bank:

- Agent_code:

  Es el nombre de la categoría creada en TLM. Puede haber varios corr_acc_no asociados a un Agent_code.

- Corr_acc_no:

Son los números de cuentas asociadas a las categorías.

- Local_acc_no:

  Son los nombres literales asociados a las cuentas, son importantes puesto que las cuentas no siempre son el mismo número dependiendo del entorno de la base de datos. Sin embargo, el Local_acc_no siempre es el mismo, lo que posibilita su búsqueda generalizada mediante un like.

Índices sobre Fils:

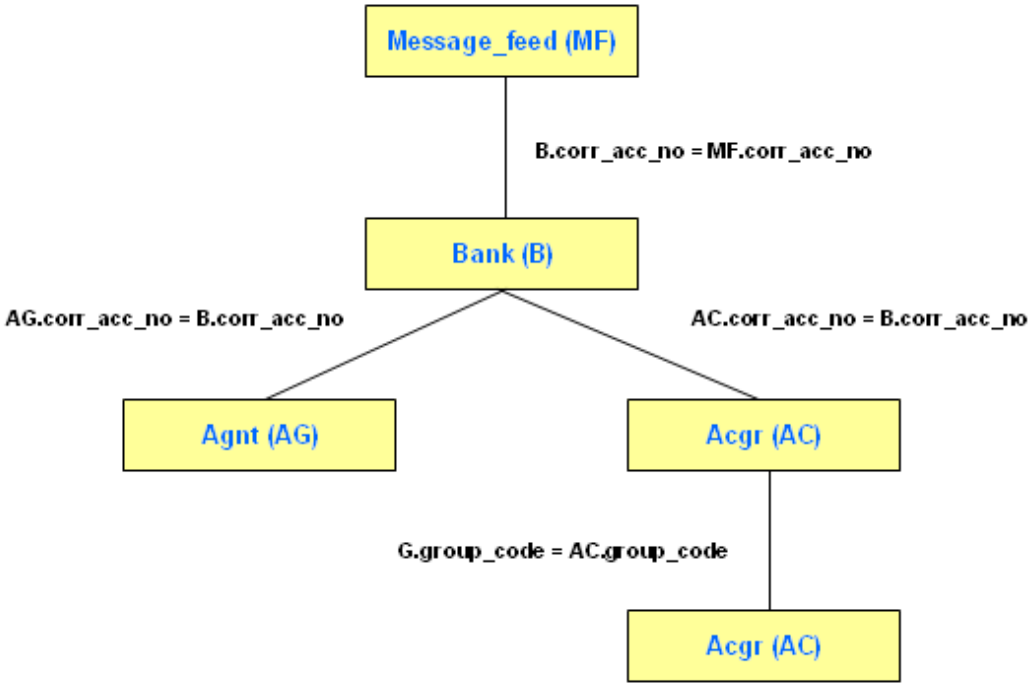
- El campo file_id:

Es el número que se le asocia a un fichero, que se identifica por su nombre completo con directorio donde se deposita en la carga fecha y hora.

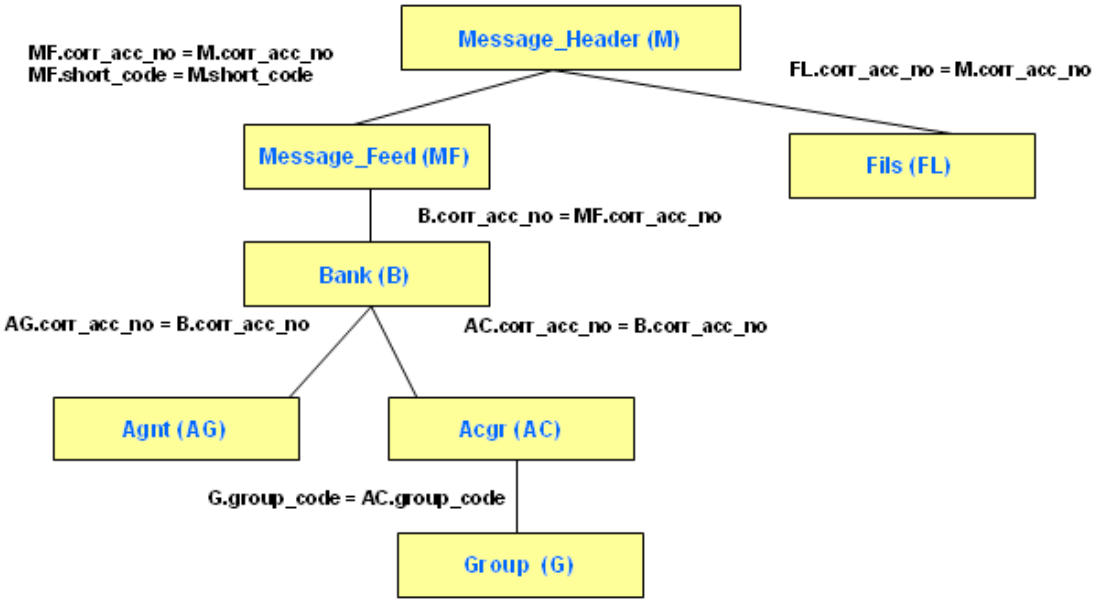
3.5.2 Entidades y Joins de los universos TLM:

A continuación se muestran las entidades más importantes y la relaciones entre estas, de los tres universos más importantes en SmartSchema y TLM, universo Message feed, Universo Message Type y Universo Item.

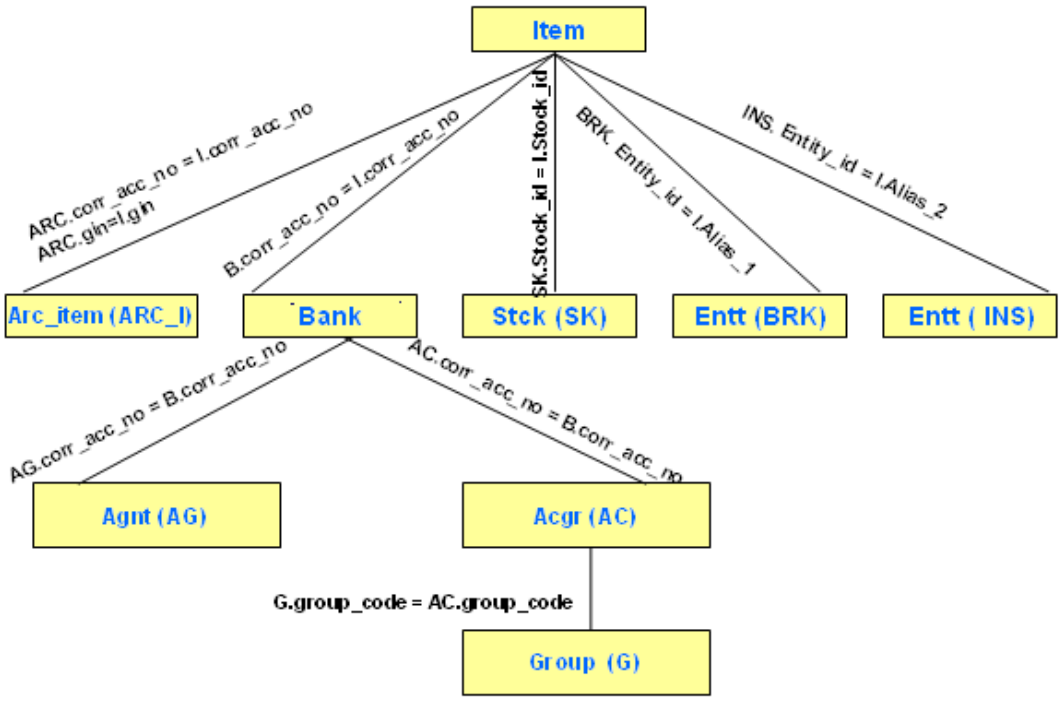
3.5.2.1 Universo Message Feed:

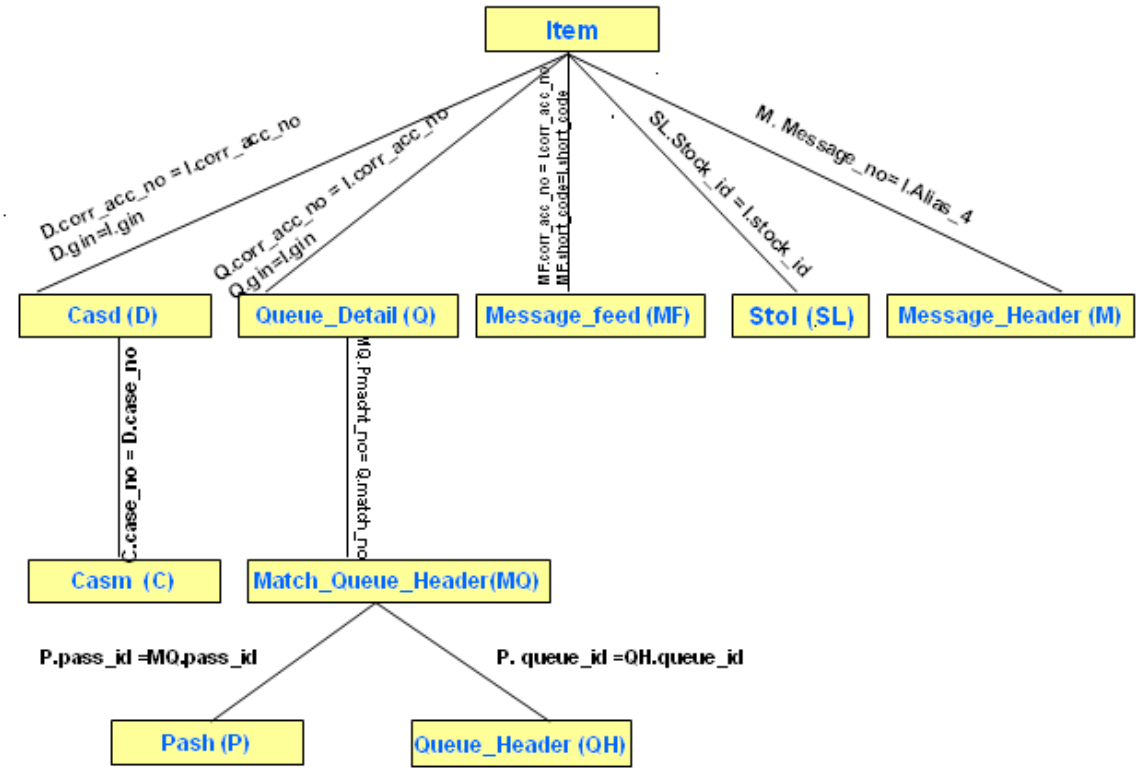


3.5.2.2 Universo Message Header:



3.5.2.3 Universo Item:



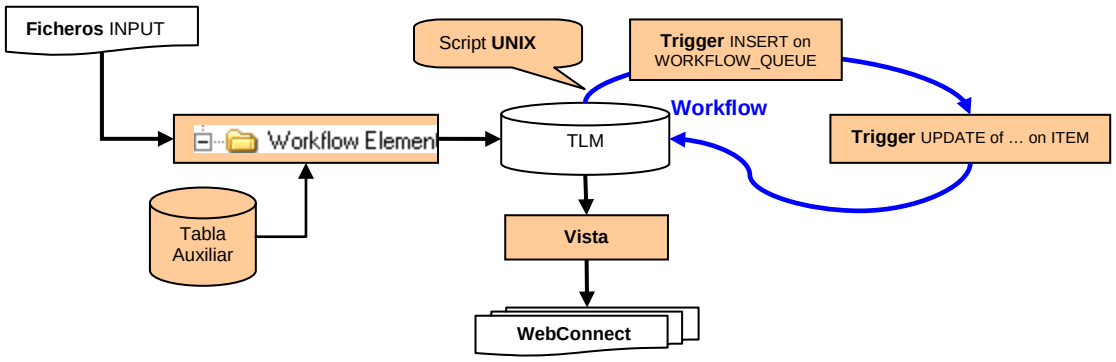

Item
D.corr_acc_no = I.corr_acc_no
D.gin=I.gin
Q.corr_acc_no = I.corr_acc_no
Q.gin=I.gin
MF.corr_acc_no = I.corr_acc_no
MF.short_code=I.short_code
SL.Stock_id =I.stock_id
M. Message_no= I.Alias_4
Casd (D)
Queue_Detail (Q)
Message_feed (MF)
Stol (SL)
Message_Header (M)
C.case_no = D.case_no
MQ.Pmatch_no= Q.match_no
Casm (C)
Match_Queue_Header(MQ)
P.pass_id =MQ.pass_id
P. queue_id =QH.queue_id
Pash (P)
Queue_Header (QH)

3.6 Elementos de Programación:

Elementos auxiliares en TLM:

Además de los procesos indicados, el sistema de conciliaciones financiera tiene otra serie de procesos externos que indicamos a continuación. Mucho de ellos son procesos basados en bloques PL/SQL u otros objetos en base de datos como tablas auxiliares, vistas, etc. También utilizamos las posibilidades de las shell a través de la combinación de comandos en shell script.

A continuación se explicará la funcionalidad de los distintos elementos auxiliares que intervienen en el sistema de conciliación financiera.



TRIGGER T_MI_INSERT_ITEM (Ordenación de precios)	
Ejecución	Siempre que insertamos información y se cumple que ese tipo de cruce necesita ordenación de precios para el lado indicado. ***:new.flag_9=15 and :new.ls_type=10087***
Entrada	El atributo value puede tomar distintos valores que indica el tipo de precio.
Salida	El disparador coloca el precio en el campo correspondiente según indique el valor del campo value. Por ejemplo si value tiene valor bid, el disparador insertara en este campo el valor del precio. ***If upper(:new.STRING_23) like 'BID%' then :new.amount_12:=:new.amount_8;***

Ejemplo:

Entrada

FECHA	FEED	LADO	INDEX	PILLAR	ORDINATE	BID	ASK
01/07/2009	MI_AC_MAD	L	DI_FUT_1Y_OFF	OCT09	10.000000000	23.8875	23.8875

FECHA	FEED	LADO	INDEX	PILLAR	ORDINATE	VALUE	PRICE
01/07/2009	MI_MX_MAD	S	DI_FUT_JAN12OFF	JUL10	75,000000	bid	22,0500
01/07/2009	MI_MX_MAD	S	DI_FUT_JAN12OFF	JUL10	75,000000	ask	22,0500

Salida:

FECHA	FEED	LADO	INDEX	PILLAR	ORDINATE	BID	ASK
01/07/2009	MI_AC_MAD	L	DI_FUT_JAN12OFF	JUL10	10.00000000	23.8875	23.8875

FECHA	FEED	LADO	INDEX	PILLAR	ORDINATE	BID	ASK
01/07/2009	MI_MX_MAD	S	DI_FUT_JAN12OFF	JUL10	10.000000	23.8875	0
01/07/2009	MI_MX_MAD	S	DI_FUT_JAN12OFF	JUL10	10.000000	0	23.8875

TRIGGER T_MI_INSERT_ITEM (inserción de entry_date)	
Ejecución	Siempre que insertamos información
Entrada	El campo entry_date de tipo fecha entra con valor por defecto '01/01/1900', al ser un índice nos interesa darle la fecha de carga del fichero, de esta forma conseguiremos mayor rapidez y limitaremos las búsquedas en las conciliaciones a únicamente los registros cargados hoy.
Salida	El disparador inserta la fecha de carga de los registros en la base de datos. `:new.entry_date := trunc(sysdate);`

TRIGGER T_MI_INSERT_ITEM (marcar registros a desestimados)	
Ejecución	Siempre que insertamos información para un nodo y un tipo de cruce determinado.
Entrada	Se trata de marcar el campo estado a desestimado para que no entre en las comprobación posteriores y optimizar recursos. El falg_8 entra por defecto a 0, y se marca a 1 junto con el último registro.

Salida	El disparador marca el estado a 11 y el último registro a 1, y asigna al campo número de cruce el valor del máximo gin.
	
	`:new.flag_8 := 11 ; --Desestimado`
	`:new.flag_2 := 1 ; -- Tratado`
	`:new.match_no := :new.gin;`

TRIGGER T_UPDATE_FLAG_8 (Disparador para evitar entradas en la tabla audit. Trail)	
Ejecución	Siempre que modifiquemos los campos estado y último registró para los nodos indicados. El estado sea mayor de 0 o distinto de 4. *BEFORE UPDATE OF flag_8 ON item* *when* *(new.ls_type in (10072, 10073, 10087,10141)* *and new.flag_8>0 and new.flag_8<>4)*
Entrada	El estado indica el resultado de la conciliación, toma un valor determinado dependiendo de si el o los registros han conciliado correctamente o al contrario. Tanto el atributo estado como el atributo ultimo registro tienen valor 0 en la entrada de base de datos.
Salida	Durante la conciliación el campo estado será modificado y el flag_2 será marcado a 1 en el disparador evitando una entrada en la tabla Audit. Trail. *:new.flag_2 := 1 + :old.flag_2 ;*

TRIGGER T_UPDATE_FLAG_8 (Disparador para evitar entradas en la tabla audit. Trail)	
Ejecución	Siempre que modifiquemos los campos estado y ultimo registro para los nodos indicados. El estado sea mayor de 0 o distinto de 4. *BEFORE UPDATE OF flag_8 ON item* *when* *(new.ls_type in (10072, 10073, 10087,10141)* *and new.flag_8>0 and new.flag_8<>4)*
Entrada	El estado indica el resultado de la conciliación, toma un valor determinado dependiendo de si el o los registros han conciliado correctamente o al contrario. Tanto el atributo estado como el atributo ultimo registro tienen valor 0 en la entrada de base de datos.
Salida	Durante la conciliación el campo estado será modificado y el flag_2 será marcado a 1 en el disparador evitando una entrada en la tabla Audit. Trail. *:new.flag_2 := 1 + :old.flag_2 ;*

AWK Reformat_MI_WK	
Ejecución	La llamada a la awk se realiza desde el Host previo a la cargar del fichero.
Entrada	Se reformatea el campo feed o cuenta, que llega con un único feed. Además se separan los feed en varios dependiendo del valor del campo fecha. Esta separación se hace para evitar la sobrecarga de la conciliación debido al volumen de datos del fichero.
Salida	Se genera un nuevo fichero con un feed distinto según la fecha del dato.

AWK Reformateo de cabeceras	
Ejecución	La llamada a la awk se realiza desde el Host previo a la cargar del fichero.
Descripción	Para cargar los ficheros necesitamos que la descripción del nombre del atributo sea univoca.
Entrada	En la entrada tenemos un fichero con campos en la cabecera que se repiten.
Salida	Reescribimos la cabecera igual que la original, exceptuando el campo ambiguo el cual lo renombramos.

Ejemplo:

Entrada:

```
FECHA;FEED;LADO;TYPE;MAINLABEL;SECONDLABEL;TYPE;MAINLABEL;SECONDLABEL;MATURITY;BID;ASK
```

Salida :

```
FECHA;FEED;LADO;TYPE;MAINLABEL;SECONDLABEL;TYPE1;MAINLABEL1;SECONDLABEL1;MATURITY;BID;ASK
```

AWK Extracción de datos de tipo cadena	
Ejecución	La llamada a la awk se realiza desde el Host previo a la cargar del fichero.
Descripción	En algunos casos necesitamos extraer parte de la información de campos que contienen algunos campos del fichero, generar campos nuevos con esta información.
Entrada	En la entrada tenemos el fichero original con sus campos originales, alguno de ellos de esos campos contienen la información especifica a extraer. Utilizamos las funciones propias del lenguaje AWK como son la función split. Esta función guarda en un array los fragmentos de la cadena, separándolos por un carácter especifico. Luego accederemos a esta información accediendo al array , indicando en el índice el orden según el orden en que se haya separado. Entrada Split : ***|BCPN.IN|BVL|Securities|9m|5|SMS*** Carácter separador : |
Salida	Obtenemos un nuevo fichero con campos nuevos que contiene datos extraídos de otros campos. Utilizamos las funciones propias del lenguaje AWK como son la función split. ***a[1] = BCPN.IN*** ***a[2]= BVL*** ***a[n]= SMS***

Ejemplo:

Entrada:

```
FECHA;FEED;LADO;SYMBOL;ALIAS_MUREX_EU;ALIAS_MUREX_EU_2;ALIAS_MUREX_LATAM;ALIAS_MUREX_LATAM_2
30/11/2009;MI_AC;L;VOEQ-BCO_COM_POR-9M-105;|BCPN.IN|BVL|Securities|9m|5|SMS;;;
```

Salida:

```
FECHA;FEED;LADO;SYMBOL;ALIAS_MUREX_EU;LABEL;MARKET;ALIAS_MUREX_EU_2;ALIAS_MUREX_LATAM;ALIAS_MUREX_LATAM_2
30/11/2009;MI_AC;L;VOEQ-BCO_COM_POR-9M-105;|BCPN.IN|BVL|Securities|9m|5|SMS;BCPN.IN;BVL;;;
```

PROCEDURE MI_INFORME	
Ejecución	El procedimiento es llamado en el script de automatización después de realizarse la conciliación.
Descripción	Este procedimiento muestra la información resultante de la conciliación, dependiendo del tipo de cruce se indica el producto, para cada producto se requiere mostrar distintos datos.
Objetos o funciones	Este procedimiento muestra los datos mediante un cursor que hace referencia a una vista. Para mostrar los datos utiliza el paquete de Oracle dbms_output
Entrada	Se llama al procedimiento pasándole por variable el tipo de cruce o producto, y la fecha de la que requerimos de los datos.
Salida	Se genera un informe de datos clasificado por estados

PROCEDURE MI_CARGA_CRUCE	
Ejecución	El procedimiento es llamado en el script de automatización después de realizarse la conciliación.
Descripción	Este procedimiento muestra la información resultante de la conciliación, dependiendo del tipo de cruce se indica el producto, para cada producto se requiere mostrar distintos datos.
Objetos o funciones	Este procedimiento muestra los datos mediante un cursor que hace referencia a una vista. Para mostrar los datos utiliza el paquete de Oracle dbms_output
Entrada	Se llama al procedimiento pasándole por variable el tipo de cruce o producto, y la fecha de la que requerimos de los datos.
Salida	Se genera un informe de datos clasificado por estados

PROCEDURE MI_UPDATE_PATRON_FICHEROS	
Ejecución	El procedimiento es llamado en el script de automatización antes de realizarse la carga.
Descripción	Algunas conciliaciones pueden lanzarse varias veces si no se ha obtenido el resultado esperado, en este procedimiento se hace un borrado lógico de estos registros si existen, para volver a cargar los nuevos ficheros. Y comprueba el patrón de fichero que se espera, ya que alguno llega con fecha del día D y otros de D-1, hay que comprobar las fechas para evitar cargar ficheros de distinta fecha.
Objetos o funciones	Utilización de la función Oracle replace para generar el formato de fichero con la fecha esperada a partir de uno previo sin declaración de la fecha.
Entrada	Se llama al procedimiento pasándole por variable el tipo de cruce o producto.
Salida	Modificación si es necesaria en caso de existir reformateo.

PROCEDURE INSERCION_WORKFLOW	
Ejecución	El procedimiento es llamado en el script de automatización después de la comprobación de carga.
Descripción	Este procedimiento crea una entrada en la tabla workflow_queue, una vez creada esta entrada, el proceso workflow salta y comienzan a ejecutarse las reglas de conciliación realizadas en la herramienta de conciliación.
Objetos o funciones	Creamos un cursor que lee el número de cuenta del producto indicado en la variable de entrada, y realiza un insert en la tabla con los valores correspondientes.
Entrada	Se llama al procedimiento pasándole por variable el tipo de cruce o producto.
Salida	Se genera una entrada en la tabla workflow_queue

Capitulo IV

Experimentación

4.1 Ejemplo de Funcionamiento TLM

TLM muestra un árbol al margen izquierdo, donde podremos ir creando los distintos objetos.

4.1.1 Creación de una cuenta:

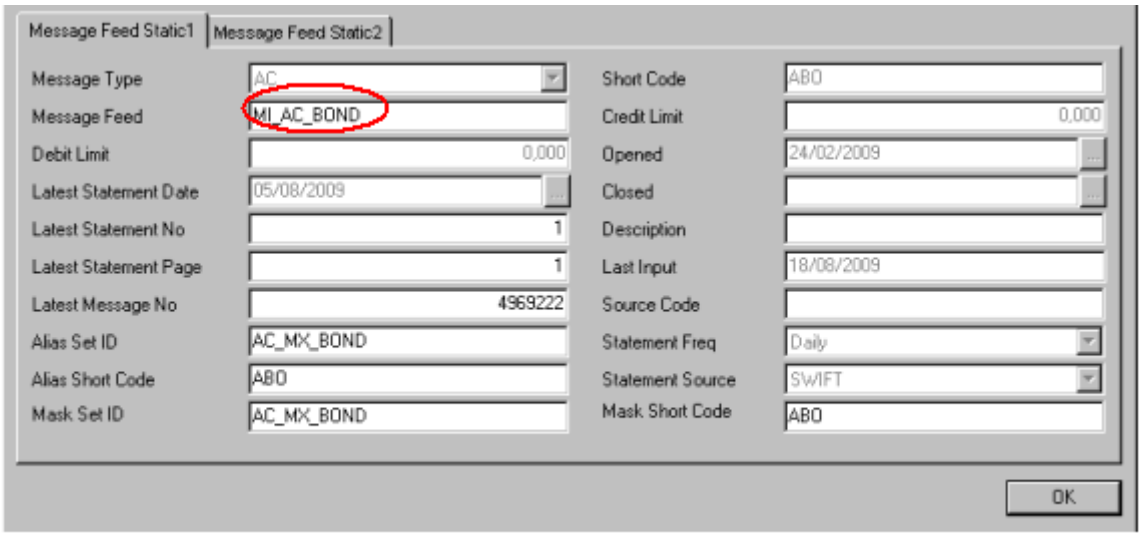
En primer lugar deberemos de tener creada una categoría. Para ello deberemos de darlo de alta en Business data.
Se abrirá una ventana donde deberemos de introducir el nombre que deseemos dar a la categoría. Tras crear las categorías podemos comenzar a crear las cuentas.

Árbol para la creación de categorías.

Para crear la cuenta deberemos de tener en cuenta los datos parametrizado en el SmartSchema, y los campos obligatorios que deben venir en las extracciones. En este caso el FEED, es el elemento más importante. Ya que tenemos que hacer coincidir el nodo del SmartSchema, y el FEED del fichero.

```
FECHA;FEED;LADO;ALIAS;PRICE;PRICE1
26/08/2009;MI_AC_BOND;L;|ARARGE031633|DEFARG49MAY8.75|BD USD GVT|PRICE|;28.2847;28.2847
```

El campo Message_type dará la opción de elegir el nivel dentro de la jerarquía de nodos creada en el SmartSchema.

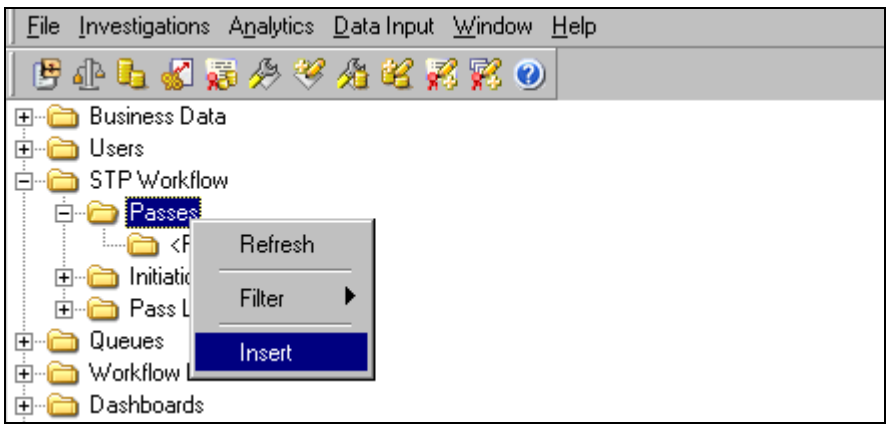


4.1.2 Creación de una Initiation:

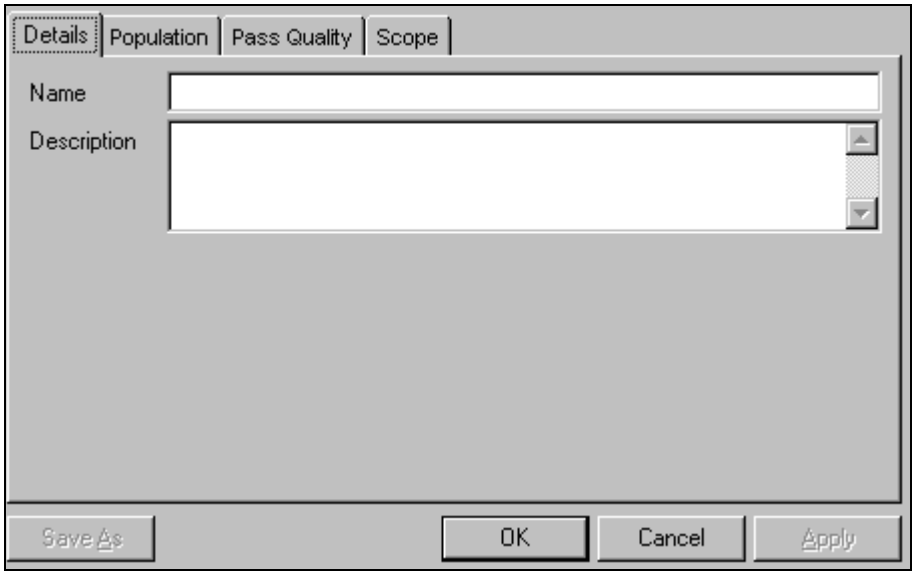
La initiation es una composición de pases, por lo que crearemos en primer lugar un pase.

Para crear un pase nos situaremos dentro el apartado STP Workflow, y elegiremos la opción pass.

4.1.2.1 Creación de Pases:



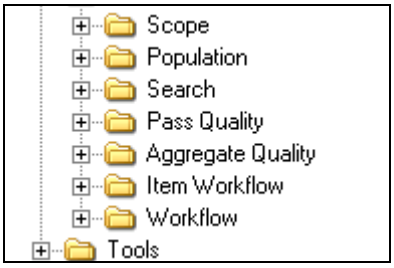
En primer lugar daremos un nombre al Pass, y una pequeña descripción. Y comenzaremos a añadir los objetos que se dividen en el navegador de pantalla mediante pestañas.

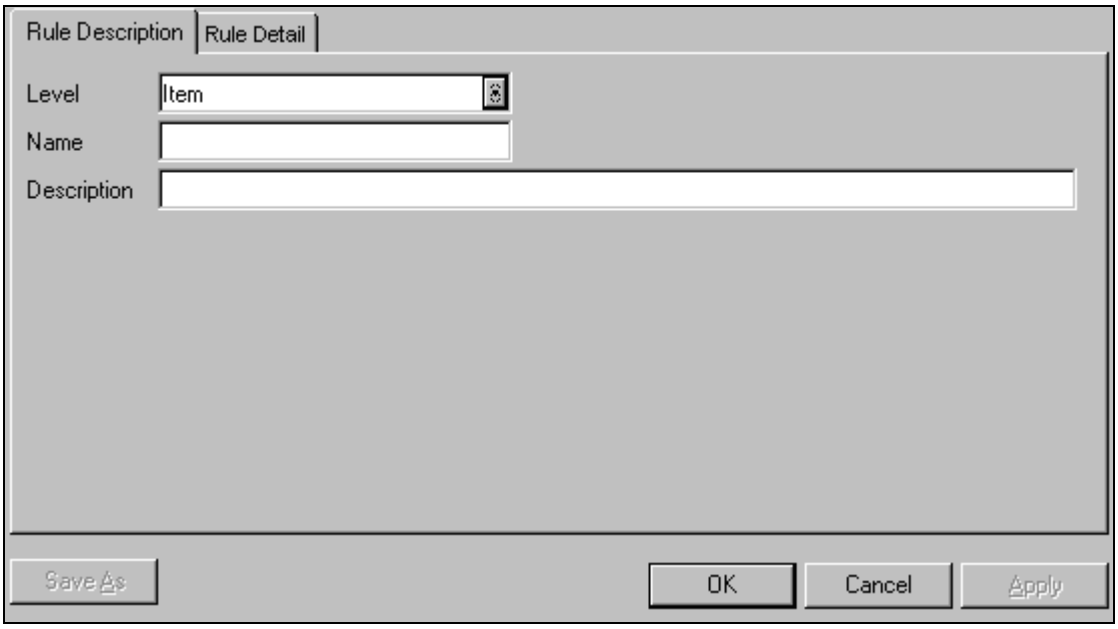


El siguiente paso será crear la Rule Scope.

4.1.2.2 Crear Rules:

Las Rule en general presentan todas la misma apariencia, un nombre, una descripción y el nivel dentro de la jerarquía.

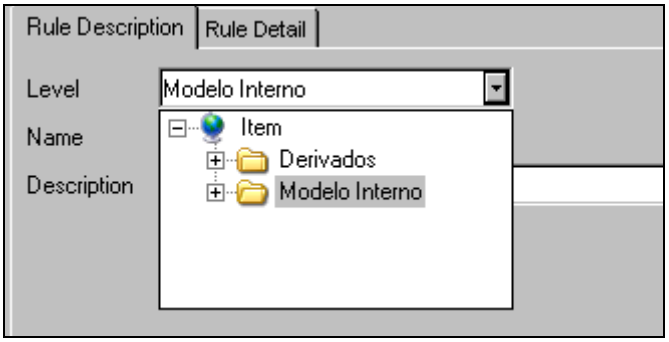




Pantalla inicial para crear reglas.

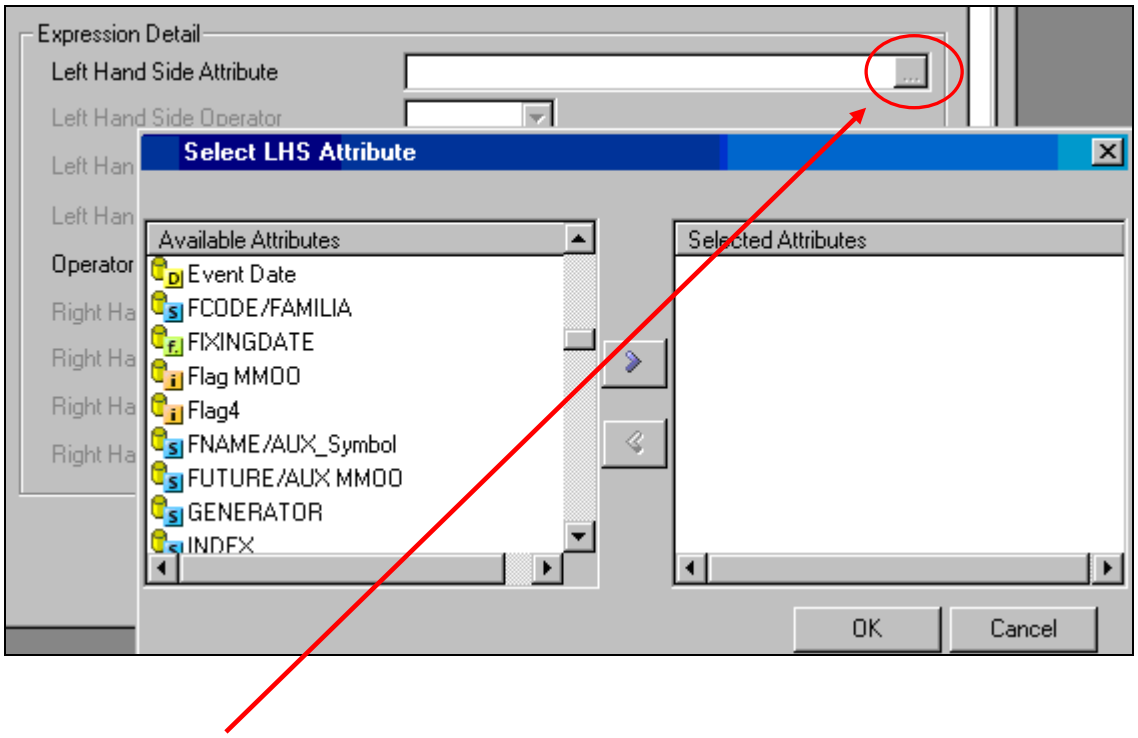
Debemos de tener en cuenta que si el nivel dentro de la jerarquía es el menor posible, no podremos seleccionar ni datos, ni campos de niveles superiores. En el caso del Scope que se seleccionan registros de varios nodos para cruzar, se suele seleccionar la categoría común.

En este ejemplo Modelo Interno es la categoría que engloba los nodos de carga de los dos ficheros, los nodos son AC y MX.

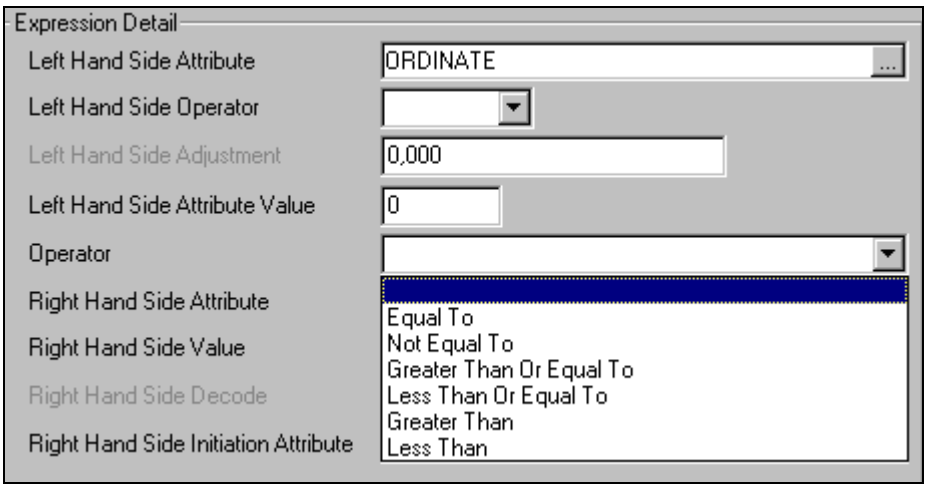


Selección de nivel en la jerarquía del universo

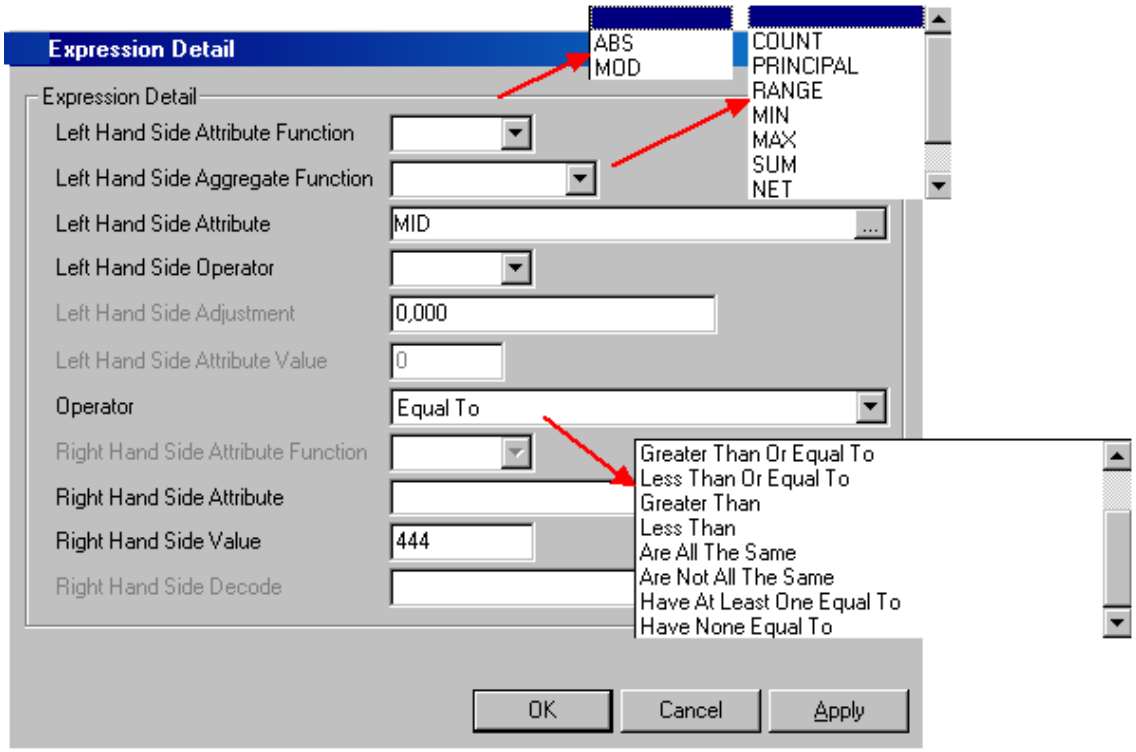
Una vez seleccionado el nivel dentro de la jerarquía donde queremos actuar, comenzamos a parametrizar las reglas.

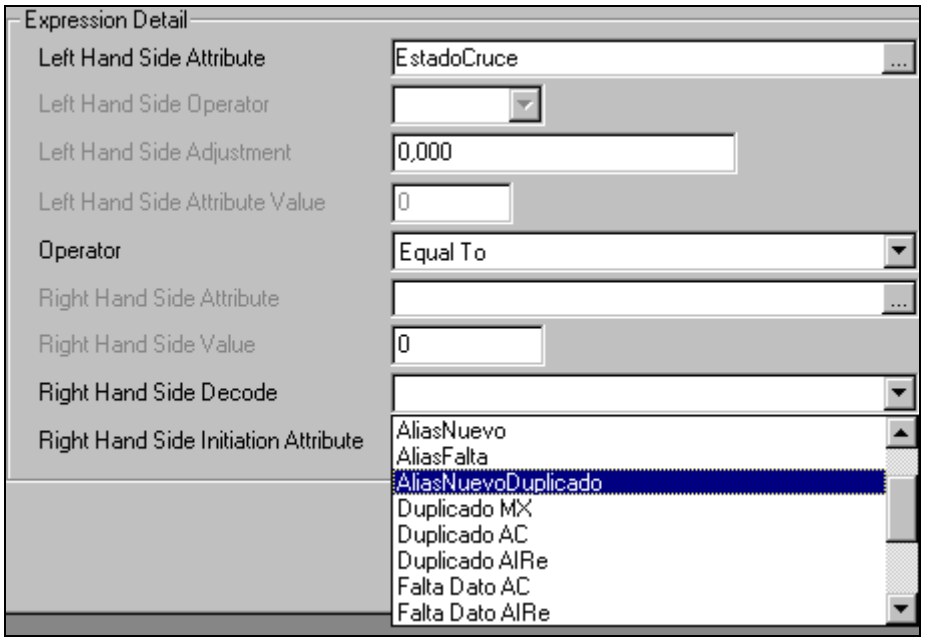


Seleccionamos el campo LHS, seleccionamos el operador y la constante o campo al que queremos comparar.

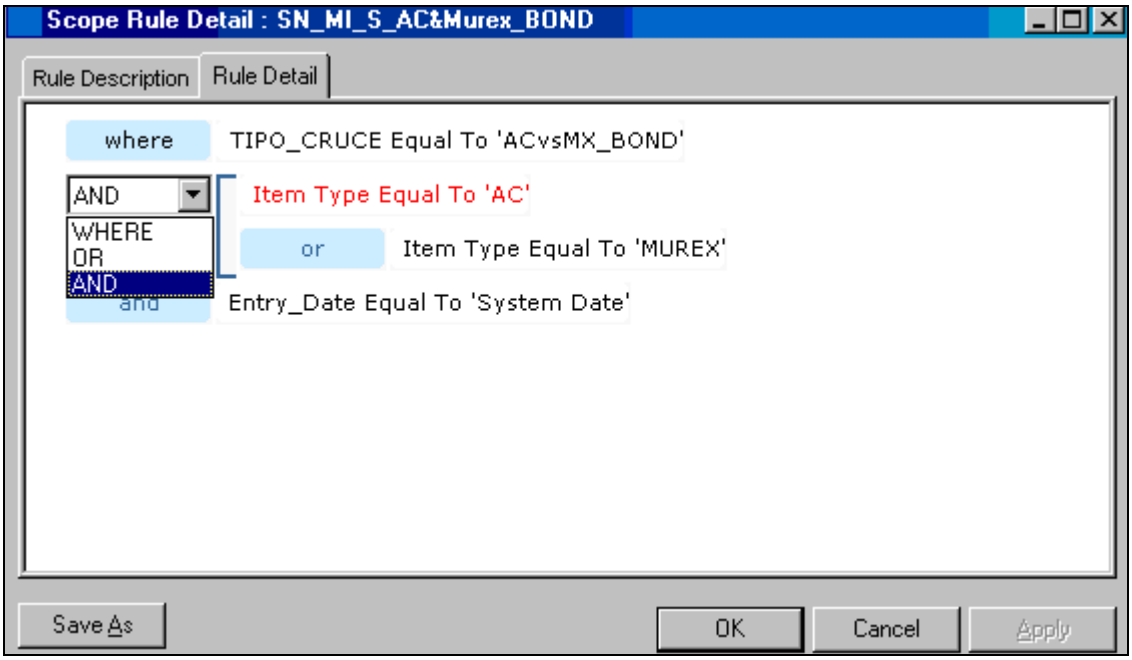


Se puede asignar a una constante, la cual se introduce en el campo RHS value, o se puede asignar a un atributo.

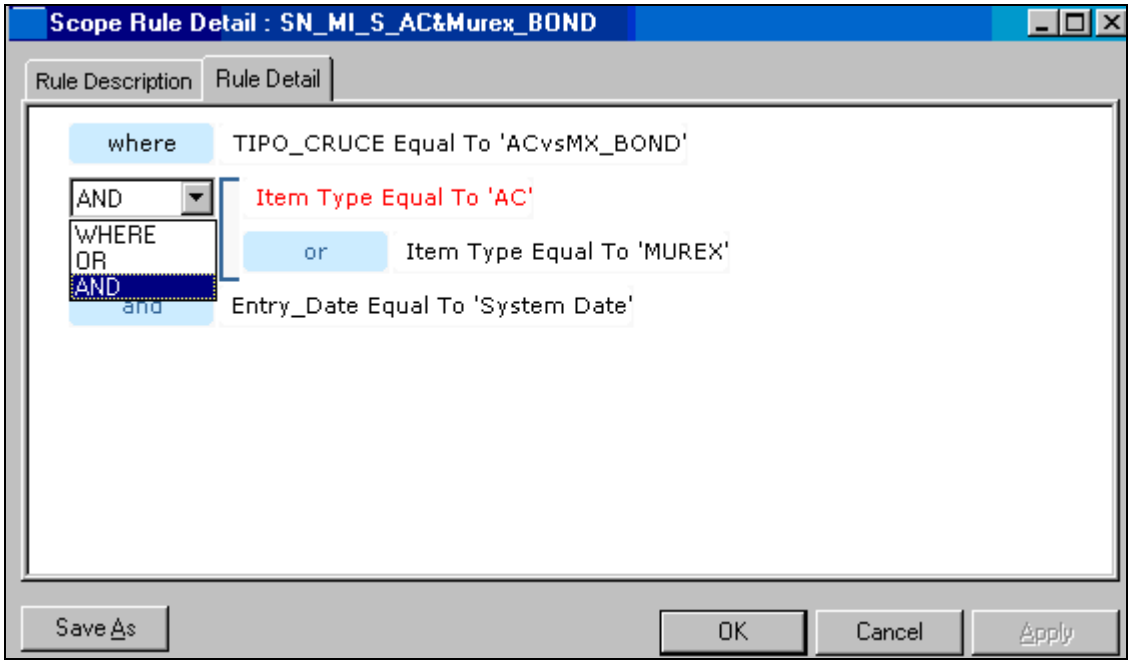




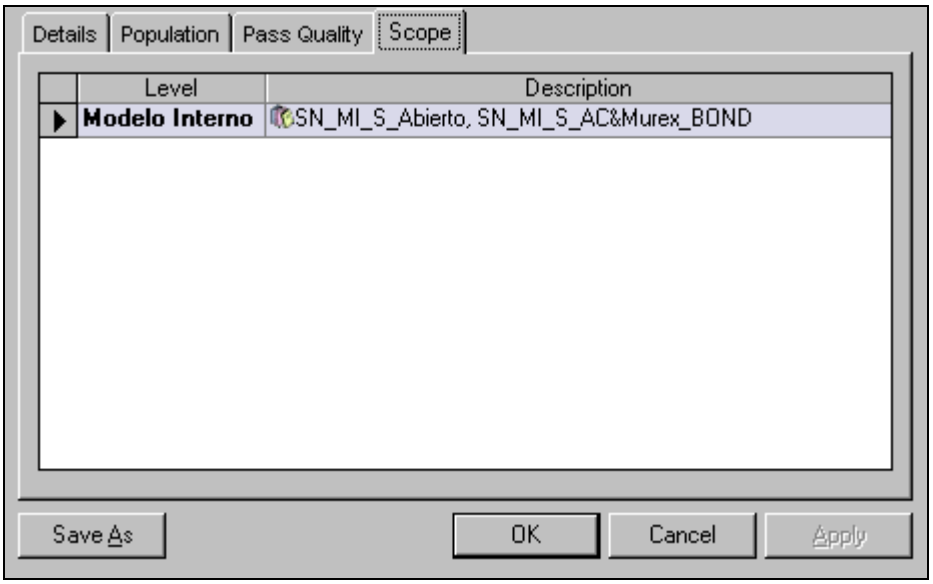
Tras insertar más de una regla, podremos unirlas con los operadores.



Además podemos crear subsentencias de otras reglas, para ello seleccionaremos la opción indent que hará que la sentencia quede aun subnivel inferior.



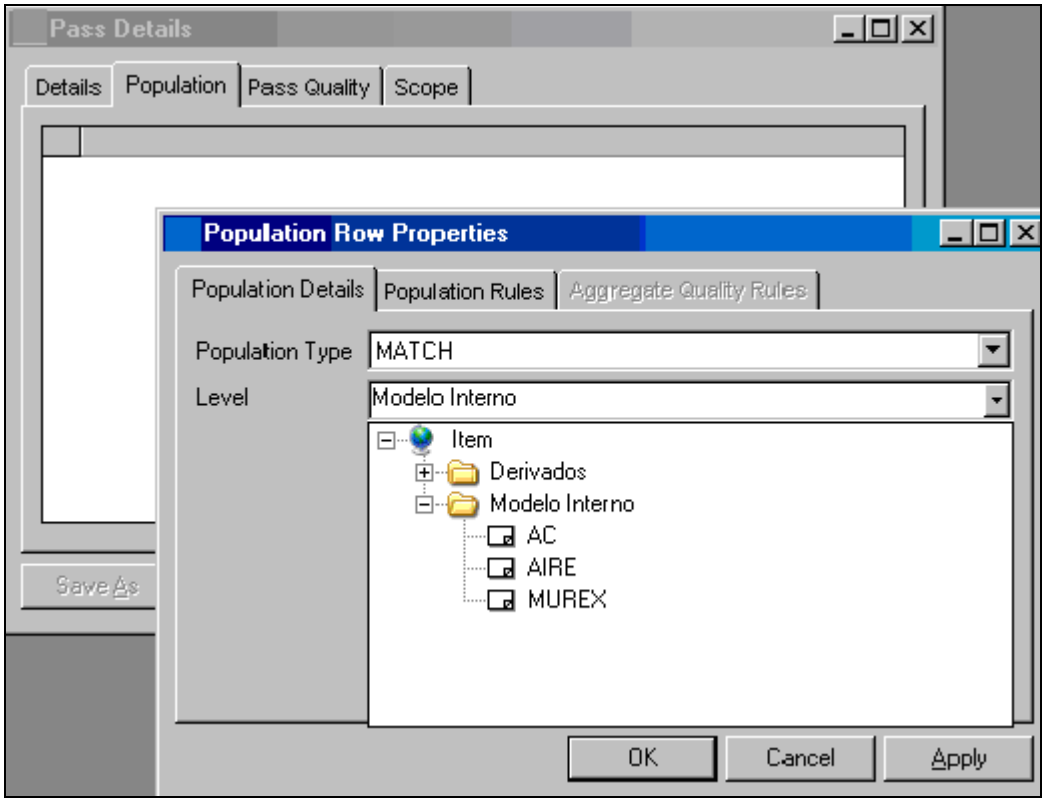
Por ultimo asociaremos las reglas scope a la pestaña que asocia estas reglas, a las ejecución del flujo de trabajo. Las reglas del Scope y la asociación de las reglas deben estar asignadas al mismo nivel, o a un nivel inferior al del Scope.



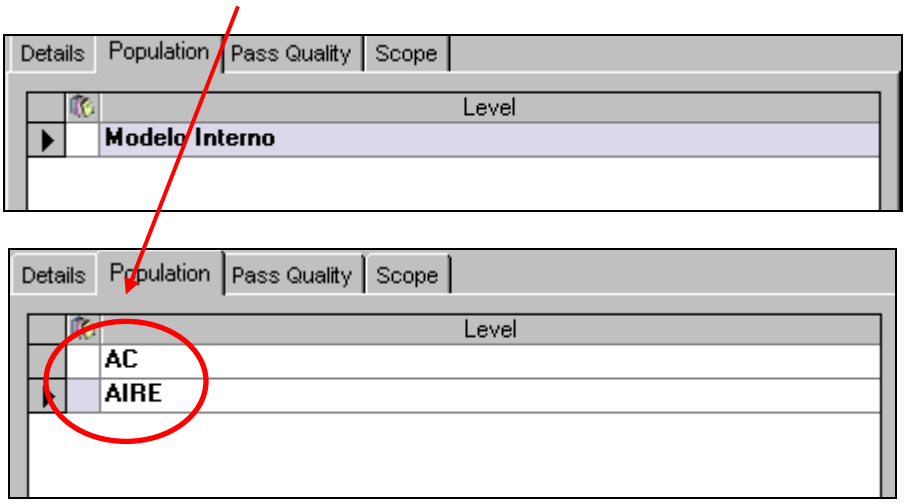
El siguiente paso será crear la population.

4.1.2.3 Crear una Population:

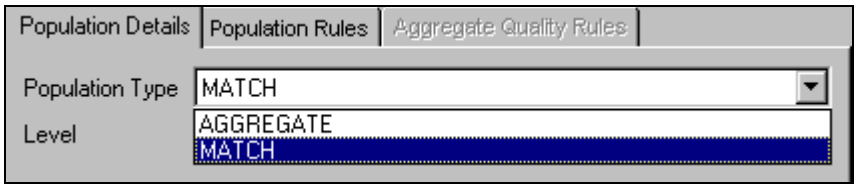
Para crear una population debemos de seleccionar en primer lugar el nivel de la categoría donde queremos agrupar según los atributos.



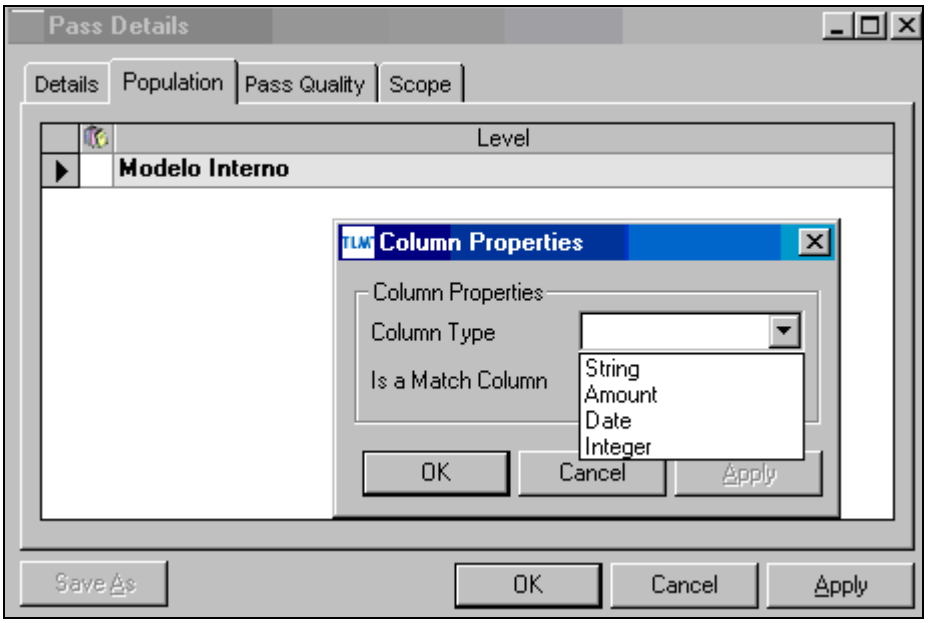
La opción a escala superior de los nodos se selecciona cuando los atributos son iguales tanto para la partida como para la contrapartida. En caso de ser distintos deberemos de añadir dos population o las que consideremos necesarias, una para cada nodo.



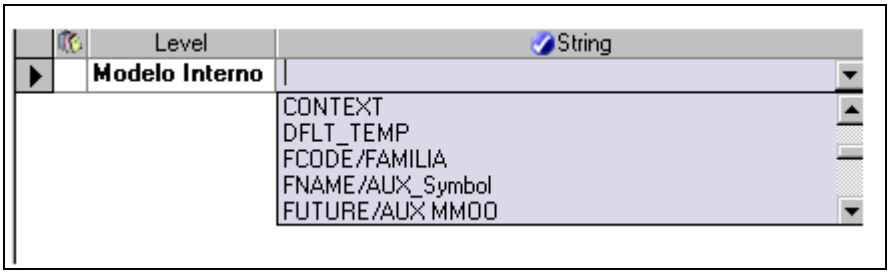
A la hora de añadir los nodos tendremos que indicar que tipo de population, queremos seleccionar match o aggregate.

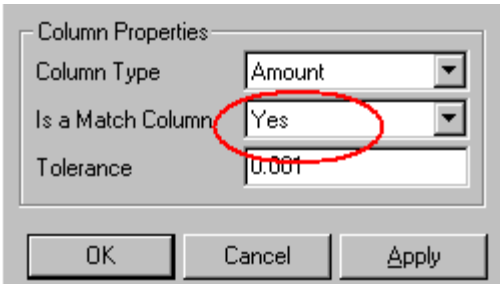


Una vez elegido el nivel, agregaremos los atributos. Para seleccionar el atributo, deberemos de pulsar el botón derechos donde seleccionaremos la opción agregar columna. Tendremos que seleccionar el tipo del atributo.



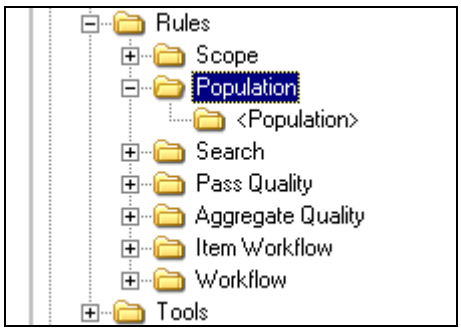
Una vez seleccionado el tipo del atributo, especificaremos el campo en concreto. En el caso de que sea un registro perteneciente al tipo amount que es un campo numérico decimal, podremos especificar una tolerancia, para los casos de registros con un redondeo despreciable.
De este modo si las cantidades numéricas están dentro del redondeo, se considera la misma cantidad y los registros quedarán agrupados.



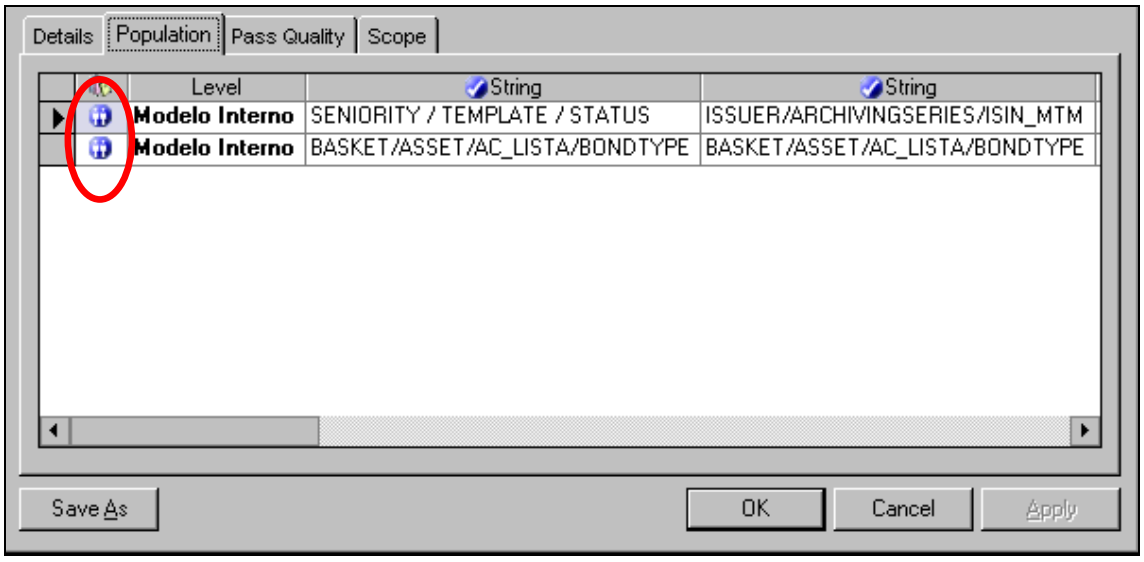


4.1.2.4 Crear una Population Rule:

Para crear una population rule, deberemos de procedes al igual que todas las reglas. En primer lugar deberemos de crear la regla en su apartado. Dándole el nivel necesario dentro de la jerarquía del universo, y teniendo en cuenta el nivel de la population. Siempre tendrá que ser menor igual que el nivel del objeto al que vaya a ser asociado.



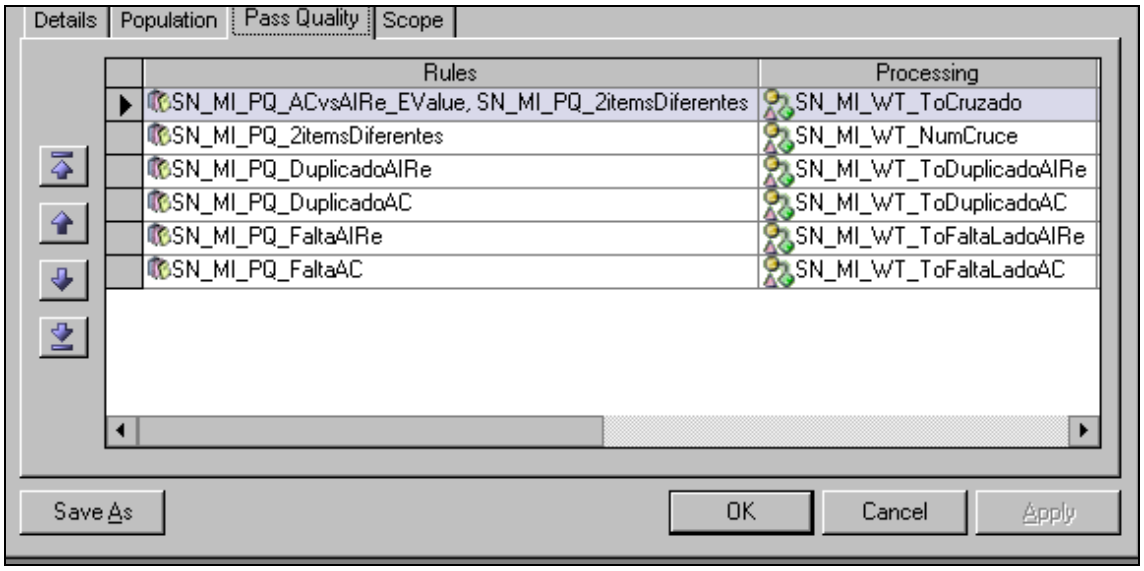
A continuación deberemos de asociarlo a las distintas population (agrupaciones) a las que la regla aplique.



Este ejemplo agrupara por un sitio o por el otro en función de las reglas asignadas.

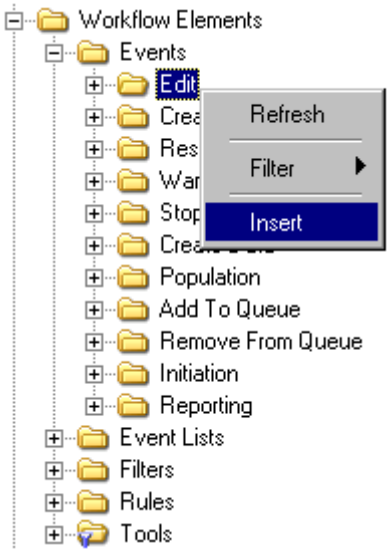
4.1.2.5 Crear una Pass_Quality :

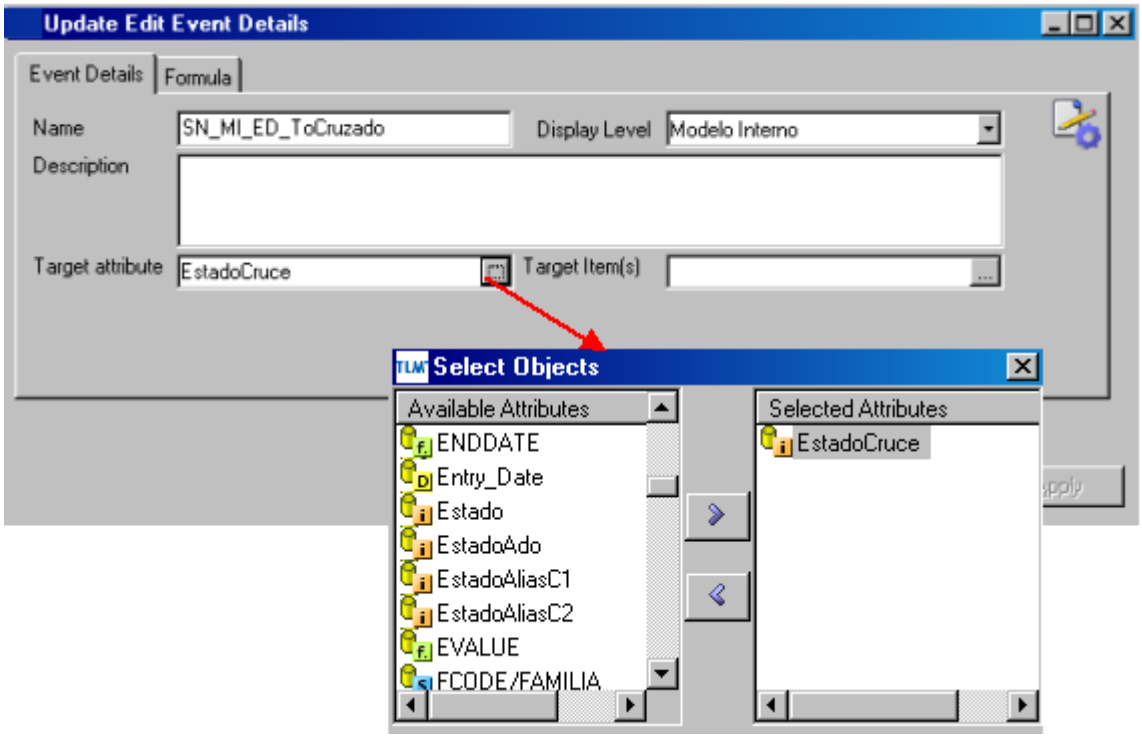
Para crear una Pass_Quality, deberemos crear previamente alguno de los objetos asociados en la Pass_Quality. Por cada línea de secuencia irán n reglas asociadas a una única Tool asociada a su vez a una lista de eventos. Las flechas del margen izquierdo servirán para ordenar las reglas que se ejecutarán secuencialmente según el orden.



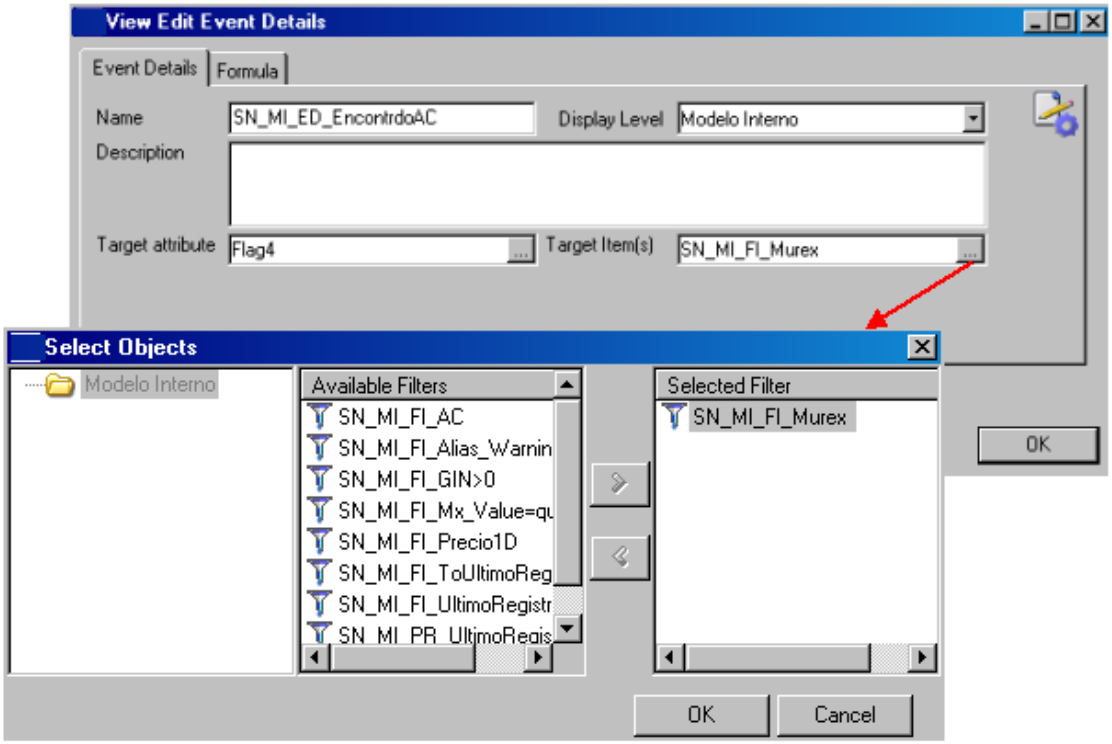
4.1.2.6 Creación de Eventos:

La ventana del evento, constara de dos partes o pestañas. En Event Detail, daremos una descripción del evento, nombre y nivel de jerarquía sobre la tabla item, y además tendremos un nuevo campo Target Attribute. En el campo Target Atribute seleccionaremos el campo sobre el que queremos modificar o asignar un valor.





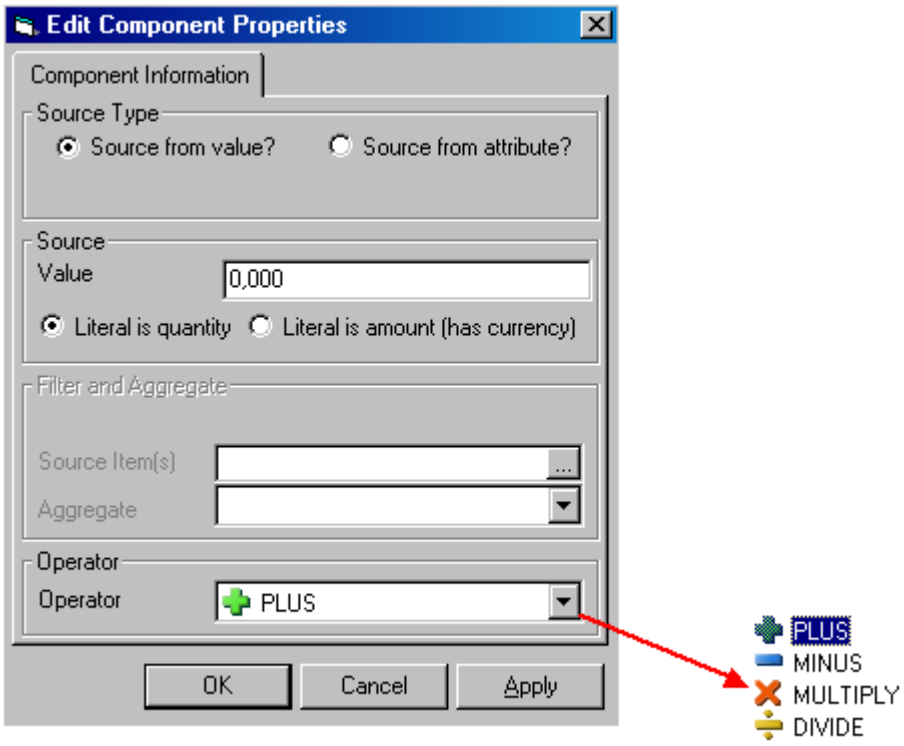
En Target Item(s), podremos añadir e manera opcional un filtro. Por ejemplo si tenemos una agrupación con registros de AC y MX que cumplen una Pass Quality, y solo queremos que cambien su estado los registros de MX, deberemos de crear un filtro con esta condición y asociarlo al evento. Los filtros se crean del mismo modo que las reglas.



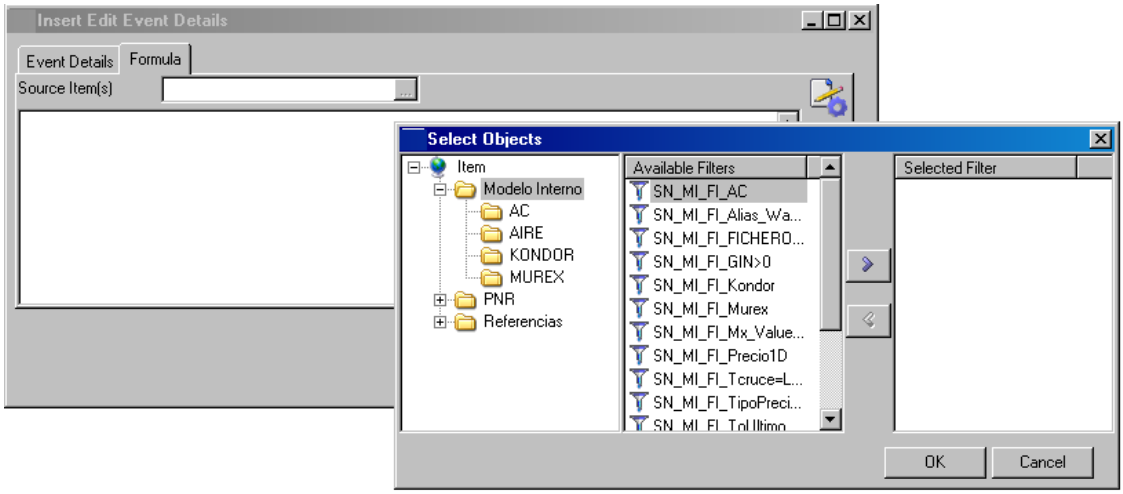
Para seleccionar el filtro que queremos agregar deberemos de situarnos al nivel creado, en ese nivel nos aparecerán todos los creados. Para seleccionar el que queremos añadir, pondremos el cursor encima del nombre del filtro y pulsaremos la flecha hacia la derecha. Para quitarlos basta con pulsar la flecha con dirección a la izquierda.

En la pestaña Formula, para añadir un valor deberemos de pulsar el botón derecho. Veremos una nueva ventana.

Source Type, es el origen del valor que le pasaremos. Puede ser una cantidad introducida por nosotros para ello elegiremos la opción Source value y rellenaremos el campo Value. También tendremos la opción de realizar distintos tipos de operaciones con el valor del atributo por una cantidad prefijada en Value.

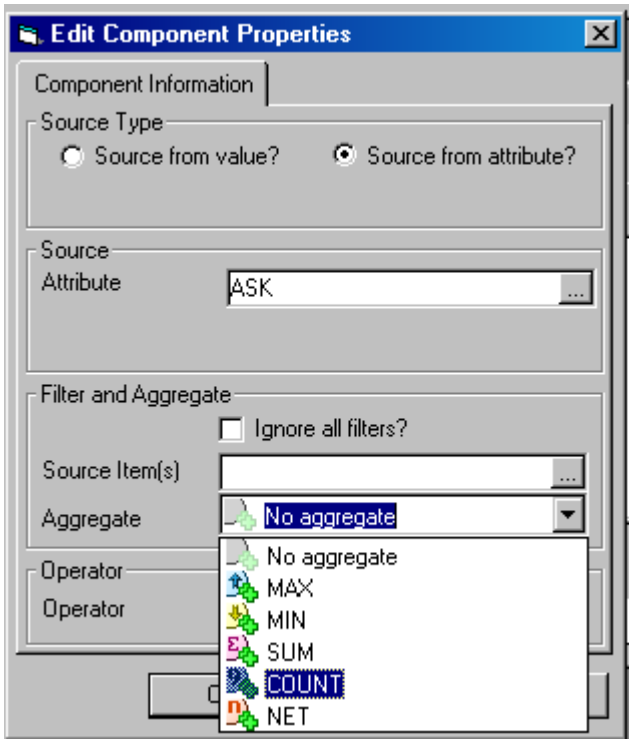


Otra posibilidad es seleccionar el valor de otro atributo para que transfiera el valor al atributo que hayamos seleccionado previamente para modificar. Para ello basta con seleccionar la opción Source From Attribute activándose la opción source item.



De esta forma el campo seleccionado únicamente tomara el valor de otro atributo si cumple la regla creada en los filtros. Por ejemplo si únicamente queremos que transfieran el valor aquellos registros inferiores a la unidad, tendremos que crear el filtro con dicha regla.

Cuando trasfiramos el valor de un campo a otro también tendremos la opción de utilizar funciones de agregación sobre las agrupaciones de registros seleccionadas.

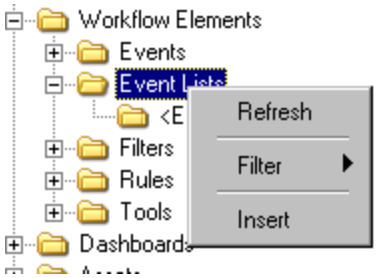


Una vez creados los eventos podremos asociarlos a los Event List.

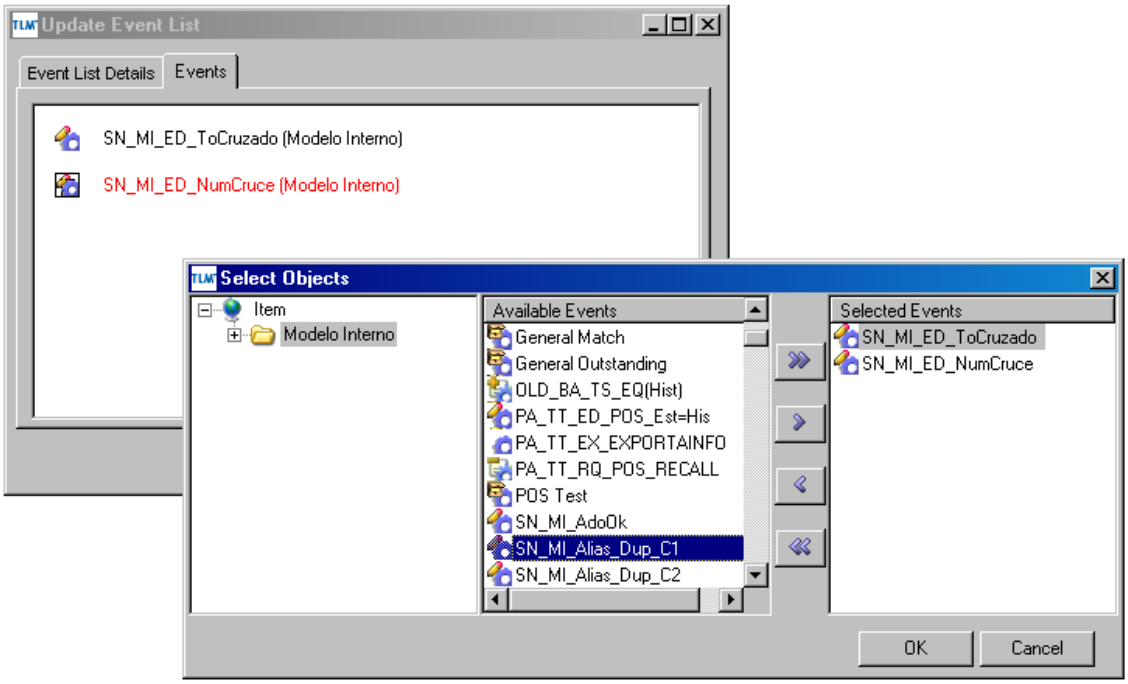
4.1.2.7 Creación de Event List:

Para crear un Event List hemos de haber creado previamente los eventos que queremos incluir.

En el navegador de pantalla para crear un Event List, nos mostrará la siguiente ventana. Constará de dos pestañas principales, seleccionaremos el nivel de jerarquía del Event List y el nombre que le atribuiremos y una pequeña descripción de su funcionalidad.



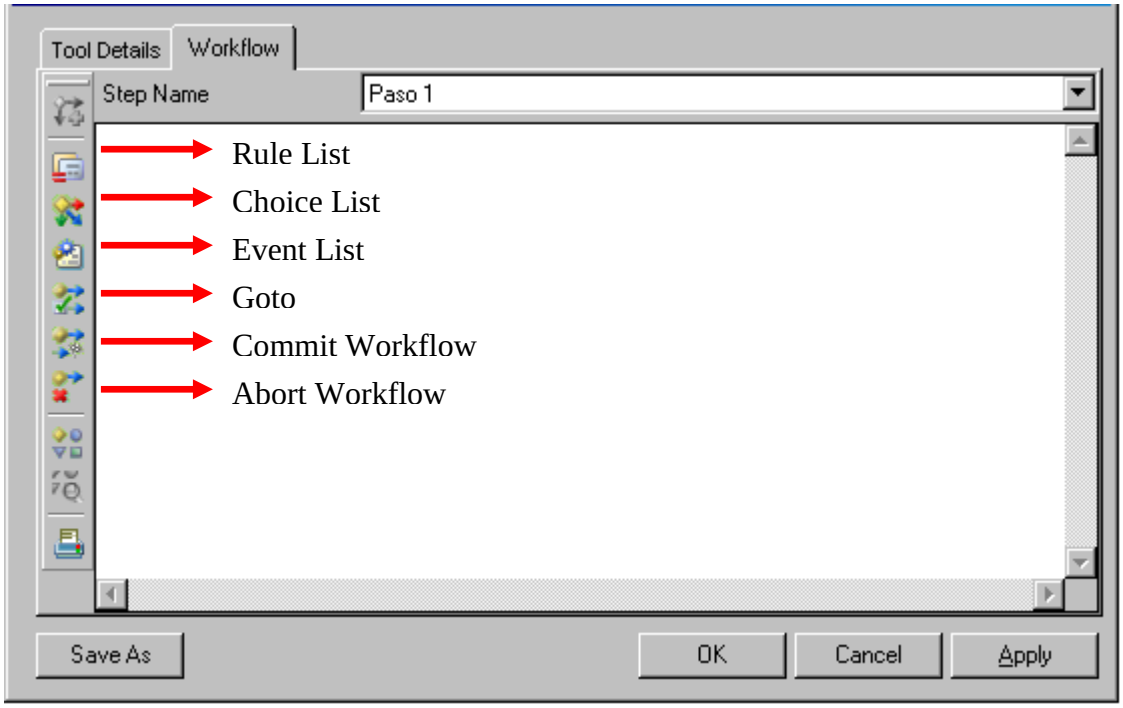
Los eventos los podremos seleccionar de una nueva ventana que nos mostrará toda la lista de eventos. Para seleccionarlo simplemente tenemos que posicionar el cursor sobre el evento y pulsar la flecha derecha donde iremos seleccionando todos los necesarios, la doble flecha actuara sobre todos los eventos del nivel, o sobre la agrupación que hayamos seleccionado. La lista de eventos se ejecutara de modo secuencial.



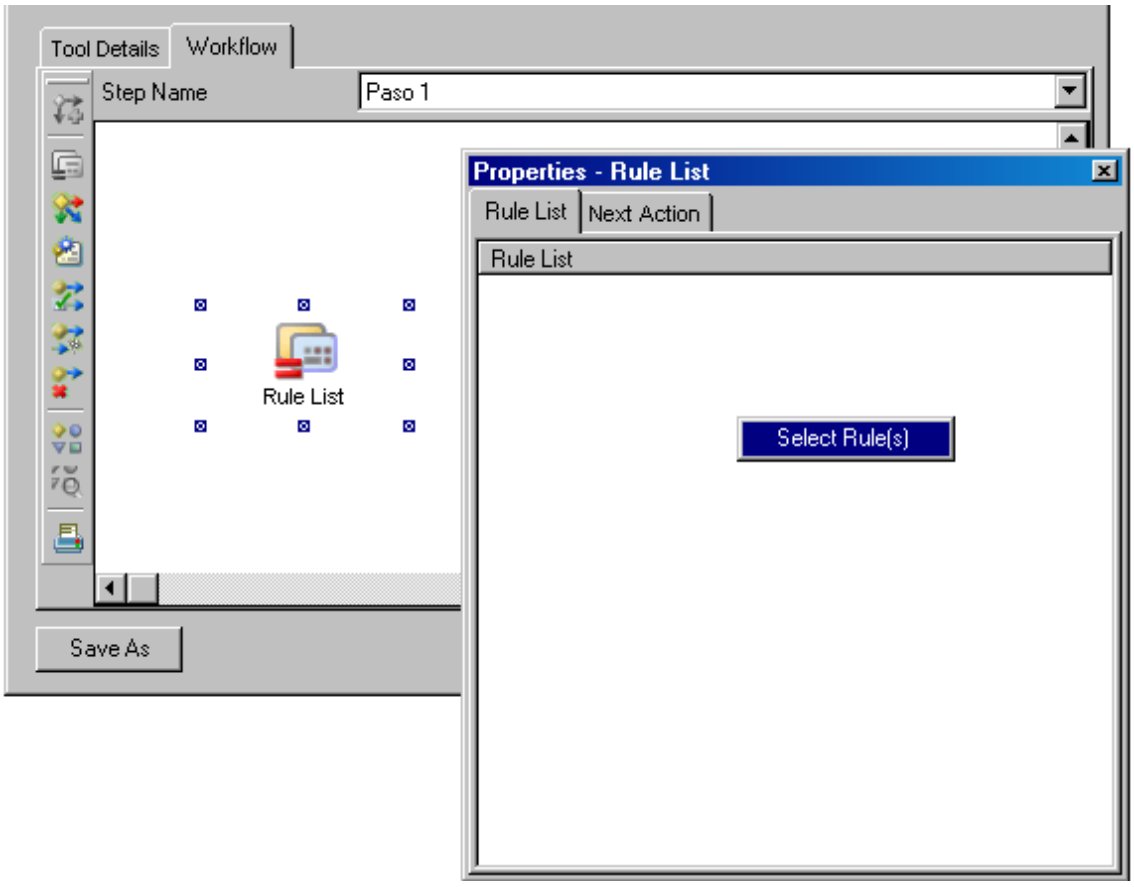
El Event List de ejemplo ejecutara dos eventos, el primero pondrá un campo al estado cruzado, y el segundo dotará a otro campo de un número de cruce.

4.1.2.8 Creación de Tool:

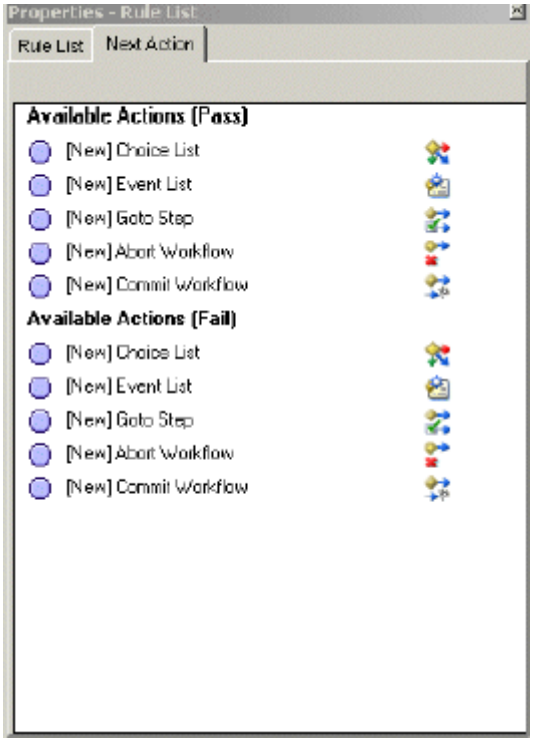
Para crear una Tool debemos haber creado previamente los Event List que vayamos a necesitar.
Como todos los objetos en TLM deberemos de seleccionar el nivel dentro de la jerarquía donde queremos que actué. En el caso de las Tool, el universo será item.



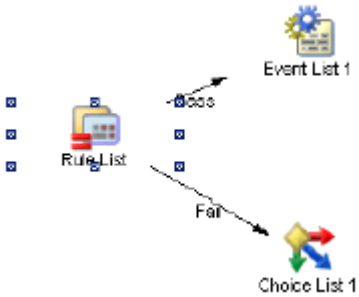
Si seleccionamos una Rule List, podremos condicionar la agrupación con una nueva regla,



En next action indicaremos el tipo de operación a realizar en dichos registros. Tendremos dos opciones, la opción en caso de cumplir la regla, y la opción en caso de no cumplirse.

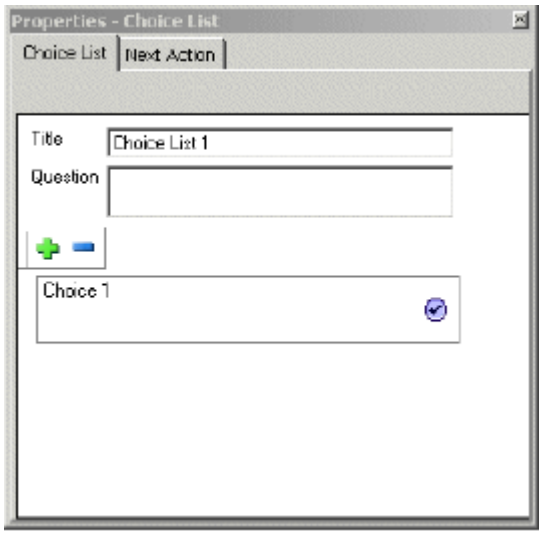


Por ejemplo, supongamos que apliquemos un Event List en caso de cumplirse la regla y en caso de no ser así, un Choice List.

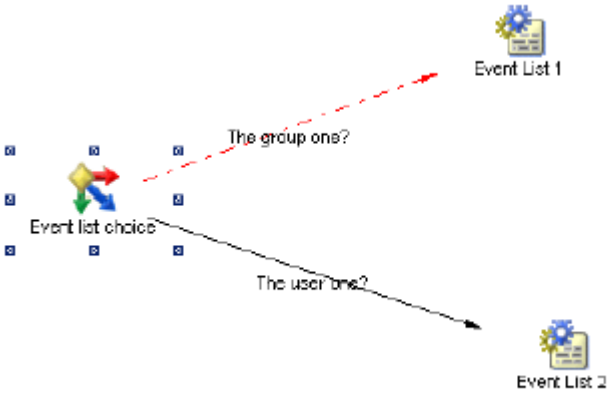


La opción del Choice List, le permite al usuario una serie de mensajes o de cuestiones. Sobre la acción del usuario se puede asociar una acción.

La Choice List, nos mostrará la siguiente pantalla, donde diseñaremos las preguntas que queremos mostrar al usuario.



En Next action, especificaremos lo que ocurre en casa una de las posibles elecciones del usuario, mediante Event List.



4.1.2.9 Creación De Informes:

Los dashboards son los informes que creamos en TLM para mostrar el resultado de la conciliación.

Los Grid son los informes más habituales, están compuestos por una Search List, que es una regla que sirve para que el usuario realice búsquedas de partidas en concreto. Y de una vista, que mostrará los campos según lo hayamos diseñado en esta.

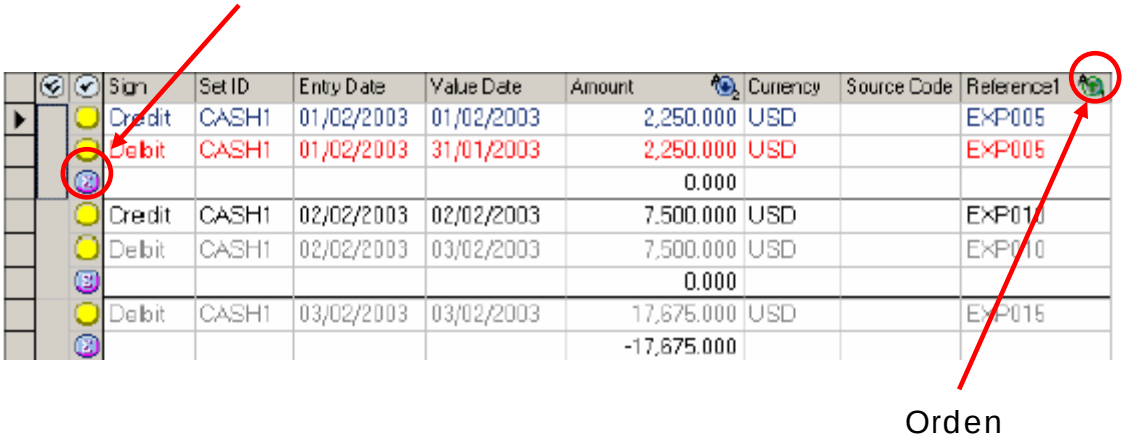
Creación de vistas:

Para crear la vista deberemos de asociar los nodos como objetos, los nodos se asocian en función de los distintos valores que pueden tomar los atributos al nivel inferior de herencia que nos interese mostrar.

Una vez seleccionado los objetos a mostrar, seleccionaremos los campos. Tendremos múltiples opciones de agrupación y ordenación sobre los campos así como opciones gráficas.

Para ordenar por atributo le asociaremos el tipo de orden a la columna bien ascendente o descendente, se puede realizar más de un orden, el orden descendente o ascendente vendrá marcado por una flecha que indicará el sentido del orden. El orden primario, que se marcará con un "1" que será el campo principal de orden y así sucesivamente.

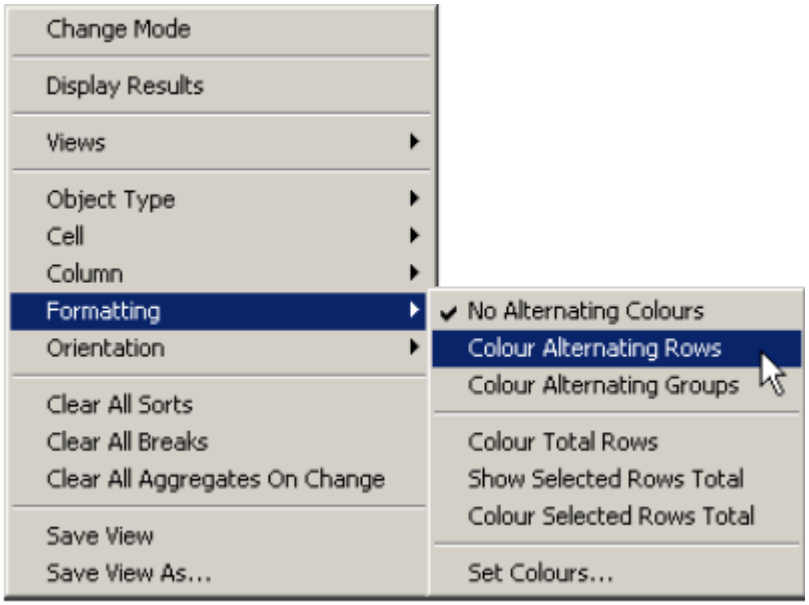
Para la agrupación de registros, se utilizará la opción Break on Change, donde seleccionaremos los campos por los cuales se agrupará. Además podemos añadir funciones de agregación, como puede ser por ejemplo la suma.



Sign	Set ID	Entry Date	Value Date	Amount	Currency	Source Code	Reference1
Credit	CASH1	01/02/2003	01/02/2003	2,250.000	USD		EXP005
Debit	CASH1	01/02/2003	31/01/2003	2,250.000	USD		EXP005
				0.000			
Credit	CASH1	02/02/2003	02/02/2003	7.500.000	USD		EXP010
Debit	CASH1	02/02/2003	03/02/2003	7.500.000	USD		EXP010
				0.000			
Debit	CASH1	03/02/2003	03/02/2003	17,675.000	USD		EXP015
				-17,675.000			

Orden

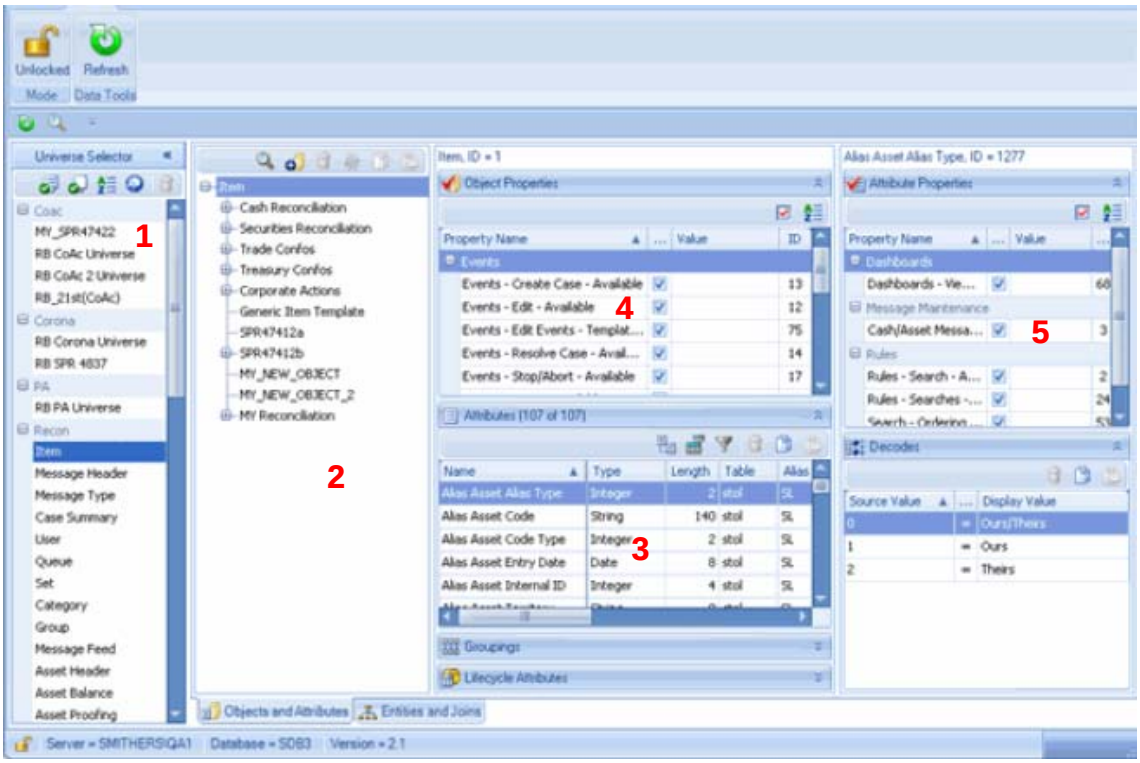
Visualmente en el informe podremos observar una raya y los atributos con el mismo color de fondo que estén agrupados.



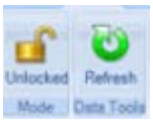
4.2 Ejemplo Funcionamiento SmartSchema

El SmartSchema permite personalizar los universos, o crear universos a medida para cada sistema de conciliación.

El interfaz del SmartSchema es el siguiente, se divide en la parte donde podemos observar todos los universos (1), la parte donde podemos ver los nodos y subnodos asociados al universo (2), la zona de los atributos en función de la jerarquía de los nodos y su asociación ya que existe la propiedad de herencia (3), la zona de las propiedades de los nodos (4) y por ultimo la zona de los propiedades de los atributos (5).



Para poder crear cualquier objeto en el SmartSchema, debemos de quitar el seguro, este seguro evita que modifiquemos algún elemento por error.



4.2.1 Crear nodos:

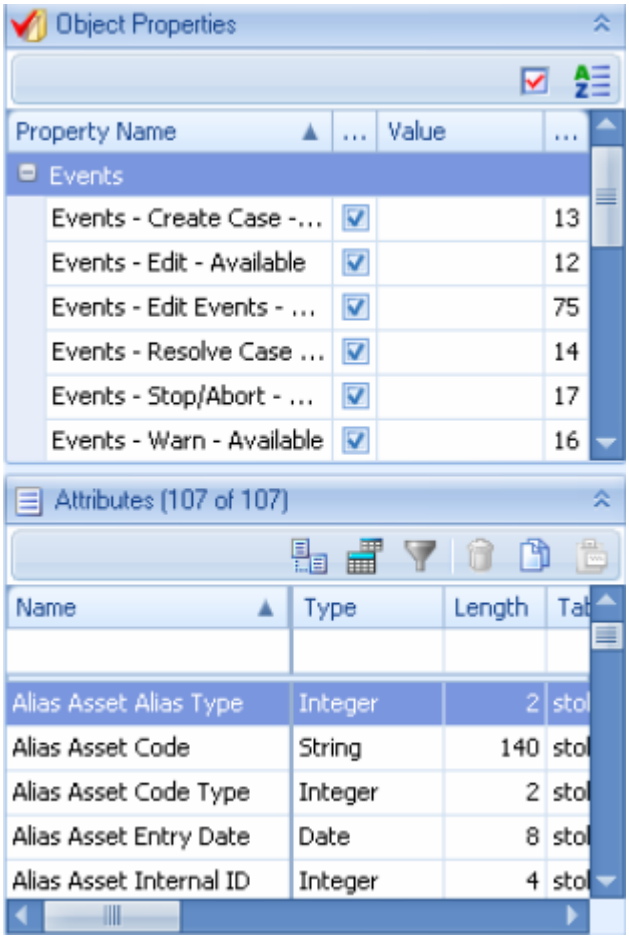
Para crear un nodo dentro el universo seleccionado deberemos posicionarnos en la zona donde se muestran a la altura en la jerarquía donde queramos crearlo.

Para crear nodos al nivel de item, que es lo habitual, deberemos de asociar los tres universos, Message Header y Message Type.

Para ello deberemos de de situarnos en cada uno de los universos y relacionarlo mediante la opción de /* Poner ejemplo y explicar mas fácil */

4.2.2 Propiedades de los nodos:

Para asignar las propiedades a los nodos tales como signo, y alguno de sus comportamientos deberemos de seleccionar en la lista de opciones aquellas que nos interesen marcar.



Al dejar el cursor encima de un atributo nos mostrara de forma automática la siguiente ventana donde se puede observar el número de atributos que tiene creado un nodo y cuanto de ellos heredados.

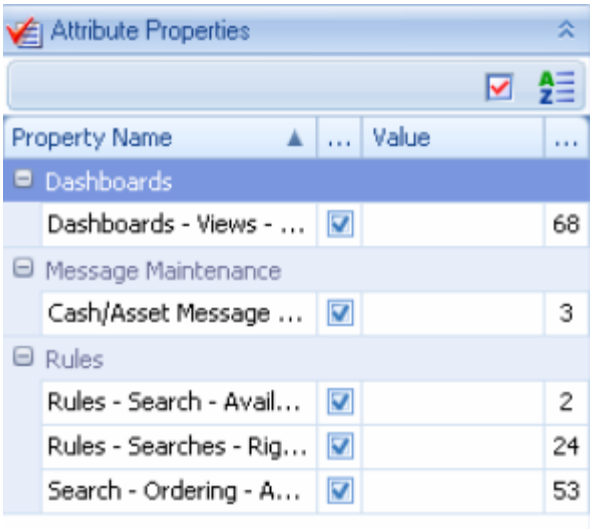
4.2.3 Crear atributos:

Para crear atributos deberemos de posicionarnos al nivel jerárquico al que queremos que se muestre dentro del universo. Si un atributo solo pertenece a un nodo deberemos de posicionarnos al nivel del nodo, si lo comparten dos nodos, al nivel superior que engloba a los dos.

Deberemos de darle un nombre elegir, un tipo de dato y dotarle del tamaño adecuado.

4.2.4 Propiedades de los Atributos:

Para dotar a un atributo de propiedades, que regirán su comportamiento dentro de TLM, deberemos de posicionarnos encima del atributo, donde al margen izquierdo se nos mostrará una lista de propiedades de las cuales seleccionaremos aquellas que nos interesen.



4.2.5 Generar el KB desde SmartSchema:

Desde el universo Message Type, podremos generar el KB.
Importar y Exportar los datos:

Los datos pueden exportarse e importarse en el SmartSchema, esta función se realiza al cambiar de entorno o para realizar copias de seguridad de los entornos.

Para exportar, no seleccionaremos los elementos que deseamos exportar, directamente se creará un fichero con los datos de todo el esquema.
Sin embargo, en importar si deberemos de seleccionar aquellos elementos que queremos que sean importados.

4.3 Experimentación del cliente ligero (versión WEB) - WEBCONNECT

La finalidad de las conciliaciones financieras es tener un informe con el resultado de las partidas. Las partidas únicamente pueden servir para certificar precios con las aplicaciones propias de los bancos y comprobar que han viajado de una aplicación a otra correctamente, o puede que sea necesario operar con ellas manualmente hasta cuadrar parcialmente todo los ficheros. Para ello los usuarios tienen una aplicación WEB a través de la cual pueden consultar la información necesaria y realizar operativa manual.

4.3.1 Conceptos Generales sobre WebConnect

¿Cómo acceder WebConnect/TLM Recons?

Para realizar la conciliación o consultar los resultados de la misma es necesario acceder a WebConnect/TLM Recons, a través de la siguiente dirección de Internet:

Entorno Preproducción:

http://webconnectpre.pre.corp/WebConnect/login/login.jsp

Entorno de Producción:

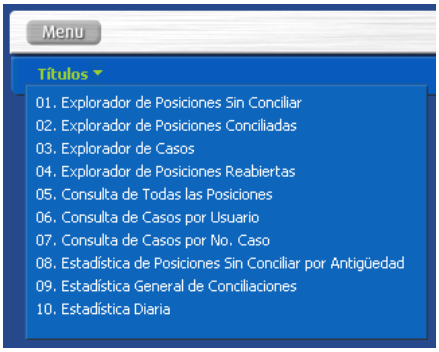
http://webconnectpro.gs.corp/WebConnect/login/login.jsp
Introduciendo el usuario y la password en la siguiente ventana:

Accedemos a la ventana principal de WebConnect/TLM Recons:

Menu Principal de WebConnect/TLM Recons

Para acceder al menú de usuario debemos pulsar sobre la opción [Menú] en la parte superior izquierda de la ventana:

Pulsando sobre la flecha accederemos a los informes disponibles para cada usuario, que más tarde comentaremos:



En cuanto a las opciones de menú disponibles: Options, Help y Logout.

- En la pestaña de ayuda, **Help,** se encuentra el botón de acceso a la información de ayuda.

- La pestaña de **Logout** es la opción que nos permitirá salir de la página de forma segura.

- La pestaña de **Options** es la opción que nos permitirá el cambio de contraseña.

4.3.2 Utilización de las Hojas de Trabajo

¿Qué es una hoja de trabajo?

Las hojas de trabajo ayudan al usuario a realizar varias tareas en paralelo al poder éste acceder a una o varias hojas al mismo tiempo, mostrando distintos informes o vistas.

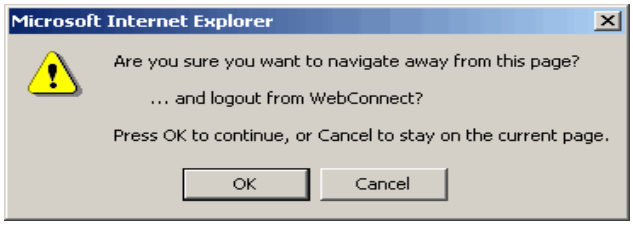
La primera hoja de trabajo se abre automáticamente cuando se comienza una sesión en WebConnect. Se pueden crear hojas de trabajo adicionales, hasta un máximo de cinco, haciendo clic en el icono Add Worksheet en la parte inferior derecha de la ventana.

Para cerrar una hoja de trabajo y los informes mostrados dentro de ésta, pulsaremos el icono  situado en la esquina inferior derecha de la misma pestaña.

Salir de WebConnect/TLM Recons:

Para desconectarse de WebConnect/TLM Recons, hacer clic en [Menú], y seleccionar **Logout**:

También puede desconectarse utilizando la tecla [F5] o se haciendo clic en el botón  , en estos dos casos aparecerá el siguiente mensaje:



Presionaremos [OK] para salir o [Cancel] para permanecer en WebConnect/TLM Recons.

4.3.3 Informes en WebConnect

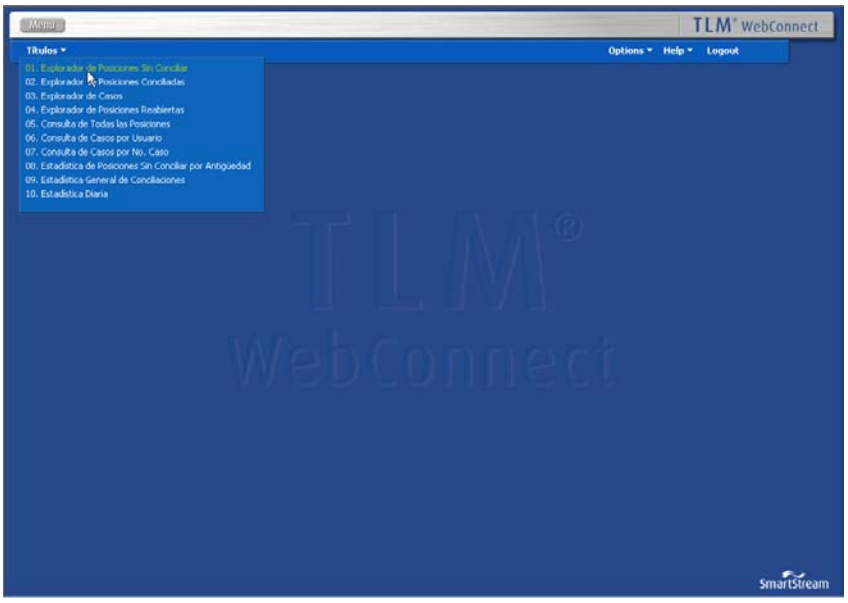
En los diferentes informes que se muestran en la aplicación, se presentará información concreta al usuario en distintos formatos, siendo los más comunes:

- Exploradores
- Tablas de Datos
- Gráficos

4.3.3.1 Como acceder a un informe:

1) Hacer click en el botón Menú, y a continuación seleccionar la opción que se desea consultar (Títulos):

2) Al pulsar sobre la flecha se despliega un menú con los distintos informes creados para la consulta de la conciliación de Títulos del Área de Seguros.



4.3.3.2 Cerrar una ventana:

Para cerrar un informe, seleccionaremos el icono localizado en la parte superior derecha del mismo. Al cerrar una hoja de trabajo se cerrarán automáticamente todos los informes mostradas dentro de la misma.

Botones de navegación:

Utilizaremos los siguientes botones para navegar dentro de cualquier informe de WebConnect:

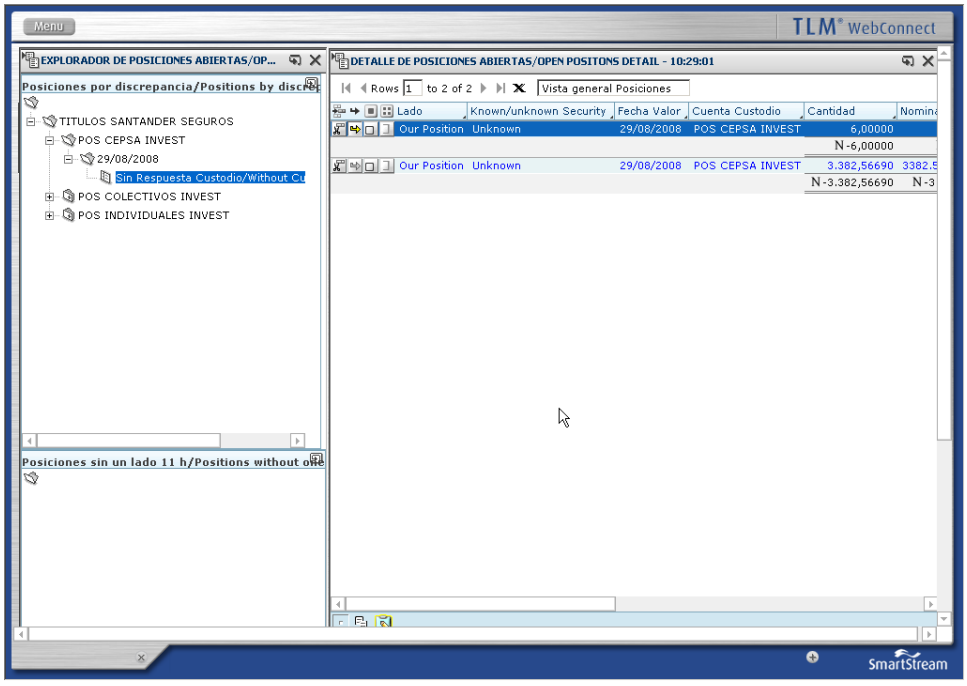
Rows 1 to 30 of 5000	Indica el número de registros mostrados en la ventana junto con el número total de registros que cumplen las condiciones de búsqueda.
▶	Avanza pagina
▶|	Avanza hasta la última página
◀	Retrocede una página
|◀	Retrocede hasta la primera página
	Exporta a Excel

Tipos de ventanas:

Los datos serán presentados en tres tipos de informes:

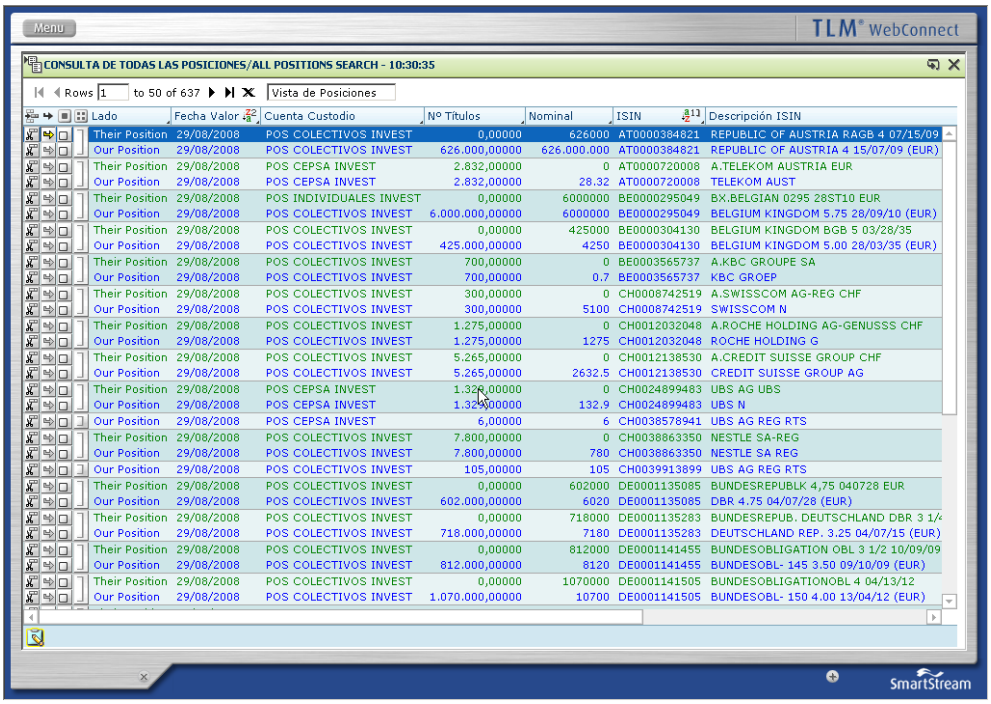
- **Exploradores:**

  Muestran de forma agrupada el número de registros que cumplen un criterio, permitiendo seleccionar el detalle de los registros que forman parte de la agrupación. Ejemplo: Explorador de Posiciones Sin conciliar:



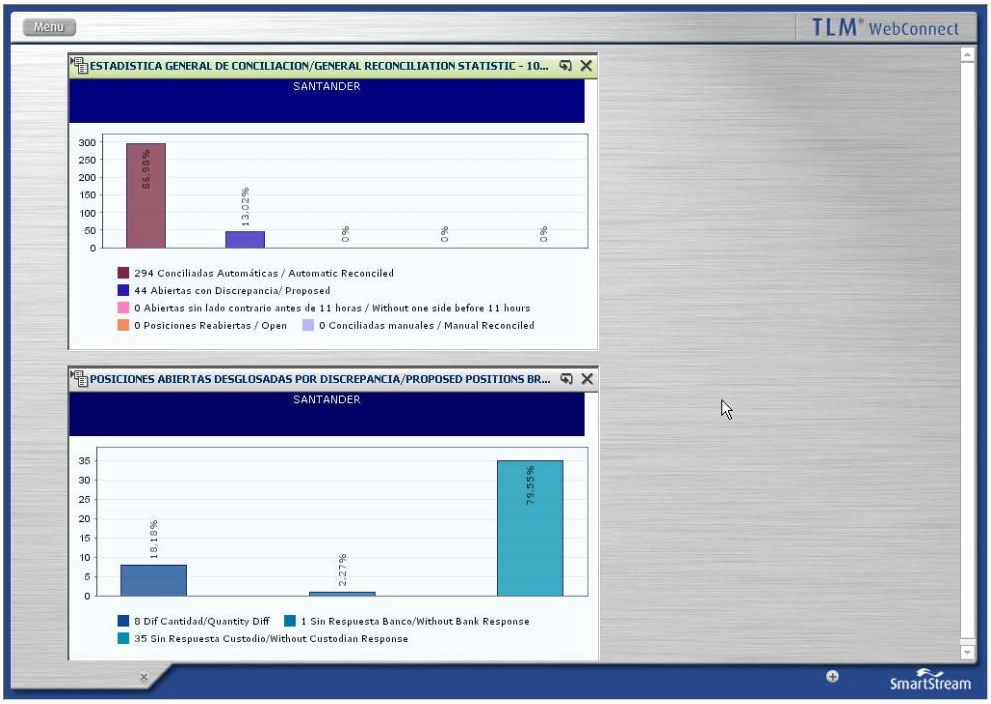
- **Tablas:**

  Muestran las partidas ordenadas de un modo predeterminado permitiendo su consulta. Ejemplo: Consulta De Todas las Posiciones.



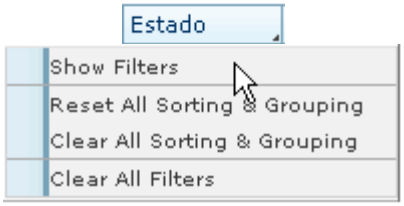
- **Gráficos:**

  Mostrará los datos de un modo más global, permitiendo incluir gráficos de barras y otros controles. Ejemplo: Estadística General de Conciliaciones.



4.3.3.3 Ordenación de los datos:

En los informes de tipo tabla, en algunas de las cabeceras de las columnas aparece el símbolo que indica que hay un menú desplegable.
En función de sí pulsamos sobre él directamente o pulsamos botón derecho del ratón se desplegarán distintos submenús:

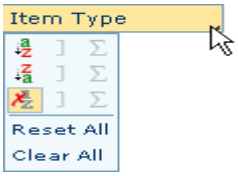
Pulsando sobre la cabecera del campo Estado directamente accedemos al siguiente submenú:	Pulsando con el botón derecho del ratón sobre la cabecera del campo Estado se despliega el siguiente submenú:
	

En los puntos siguientes se explicará el funcionamiento de dichos submenús.

Establecer un orden:

Al ejecutar un informe, los datos aparecen ordenados según se haya indicado en la parametrización, sin embargo, el usuario puede modificar el orden establecido seleccionando aquellos campos por los que desea ordenar e indicando el tipo de orden deseado. Para ello:

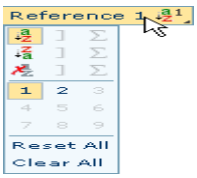
1. Hacer clic en la cabecera de la columna que se desea ordenar para mostrar el menú de opciones.



2. Para ordenar las columnas de registros en un orden ascendente, seleccionar . Para ordenar los datos en orden descendente, seleccionar . Una vez seleccionado el orden, en la cabecera se mostrará un símbolo indicando el tipo de orden seleccionado.
Si la columna no tiene todavía asignado un orden, no se mostrará ningún símbolo en la cabecera y dentro del menú de la cabecera, el icono aparecerá seleccionado.

Configurar la prioridad de ordenación:

1. Hacer clic en la cabecera de la columna cuya prioridad de ordenación se desea cambiar. Aparecerá el siguiente menú:



2. Elegir el orden que se desea para dicha columna.
La prioridad del orden de las columnas se mostrará en la cabecera de las mismas mediante un número del 1 al 9. En el siguiente ejemplo, la ordenación se hará en primer lugar por la columna Reference 1 y a continuación por Value Date:

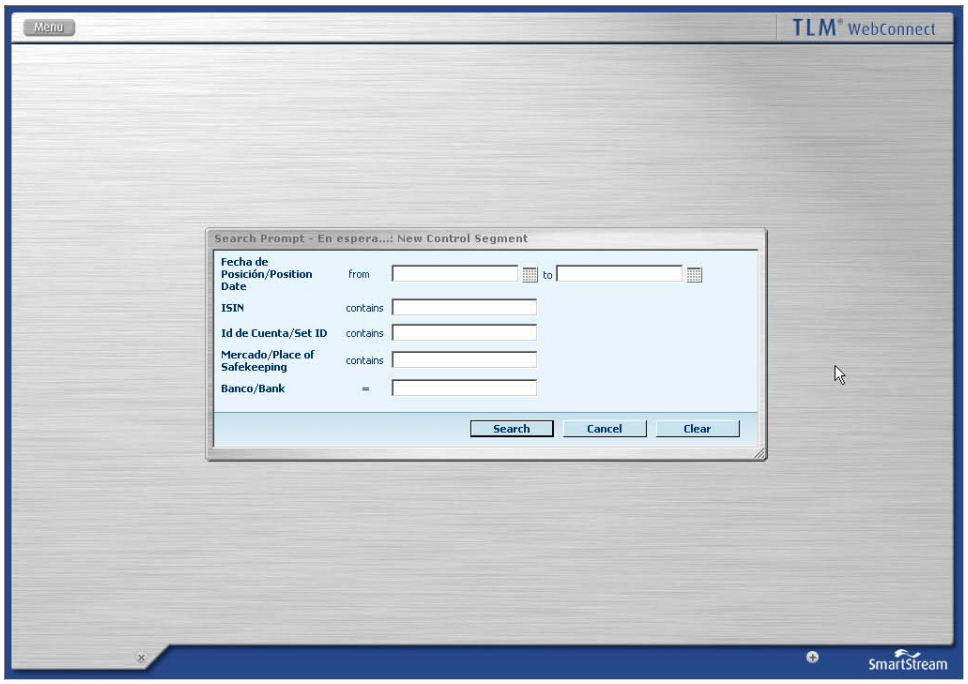
Eliminar la opción de ordenar:

- Para eliminar la ordenación de una columna, seleccionar en el menú desplegable de dicha columna.
- Para volver a las opciones de orden o agrupación definidas por defecto en el informe, seleccionar **Reset All** en el menú desplegable de cualquiera de las columnas.

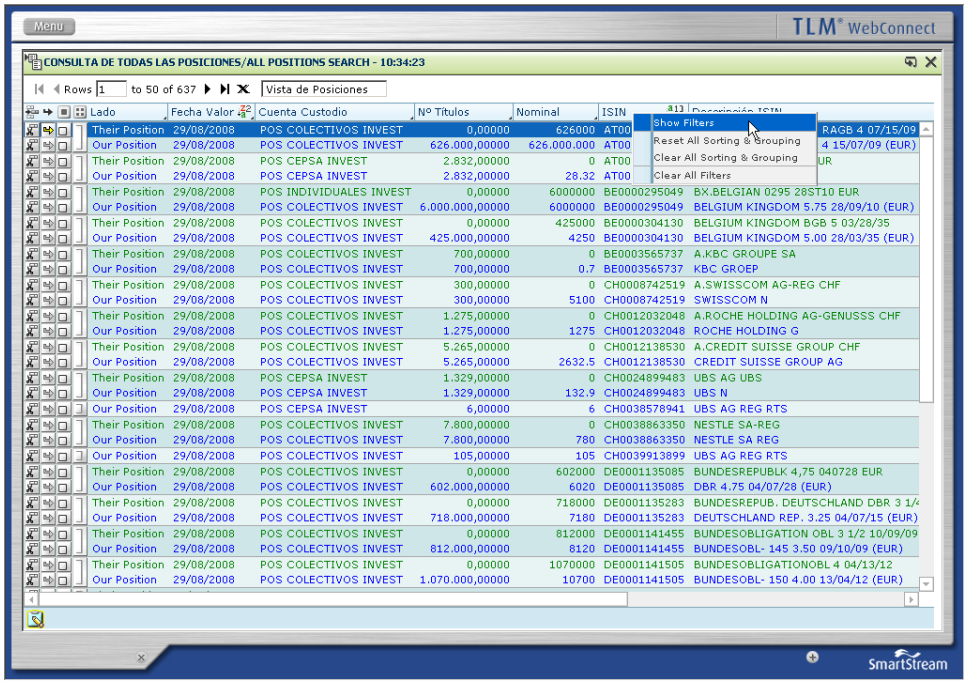
- Para eliminar cualquier selección de orden o agrupación, seleccionar **Clear All** en el menú desplegable de cualquiera de las columnas.

4.3.3.4 Filtrar datos por columnas:

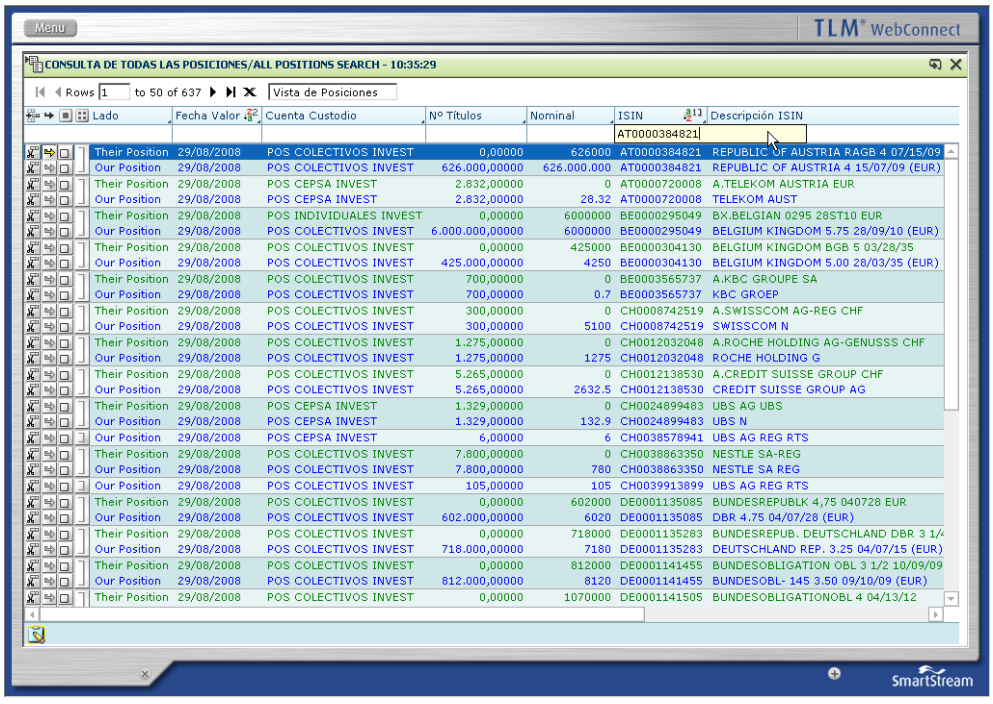
Al ejecutar la mayoría de los informes, se despliega una pantalla donde el usuario puede seleccionar los datos que desea ver en el informe:

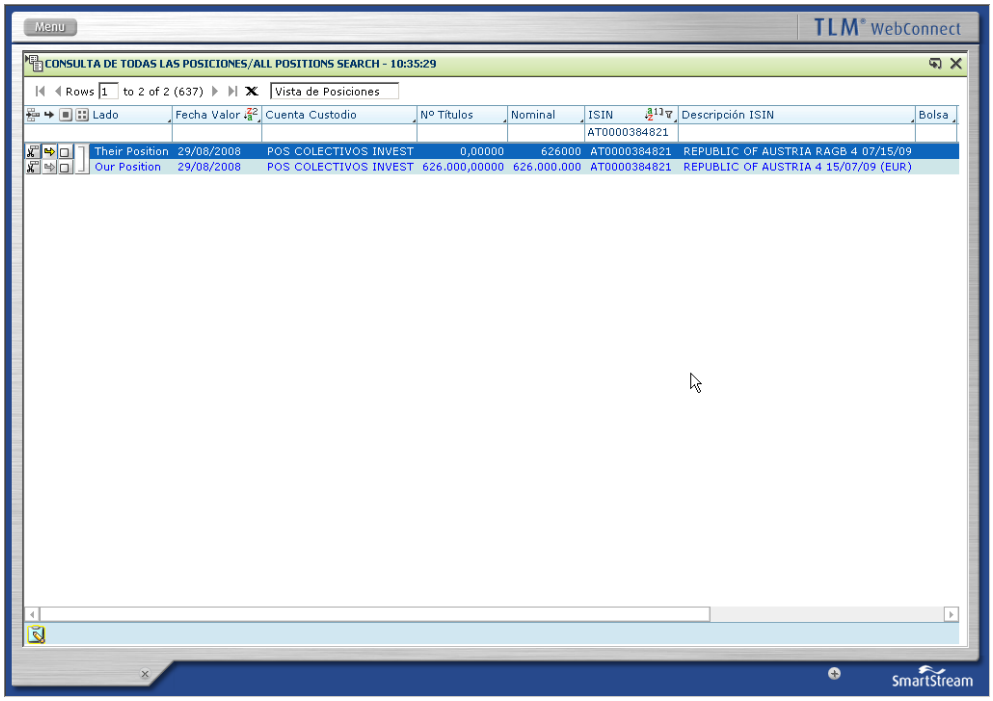


TLM también permite continuar filtrando la información mostrada una vez ejecutado el informe, para ello, hacer clic con el botón derecho en la cabecera de la columna que deseamos filtrar y seleccionar **Show Filters.** Aparecerá una nueva línea en la que se mostrarán, debajo de cada cabecera los filtros aplicados a la misma, si existe alguno.



A continuación, introducir el texto por el que queremos filtrar en la celda correspondiente al campo que deseamos filtrar. Pulsar <Enter> para aprobar el criterio (o <Esc> para cancelar).
Debe tenerse en cuenta que el filtro se activa haciendo clic en cualquier sitio fuera del mismo.





Cuando se está utilizando un filtro, el símbolo aparece en la cabecera de la columna que ha sido filtrada.

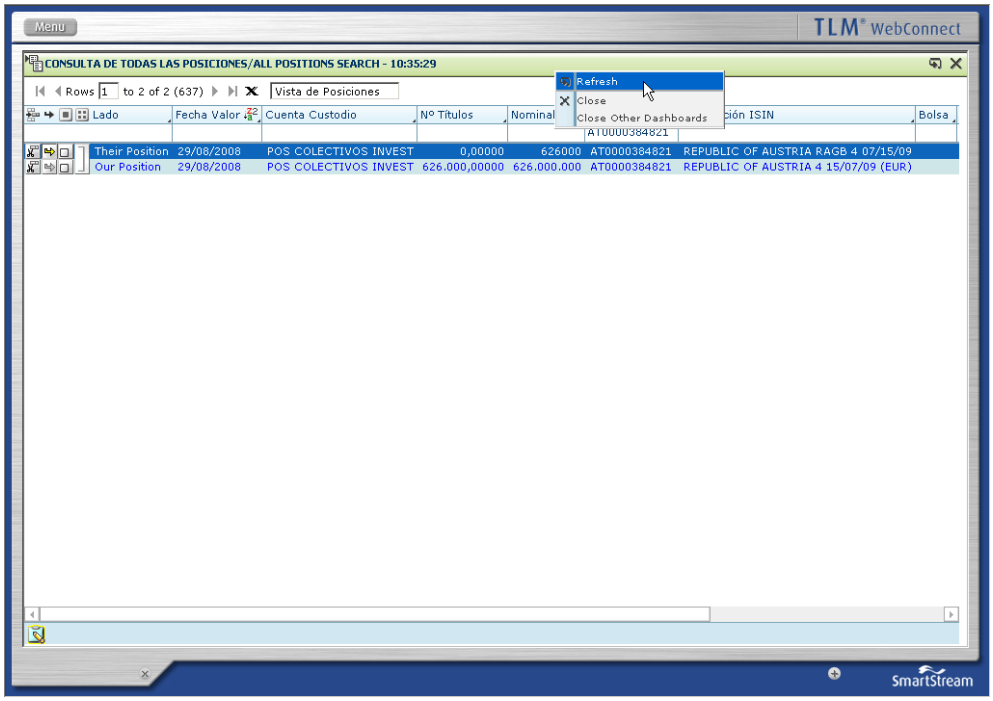
4.3.3.5 Eliminar filtros:

Para eliminar el filtro en una columna, el usuario puede borrar el texto en el filtro de esa columna. Si se desean eliminar todos los filtros, hacer clic con el botón derecho en cualquiera de las columnas o filtros y seleccionar **Clear All Filters.**

4.3.3.6 Actualizar datos:

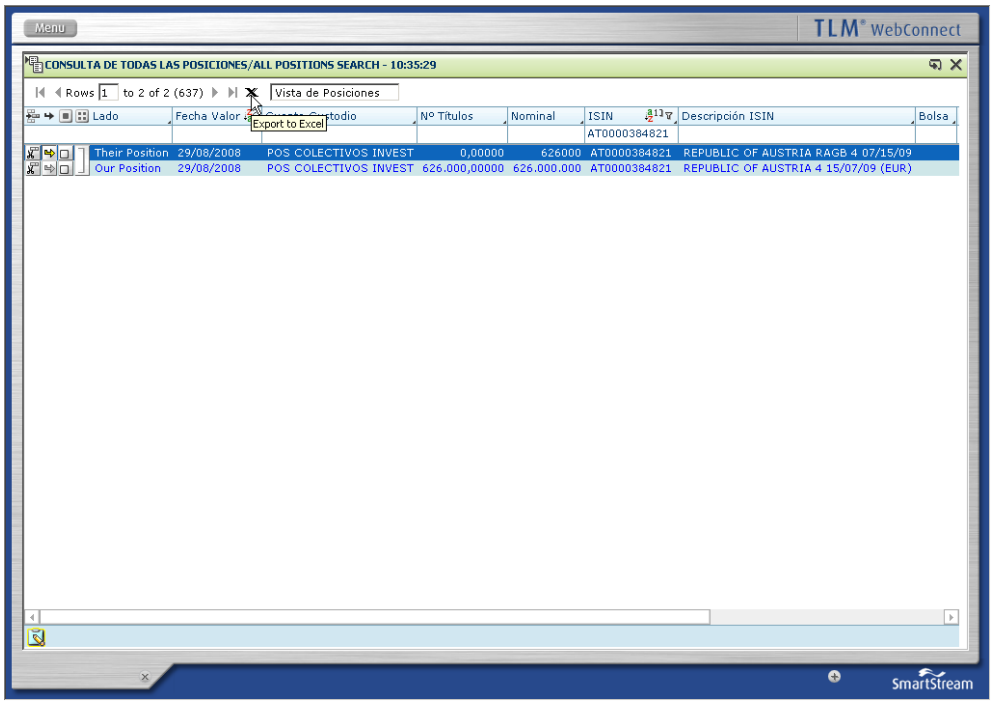
Para hacer una actualización manual se puede:

1- Hacer clic en el icono de actualización, que se encuentra en la parte superior derecha del título del informe.
2- Hacer clic con el botón derecho sobre ésta y seleccionar **Refresh** para actualizar.

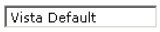


4.3.3.7 Exportar datos a una hoja de Excel:

WebConnect permite exportar los datos seleccionados en una ventana a una hoja de cálculo, para ello, una vez situados en la ventana cuyos datos queremos exportar, pulsamos . Automáticamente se abrirá una hoja Excel que contendrá los datos seleccionados en la ventana.



4.4Vistas :

WebConnect/TLM Recons permite mostrar los datos de cada informe de forma diferente, es decir, los mismos datos pueden mostrarse cambiando el orden de las columnas, agrupando los datos según diferentes criterios, etc.
Estas distintas formas de presentar la información se llaman vistas. Para cada informe existe una vista por defecto, que es la que se muestra al abrir el informe, pero además pueden existir más vistas en cada informe que pueden ser seleccionadas por el usuario, para ello, seleccionar la vista deseada en el desplegable 

que se encuentra en la parte superior izquierda de cada informe.

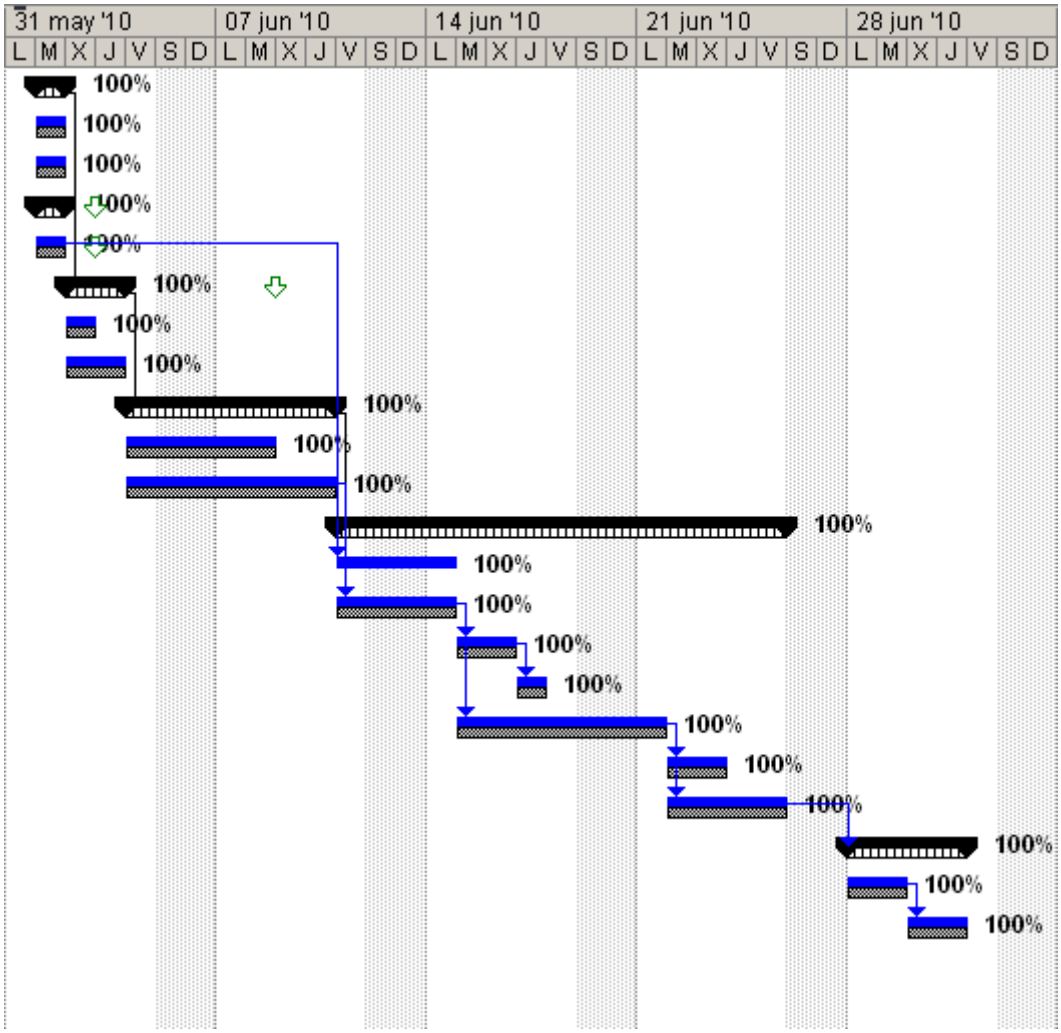
Capitulo V

PLANIFICACION DE PROYECTOS Y COSTES

5.1 Estimación del plan de trabajos en jornadas

El siguiente diagrama muestra las tareas en la elaboración del proyecto, y la estimación del tiempo necesario para el desarrollo de estas.

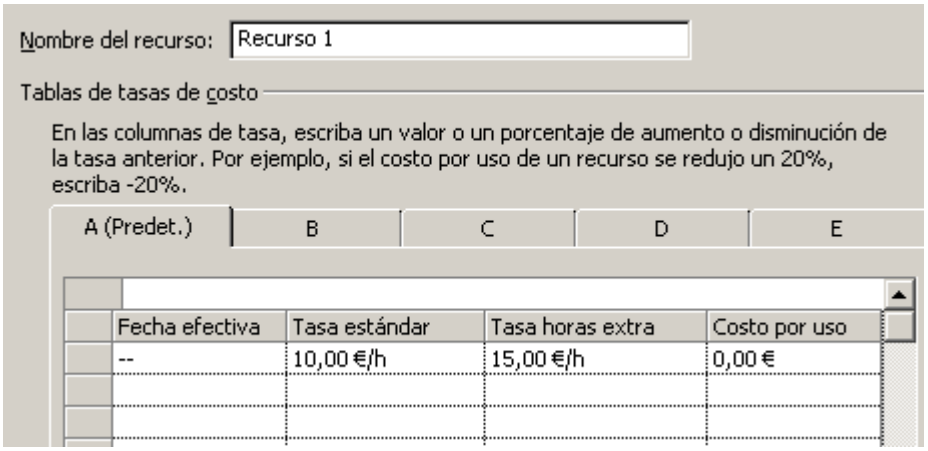
Diagrama de Gant con la duración de las tareas, el porcentaje de realización y la relación de predecisión de las actividades.



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Capitulo 1 - Introducción	**1 día**	**mar 01/06/10**	**mar 01/06/10**		**Recurso 1**
Marco del Proyecto	1 día	mar 01/06/10	mar 01/06/10		Recurso 1
Objetivo	1 día	mar 01/06/10	mar 01/06/10		Recurso 1
Arquitectura del sistema	**1 día**	**mar 01/06/10**	**mar 01/06/10**		
Documentación	1 día	mar 01/06/10	mar 01/06/10		Recurso 1
Capitulo 2 - Planificación del proyecto	**2 días**	**mié 02/06/10**	**jue 03/06/10**	**1**	
Plan de trabajo	1 día	mié 02/06/10	mié 02/06/10		Recurso 1
Estudio de Costes	2 días	mié 02/06/10	jue 03/06/10		Recurso 1
Capitulo 3	**5 días**	**vie 04/06/10**	**jue 10/06/10**	**6**	
Definición de Requisitos	3 días	vie 04/06/10	mar 08/06/10		Recurso 1
Documentacion	5 días	vie 04/06/10	jue 10/06/10		Recurso 1
Capitulo 4 - Metodología del Desarrollo	**11 días**	**vie 11/06/10**	**vie 25/06/10**	**9**	
Documentacion	2 días	vie 11/06/10	lun 14/06/10	5	
Metodoloía Detallada para el desarrollo del proyecto	2 días	vie 11/06/10	lun 14/06/10	11	Recurso 1
Arquitectura Detallada	2 días	mar 15/06/10	mié 16/06/10	14	Recurso 1
Explotación del sistema	1 día	jue 17/06/10	jue 17/06/10	15	Recurso 1
Modelo Entidad Relación	5 días	mar 15/06/10	lun 21/06/10	14	Recurso 1
Diseño Físico	2 días	mar 22/06/10	mié 23/06/10	17	Recurso 1
Elementos de programación	4 días	mar 22/06/10	vie 25/06/10	17	Recurso 1
Capitulo 5 - Experimentación	**4 días**	**lun 28/06/10**	**jue 01/07/10**	**19**	
Experimentación TLM	2 días	lun 28/06/10	mar 29/06/10		Recurso 1
Experimentación SmartSchema	2 días	mié 30/06/10	jue 01/07/10	21	Recurso 1

5.2 Costes del proyecto:

El siguiente cuadro muestra los costes del proyecto asignando un único recurso con el siguiente coste.



Coste total del proyecto:

Nombre de tarea	Costo total	Línea de base	Variación	Real	Restante
Capitulo 1 - Introducción	**320,00 €**	**320,00 €**	**0,00 €**	**0,00 €**	**320,00 €**
Capitulo 2 - Planificación del proyecto	**240,00 €**	**240,00 €**	**0,00 €**	**0,00 €**	**240,00 €**
Capitulo 3	**640,00 €**	**640,00 €**	**0,00 €**	**0,00 €**	**640,00 €**
Capitulo 4 - Metodología del Desarrollo	**1.280,00 €**	**1.280,00 €**	**0,00 €**	**0,00 €**	**1.280,00 €**
Capitulo 5 - Experimentación	**320,00 €**	**320,00 €**	**0,00 €**	**0,00 €**	**320,00 €**

Capitulo VI

CONCLUSIONES Y ACCIONES FUTURAS

6.1 CONCLUSIONES

Los sistemas de conciliaciones financieras, son sistemas que se basan en la carga de datos. Estos datos son extraídos previamente desde las aplicaciones propias de los movimientos bancarios.

Esta información bancaria esta constituida en su mayoría por información sobre precios de distintos mercados.

La finalidad de los sistemas de conciliación financiera es comprobar que todas las aplicaciones están alineadas, y que las operaciones realizadas con estos precios no se desvirtualizan de un sistema a otro. Lo que puede suponer comprar o vender en mercados a precios por debajo o por encima de su valor.

Los sistemas de conciliación financiera se basan en tres puntos principales:

- Carga
- Cruce
- Mostrar la información

La carga en la BBDD se realiza sobre un modelo ER definido. La carga se realiza a través de un elemento de carga propio del sistema de conciliación denominado KB. Que sirve de mapeo entre los valores de las extracciones y la BBDD.

Nuestro sistema de conciliación se subdivide en dos aplicaciones que permiten modelar el diseño en función de las necesidades de carga, cruce y reporting de resultados.

Las dos aplicaciones utilizadas en este proyecto las podríamos definir como la interfaz gráfico entre la base de datos y la aplicación de parametrización, y la propia aplicación de parametrización donde se definirá la casusticas.

El interfaz gráfico permite diseñar la relación entre los campos físicos de la bbdd con los campos lógicos. A su vez podremos dotar de un mayor número de propiedades, según la necesidad, a los campos dentro de la aplicación de parametrización.

La aplicación de parametrización se basa en una sintaxis sencilla, en la que podríamos resumir que su labor es seleccionar la información dentro de la base de datos, agrupar esa información, comprobar que esas agrupaciones cumplen unas casusticas determinadas y marcar esa información a un estado determinado.

Posteriormente se realizará un reporting con la información solicitada por el cliente de negocios.

Este reporting llegara al cliente de dos maneras distintas:

- Vía WEB
- A través del correo con un informe sobre la BBDD

Vía Web permitirá al cliente hacer consultas de filtros sobre la información cargada en la BBDD. Además se podrá interactuar con el estado de los registros, permitiendo conciliar de manera manual.

A través del correo se mostrará un informe definido por el usuario. Al contrario que en el caso WEB no se podrá interactuar con el sistema, aunque obtención de la información estará más automatizada.

A parte de las aplicaciones necesitaremos una serie de elementos complementarios a los sistemas de conciliación, que optimizaran el funcionamiento y permitirán una mayor versatilidad.

En este punto entre los módulos de PL/SQL, como pueden ser vistas que faciliten el manejo de la información requerida por el usuario. Casuísticas más complejas sobre la información que las realizadas por la aplicación de paramtrización.

La arquitectura de los sistemas de conciliación permiten una adaptación total al cliente, disminuyendo en la medida de lo posible los limites de los sistemas de conciliación.

6.2 Acciones Futuras:

Las acciones futuras se basarán en integrar la funcionalidad de los elementos externos complementarios al sistema, dentro de las aplicaciones.

Evitar problemas de concurrencias entre los procesos de carga:

Cuando los procesos de carga gems intentan cargar a la vez sobre una misma cuenta se realizan bloqueos que provocan que los tiempos de carga sean mayores de los esperados.

Evitar limitaciones en los procesos de cruce workflow:

Estos procesos están limitados por el volumen a conciliar. Es decir si para una única cuenta tenemos una Initiation con más de 450.000 registros, dicho servicio se caerá. Actualmente esta limitación ha sido solucionada separando un fichero en distintos ficheros cuyos registros cumplen unas características determinadas.

Evitar Problemas en la carga de Nulos:

Cuando los procesos cargan registros vacíos, en los valores numéricos se introduce un cero. Que podría parecer lo adecuado sin embargo se puede confundir visualmente con un valor cero del campo o un valor nulo.

Evitar entradas en la tabla Audit. Trail:

Cada vez que un grupo de registros cumple unas reglas determinadas, se procede a realizar un acción sobre ellos, en general se les marca un campo denominado estado a un estado específico.

Por tanto se hace un update sobre la base de datos y se realiza una entrada sobre la tabla audit. Trail que marca los eventos por los que los registros están pasando. Cuando marcamos un gran número de elementos podemos llenar esta tabla. Para ellos en la sería interesante tener la opción de desmarcar que se genere una entrada en dicha tabla.

CheckPoint sobre las aplicaciones:

Sería interesante mejorar este aspecto de este sistema, en la ejecución automática necesitamos comprobar que la ejecución se esta realizando adecuadamente y va realizando los pasos especificados

(carga correcta, cruce correcto, informe correcto) comprobando las tablas de los servicios.
Si este sistema devolviera checkpoint de la secuencia de ejecución, se realizarían menos comprobaciones realizando llamadas externas.

Evitar realizar distinciones de diseño sobre los informes entre la parte Web y el cliente pesado:

Actualmente existen diferencias visuales entre el diseño realizado en el cliente pesado y los que mostramos en el cliente ligero. En ocasiones no sé corresponde. Una acción futura sería realizar el diseño directamente sobre el WEB.