Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Una soluzione software per la raccolta ed il trattamento dei dati nell'ambito della GDO

Relazione finale di Tirocinio Lungo

Relatore: Prof. Di Nunzio Giorgio Maria
Laureando: Rampado Roberto (Matricola 542260/IF)

Anno Accademico 2013/2014

Indice

1 Introduzione

La presente relazione e' il frutto di un progetto software sviluppato presso IBC s.r.l. al fine di soddisfare le richieste del committente B (nome completo omesso per motivi di privacy), e riepiloga un lavoro di analisi, progettazione e sviluppo della durata di circa 6 mesi.

IBC s.r.l. opera nel settore della GDO (Grande Distribuzione Organizzata) e fornisce soluzioni hardware e software per la gestione della rete di vendita di svariati clienti, dal “piccolo negozio autonomo” fino alla grande catena.

Il committente B opera nel settore del retail a livello internazionale, con filiali in Italia, Portogallo, Russia e svariate altre nazioni; esso ha deciso di affidarsi ad IBC per la gestione di un nuovo flusso dati centralizzato proveniente dalle sue filiali, il quale necessita principalmente di validazione (per appurare la bonta' del dato) e stoccaggio su base dati persistente (per motivi di storico / statistici).

Per lo sviluppo del progetto ci si e' avvalsi dello strumento di ETL open source Kettle (Pentaho Data Integration), del linguaggio di programmazione Java (con supporto allo sviluppo mediante utilizzo dell'IDE Eclipse), e del database relazionale open source PostgreSQL.

Nel corso della relazione, verranno analizzate nel dettaglio le esigenze e le specifiche fornite dal committente, la motivazione delle scelte effettuate per quanto riguarda metodologia e strumenti di sviluppo, e verranno presentati nello specifico questi ultimi (con particolare enfasi sullo strumento Kettle); in seguito verranno illustrate le varie fasi di evoluzione del progetto, a partire dalla progettazione e sviluppo della base di dati e dell'ETL vero e proprio (il “cuore pulsante” del progetto), per poi passare alla realizzazione delle maschere web per l'utente “tecnico”, verra' quindi accennato lo svolgimento dell'installazione e dei test del software, ed infine verranno effettuate alcune considerazioni sui possibili margini di miglioramento e sul ciclo di vita del software.

2 Le esigenze del committente e l'analisi dei requisiti

In questo capitolo verra' innanzitutto effettuata una panoramica sull'azienda del committente e sul suo modello di business, verra' esplicata la natura dei dati coinvolti nel trattamento previsto dal software, ed in seguito verranno riepilogati i requisiti funzionali (il comportamento che deve avere il software) raccolti durante i vari incontri con il committente, nonche' verra' ripercorso il processo di analisi dei requisiti tecnici (le necessita' in termini di risorse da mettere a disposizione) per la realizzazione del progetto.

2.1 Background

L'azienda del committente B si occupa principalmente della vendita di capi di vestiario, e possiede una rete che conta circa 6.500 negozi in piu' di 120 paesi, per un fatturato di oltre 2 miliardi di euro l'anno.

Una peculiarita' della rete di vendita del committente e' il fatto di possedere solamente una manciata di negozi a gestione diretta, mentre per i restanti viene utilizzata una formula di franchising, ovverosia un sistema di affiliazione commerciale che prevede la gestione del punto vendita da parte del proprietario del negozio, al quale viene concesso il diritto di commercializzare prodotti dell'azienda "madre", in cambio di una percentuale sul fatturato.

Sorvolando sui vantaggi di tale sistema, uno dei principali svantaggi e' il fatto che, adottando tale modello di business, l'azienda "madre" ha poca visibilita' (e quindi anche poco controllo) sui negozi che vendono i loro prodotti al cliente finale (l'acquirente dei capi di vestiario); per cercare di ovviare a tale problema, il committente B si e' rivolto ad IBC per lo sviluppo di un software che permetta la gestione e l'analisi di vari flussi dati provenienti dai vari negozi, ed in particolar modo per quanto riguarda:

- Vendite

- Movimenti
- Giacenze

Nel presente documento l'attenzione verra' focalizzata principalmente sulla gestione lato sede del flusso dati, mentre la gestione del flusso dal negozio alla sede non sara' oggetto di discussione (seppur verra' brevemente accennata nel corso dei prossimi capitoli).

2.2 I dati di interesse

Come accennato nel paragrafo precedente, i dati raccolti dai punti vendita che sono di interesse principale per il progetto sono tre, ed in particolare: Vendite, Movimenti e Giacenze.

Andiamo ora ad approfondire meglio il significato ed il motivo di interesse da parte del committente per ognuno di essi:

2.2.1 Vendite

Le vendite rappresentano il termine del ciclo di vita dei prodotti all'interno della rete di vendita, e permettono di quantificare gli introiti dei vari negozi (e quindi conseguentemente dell'azienda), oltre ad avere l'importanza fondamentale di permettere all'azienda, mediante la loro analisi, di capire l'andamento del mercato, e quindi regolare la produzione di prodotto in base alla richiesta (legge della domanda e dell'offerta).

Le vendite sono strutturate logicamente secondo una relazione di *master/detail*, ovverosia e' presente un entita' “padre” (master), in questo caso la testata dello scontrino (rappresentante l'intera transazione di vendita), ed un insieme di entita' “figlie” (detail), in questo caso i vari dettagli di vendita dei singoli prodotti afferenti al medesimo scontrino / alla stessa testata.

Il livello di dettaglio di tali informazioni riesce a caratterizzare elementi della granularita' pari alla vendita del singolo articolo all'interno di una singola transazione di vendita che avviene in un determinato negozio ed in un dato giorno.

2.2.2 Movimenti

I movimenti rappresentano gli scambi di merce che coinvolgono due nodi della rete di vendita, e si suddividono principalmente in due categorie:

- Movimenti interni: si tratta dei movimenti che avvengono strettamente fra negozi appartenenti alla rete di vendita (quindi coinvolgono solamente entita' ben definite e conosciute)

- Movimenti esterni: si tratta dei movimenti che avvengono fra negozi appartenenti alla rete di vendita ed entita' esterne (come ad esempio fornitori)

La conoscenza dell'entita' di merce che viene scambiata fra i vari negozi permette di stabilire, alla pari delle vendite, la necessita' di determinati prodotti da parte dei negozi, e quindi la domanda che si ha in determinate locazioni; tale conoscenza permette inoltre di ottimizzare la distribuzione dei prodotti ai vari negozi, determinando sin da subito le locazioni precise in cui si ha la necessita' dei prodotti.

Il livello di dettaglio di tali informazioni riesce a caratterizzare elementi della granularita' del singolo prodotto movimentato, nel contesto di un determinato documento di movimentazione (e.g. Bolla), il quale coinvolge due entita' (negozi della rete di vendita oppure negozio ed entita' esterna) in un dato giorno.

2.2.3 Giacenze

Le giacenze rappresentano, per ogni prodotto presente nel magazzino del negozio, la quantita' di merce depositata ed in attesa di essere venduta oppure movimentata (si tratta dunque di una sorta di inventario istantaneo del negozio).

Le giacenze sono di notevole importanza, poiche' permettono di capire la quantita' di merce invenduta presente nei negozi, nonche' di prevedere la necessita' di approvvigionamento di nuovi prodotti da parte dei negozi; inoltre, esse permettono di stabilire se e' possibile ottimizzare la distribuzione della merce fra i negozi, ad esempio spostando determinati prodotti presenti in giacenza e da lungo tempo invenduti in un negozio, verso un altro negozio che invece ha necessita' di tali prodotti, in quanto la sua zona presenta una forte domanda per quest'ultimi.

Il livello di dettaglio di tali informazioni riesce a caratterizzare elementi della granularita' del singolo prodotto depositato nel singolo negozio in un determinato giorno.

2.2.4 La quadratura dei dati

Particolare interesse va' posta sulla richiesta di “quadratura” dei dati; con tale termine si fa' riferimento all'equazione che mette in relazione le quantificazioni dei tre flussi sopra esposti, ovverosia:

$$\text{GIACENZA} = \text{VENDITE} + \text{MOVIMENTI}$$

La relazione sopra stabilita implica il fatto che, in un dato istante temporale, la quantita' in giacenza di un determinato prodotto sia pari alla somma di tutti i movimenti (in entrata ed in uscita dal negozio) e di tutte le vendite di tale prodotto fino all'istante temporale in esame.

2.2.5 Un elemento comune: L'Articolo

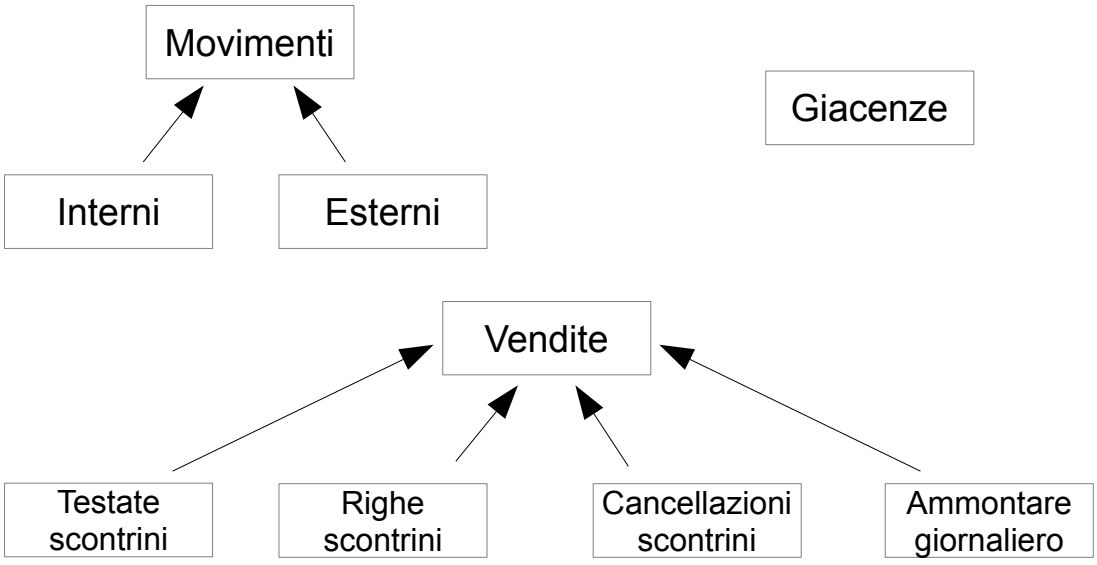
Tutte le tipologie di dati presentate hanno un elemento comune, il quale e' l'oggetto principale coinvolto nell'operazione identificata dal particolare tipo di dato (vendita/movimento/giacenza), ed e' l'Articolo; per quanto riguarda il committente B, ogni Articolo e' identificato univocamente dalla combinazione di 4 attributi:

- Style (stile, identificato da un codice alfanumerico)
- Collection (collezione, es. autunno 2013 / A13 o primavera 2014 / P14)
- Size (dimensione, es. S / M / L / XL, oppure 52 / 54)
- Color (colore, es. blu notte, indaco, etc)

Nei paragrafi seguenti, quanto verra' indicato l'Articolo, si fara' sempre riferimento alla sua composizione in 4 attributi sopra esposta.

2.3 La struttura del flusso

Il *container* (contenitore) scelto dal committente B per l'immagazzinamento ed il trasporto dei dati sopra elencati dal punto vendita alla sede e' il formato XML, accompagnato da corrispondente documento di schema XSD (quest'ultimo verra' meglio approfondito in seguito); procediamo innanzitutto con una prima caratterizzazione del dettaglio dei dati coinvolti:



2.3.1 Movimenti

I movimenti vengono sempre espressi con riferimento principale al negozio destinatario, e quindi vengono caratterizzati nelle due derivazioni, interni ed esterni,

a seconda del negozio sorgente, il quale puo' essere, per l'appunto, un negozio interno alla rete di vendita nel primo caso, oppure un ente esterno alla rete nel secondo caso.

Le informazioni principali contenute nei movimenti sono le seguenti:

- Data contabile di riferimento
- Data di consegna
- Punto vendita sorgente (interno o esterno alla rete di vendita)
- Punto vendita destinazione (negozio che ha generato l'XML)
- Articolo movimentato (style/collection/size/color)
- Quantita' movimentata
- Documento di movimento (opzionale)
- Altre informazioni secondarie

2.3.2 Giacenze

La struttura delle giacenze e' molto semplice; le informazioni in esse contenute sono le seguenti:

- Data contabile di riferimento
- Punto vendita
- Articolo (style/collection/size/color)
- Quantita' in giacenza

2.3.3 Vendite

Come gia' indicato, le vendite sono organizzate secondo una struttura di *master/detail*, per cui si ha una testata, la quale identifica lo scontrino / la singola transazione di vendita, ed una lista di dettagli, rappresentanti i singoli articoli venduti; si hanno inoltre due strutture parallele, le cancellazioni e l'ammontare giornaliero, che servono semplicemente ad annullare una determinata transazione di vendita nel primo caso (la quale non andra' in seguito considerata valida), e per riepilogare a livello “aggregato” la somma dell'ammontare di vendita della giornata per un determinato negozio nel secondo caso.

Le informazioni contenute nelle testate di vendita sono le seguenti:

- Data contabile (data di vendita)
- Negozio di emissione scontrino
- Id scontrino (numero transazione)

- Totale scontrino
- Totale ammontare iva applicata
- Totale ammontare sconti applicati
- Altre informazioni secondarie

In ogni riga dello scontrino sono contenute invece le seguenti informazioni:

- Riferimento alla testata scontrino (data/negozio/id scontrino)
- Articolo venduto (style/collection/size/color)
- Quantita' articolo venduta
- Valore di vendita
- Ammontare iva applicata
- Ammontare sconto applicato
- Numero riga
- Altre informazioni secondarie

Passando invece alle cancellazioni degli scontrini, abbiamo una struttura composta semplicemente dai riferimenti alla testata scontrino da annullare, ossia:

- Data scontrino
- Negozio
- Id scontrino

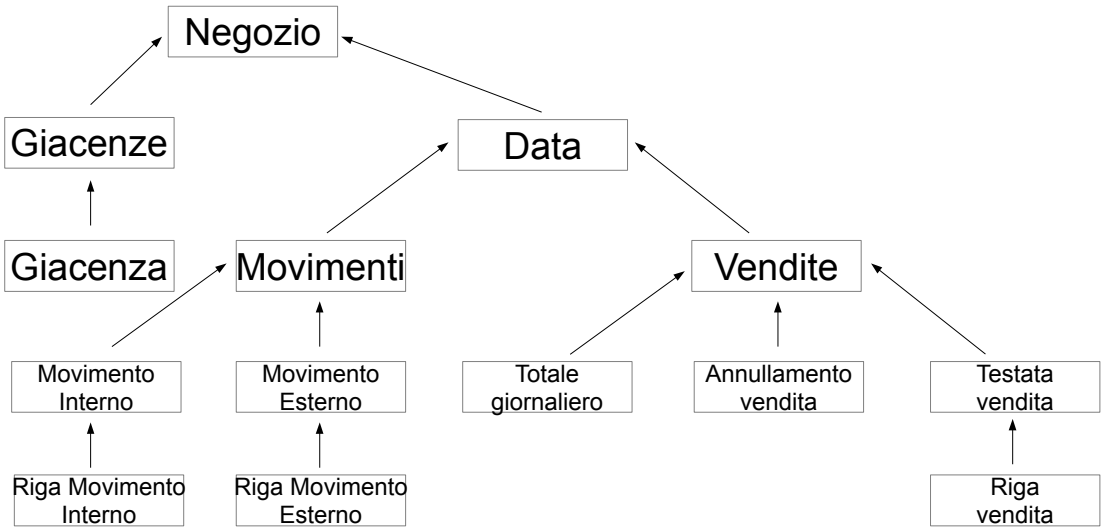
Per concludere la panoramica sui dati relativi alle vendite, l'ultima struttura ad esse collegata, l'ammontare giornaliero, e' composta da:

- Data contabile
- Negozio
- Totale dell'ammontare di vendita

2.4 La struttura dei dati all'interno dell'XML

Ogni file XML e' studiato per contenere dati relativi ad un solo punto vendita, quindi il nodo principale (radice) dell'xml serve per l'appunto ad indicare il negozio di competenza, e di conseguenza il resto delle informazioni contenute nel file sono gerarchicamente “sotto” di esso.

La struttura dell'XML e' definita mediante un file di schema XSD, il quale stabilisce la gerarchia di seguito schematizzata:



Provvediamo quindi a fornire una spiegazione maggiormente dettagliata dei vincoli presenti nella struttura, della cardinalita' delle relazioni, e delle caratteristiche di alcuni nodi di particolare interesse:

- *Nodo "Negozio"*: e' obbligatorio ed univoco all'interno dell'XML; identifica il negozio di riferimento, il quale viene utilizzato da tutte le sotto-entita'
- *Nodo "Giacenze"*: e' obbligatorio e deve avere cardinalita' 1; in esso viene indicata la data di riferimento per tutte le giacenze sotto riportate (le giacenze possono avere solamente una data di riferimento per file XML)
- *Nodo "Giacenza"*: rappresenta la singola giacenza (quindi e' riferita ad un singolo articolo); possono esser presenti da 1 ad N nodi giacenza, ognuno dei quali deve riportare un articolo di riferimento differente da quello degli altri nodi (l'articolo e' in chiave)
- *Nodo "Data"*: rappresenta un giorno contabile, puo' essere presente con cardinalita' da 0 a n al di sotto del nodo "Negozio" e viene utilizzata come data di riferimento per tutti i movimenti o le vendite sotto riportate
- *Nodo "Movimenti"*: serve solamente a raggruppare logicamente tutti i sotto nodi, e differenziarli dagli altri che sono sotto al modo "Vendite"; dev'essere univoco all'interno del nodo "Data" (quindi ha cardinalita' da 0 a 1, e possono esser presenti al piu' N nodi di questo tipo, con N pari al numero di nodi "Data" complessivamente presenti nell'XML)
- *Nodo "Movimento Interno / Esterno"*: rappresenta un singolo movimento (inteso come blocco di articoli movimentati nella stessa spedizione) mediante la specifica del negozio sorgente e di un eventuale codice documento di riferimento, e raggruppa tutte le righe di movimento sotto di esso (ha la stessa

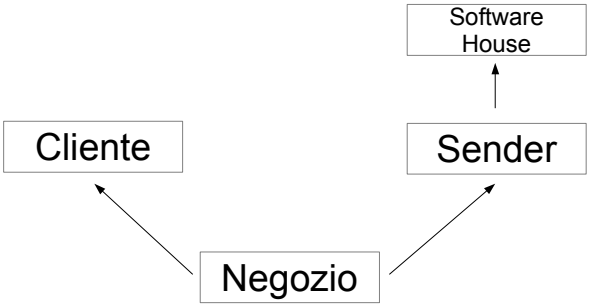
funzione della testata di vendita per le righe di vendita); ha cardinalita' da 0 a N

- *Nodo "Riga Movimento Interno / Esterno"*: rappresenta il singolo articolo movimentato all'interno del movimento complessivo indicato nel nodo padre; ha cardinalita' da 1 a N, ed inoltre e' univoco per data e per coppia di negozio sorgente / negozio destinazione (lo stesso articolo non puo' essere ripetuto a parita' di nodi padre)
- *Nodo "Vendite"*: serve solamente a raggruppare logicamente tutti i sotto nodi, e differenziarli dagli altri che sono sotto al modo "movimenti"; segue la stessa logica del nodo "movimenti", quindi ha cardinalita' da 0 a 1 per ogni nodo "Data"
- *Nodo "Totale Giornaliero"*: rappresenta l'ammontare complessivo incassato nel giorno/negozio, e logicamente dev'essere pari alla somma dei totali netti riportati nelle testate di vendita; ha cardinalita' da 0 a 1 per ogni nodo "Data"
- *Nodo "Annullamento Vendita"*: permette l'annullamento di una testata di vendita (comprensiva di tutte le sue righe), referenziata implicitamente mediante data/negozio dei nodi padre, ed esplicitamente mediante l'identificativo riportato nel nodo di annullamento; ha cardinalita' da 0 a N
- *Nodo "Testata Vendita"*: rappresenta un singolo scontrino di vendita, con riferimento al giorno/negozio indicato nei nodi padre; e' presente con cardinalita' da 0 a N, e seppur non specificato nello schema XSD, presenta dei vincoli logici aggiuntivi sui nodi figli, in particolar modo la somma dei valori di vendita delle righe di cui e' composto dev'essere uguale al valore totale di vendita riportato in testata, al netto di eventuali sconti applicati, inoltre deve rispettare il vincolo di univocita' dell'id scontrino all'interno del giorno / negozio
- *Nodo "Riga Vendita"*: rappresenta la vendita di un singolo articolo nella transazione; e' presente come sotto elemento di una testata di vendita con cardinalita' da 1 a N

2.5 Le entita' anagrafiche: Negozi, Clienti, Senders e Software Houses

Con riferimento alle strutture dati sopra esposte, si puo' notare che l'elemento predominante, da cui poi dipendono tutti gli altri, e' il negozio.

Il negozio, pero', presenta a sua volta delle dipendenze da altre entita', definite a livello anagrafico, con le quali ha le seguenti relazioni:



Come si puo' vedere dallo schema, abbiamo che un negozio appartiene sempre ad un cliente (con questo termine, si intende un affiliato del committente B), ed allo stesso tempo e' collegato ad un sender, il quale e' a sua volta dipendente da una software house.

Tali entita', con le loro relazioni, servono per stabilire una serie di regole di business stabilite dal committente B, e che verranno in seguito meglio approfondite; per il momento, preoccupiamoci solamente di capire bene il significato di queste nuove entita' appena introdotte, e del modo in cui sono relazionate fra di loro; dato che abbiamo gia' chiarito il significato dell'entita' negozio nei precedenti paragrafi, ci concentreremo solamente su clienti, senders e software houses:

2.5.1 Cliente

Il cliente, come entita', rappresenta un affiliato del committente B, e puo' possedere un singolo negozio, oppure una rete piu' o meno complessa di negozi da esso gestiti direttamente.

Ogni cliente e' libero di affidarsi ad una o piu' software house per la gestione della rete informatica che copre i suoi punti vendita, quindi, per quanto di interesse al progetto, e' lui a scegliere la soluzione software da utilizzare per la generazione dei files XML.

2.5.2 Sender

Il sender rappresenta l'entita' fisica che si occupa della spedizione dei dati verso la sede; esso puo' identificare un singolo negozio (in tal caso solitamente si ha che il software che genera gli XML e li spedisce in sede e' installato direttamente su un server fisico presente presso il punto vendita stesso) oppure un insieme di negozi che effettuano le trasmissioni dei dati mediante esso (quindi in tal caso solitamente si ha un server centralizzato che gestisce i vari punti vendita ad esso collegato).

2.5.3 Software House

La software house rappresenta l'entita' che e' responsabile della creazione fisica dei files XML; essa puo' avere uno o piu' senders (i quali, in termini pratici, rappresentano le installazioni “fisiche” del software che si occupa della generazione dei files XML, ognuna delle quali e' codificata diversamente dalle altre), ed ognuno di essi gestisce il flusso dati proveniente da 1 ad N negozi.

2.5.4 Il flusso anagrafico complessivo

I dati sopra indicati vengono forniti ad IBC direttamente dal committente B, mediante un flusso dati in formato CSV che viene alimentato giornalmente a partire dal loro sistema di Data Warehouse; cosi' facendo, la situazione anagrafica di riferimento per il software di ETL verra' sempre mantenuta aggiornata, in modo tale da permettere logiche di controllo sui dati ricevuti (tipicamente controlli di consistenza sulle combinazioni di negozi / clienti / sender / software house ricevuti direttamente come informazioni contenute nel naming dei files).

2.6 I requisiti funzionali

In questo paragrafo verranno presentati i requisiti funzionali stabiliti dal committente, ovverosia le modalita' operative che dovra' seguire il software nel processare i dati; vengono di seguito elencati per capi sommari i passaggi a cui dovranno esser sottoposti i dati nel flusso di processamento da parte dell'ETL di elaborazione, i quali verranno in seguito analizzati con maggior dettaglio:

1. Presa in carico files ZIP da cartella FTP di deposito
2. Controllo nomenclatura files secondo standard definito
3. De-compressione files ZIP e controllo integrita'/dimensione files contenuti
4. Validazione files mediante xsd, intercettazione e classificazione errori
5. Se i files non contengono errori definiti come “bloccanti”, vengono estratti i dati in essi contenuti
6. Integrazione delle informazioni relative agli errori “non bloccanti” attribuibili ai dati estratti
7. Analisi dei dati estratti, con applicazione delle regole di business stabilite dal committente
8. Esportazione dei dati in formato csv, al netto di tutte le informazioni aggiuntive relative agli eventuali errori ad essi collegati
9. Compressione dei files in formato tag.gz
10. Invio dei file compressi tar.gz al committente B

Andiamo ora ad analizzare nel dettaglio le singole operazioni che dovra' eseguire il programma:

2.6.1 Presa in carico da FTP

I files XML contenenti i dati di interesse verranno depositati su di un server FTPS da parte di un software sviluppato da IBC ed installato nei server dei vari clienti, il quale monitorizza una cartella ben definita nell'installazione, e provvede a comprimere in formato ZIP gli XML, per poi inviarli in sede in modo sicuro; il software di ETL in sede dovra' quindi preoccuparsi innanzitutto di prendere in carico i files presenti sul server FTP, ed effettuare in seguito una serie di controlli su di essi prima di procedere con l'elaborazione.

Per gestire la problematica dei files parziali (errata presa in carico di files ancora in fase di trasferimento), si e' deciso di associare ad ogni file ZIP, un file vuoto avente lo stesso nome ed estensione .ok, e quindi procedere effettuando innanzitutto la trasmissione del file ZIP, e solo in seguito alla creazione del file .ok, cosi' da permettere il riconoscimento del termine delle trasmissione anche lato sede (il file .ok funge come una sorta di “terminatore di trasmissione”).

2.6.2 Controllo nomenclatura files

E' stata stabilita una specifica nomenclatura che dovranno rispettare i files XML, la quale va' controllata subito dopo la presa in carico di ogni file, in modo da stabilire se e' possibile procedere con le elaborazioni o meno; nella fattispecie, il naming che dovra' rispettare ogni file dovra' essere il seguente:

SSS_nnnnnn_yyyyMMdd_HHmmss.zip

- *SSS* e' il codice della Software House (la lunghezza sara' da un minimo di 3 ad un massimo di 10 caratteri)
- “_” e' il carattere separatore
- *nnnnnn* e' il codice negozio nella codifica del committente B
- “_” e' il carattere separatore
- *yyyMMdd_HHmmss* e' il timestamp di scrittura del file

In aggiunta alla specifica sul naming, sara' necessario controllare, sulla base delle anagrafiche di negozi/clienti/senders/software houses ricevute giornalmente, che la combinazione di software house / negozio specificata sul nome del file sia valida (entrambi devono essere presenti in anagrafica e devono essere legati fra di loro mediante un sender).

2.6.3 De-compressione files ZIP e controllo integrita'/dimensione

In seguito i files ZIP andranno scompressi, e bisognera' controllare l'integrita' del contenuto, in particolar modo:

- Il contenuto non dev'essere danneggiato o illeggibile
- Ogni file ZIP dovra' contenere UNO ED UN SOLO FILE XML, avente naming coerente con quello del file ZIP che lo contiene

Inoltre bisognera' controllare che la dimensione del file XML scompresso non superi i 50 MB (valore stabilito sulla base di test empirici, a partire dal quale la lettura del contenuto dell'XML inizia a soffrire di problemi di performances).

2.6.4 Validazione XML ed intercettazione errori

Ogni file XML andra' poi validato mediante l'XSD fornito, e tutti gli errori di validazione andranno intercettati ed analizzati, al fine di stabilire se rappresentano errori “bloccanti”; in base alla presenza o meno di quest'ultima categoria di errori, si decide se procedere con l'elaborazione, oppure se fermarsi.

Ogni errore va' inoltre categorizzato per tipo flusso (vendite/movimenti/giacenze), ed andra' riportato il riferimento alla “foglia” dell'xml che l'ha generato (in modo tale da permettere in seguito il legame fra foglia ed errore).

Tutti gli errori, sia i “bloccanti” che i “non bloccanti”, sono stati censiti in una tabella fornita dal committente B, al fine di permetterne la codifica secondo lo standard richiesto; di default, tutti gli errori non classificabili in una delle due categorie, vanno gestiti come errori “bloccanti generici / non codificati” e devono portare al blocco dell'elaborazione del file, il quale verra' scartato con errore.

2.6.5 Estrazione dati

Per ogni file XML, bisognera' procedere con l'estrazione dei dati in essi contenuti, ed allo stoccaggio persistente su database, al fine di rendere i dati disponibili alla consultazione da parte dell'utente web.

Ogni flusso dati contenuto nell'XML andra' estratto con una diversa iterazione del file, al fine di inserire i dati in tabelle diversificate per tipo flusso; particolare attenzione va' posta all'estrazione dei dati riguardanti le giacenze, in quanto, come indicato nei paragrafi precedenti, tali informazioni sono un requisito essenziale, ed in loro mancanza e' necessario scartare il file, indicando il codice di errore relativo alla mancanza di giacenze.

2.6.6 Integrazione degli errori "non bloccanti"

Tutti gli errori “non bloccanti” rilevati nella fase di validazione andranno integrati sui dati al livello di dettaglio di pertinenza (questo solo se relativi ad una singola foglia, altrimenti andranno indicati a livello generico di file); tale integrazione e' necessaria al fine di poterli visualizzare nelle maschere di consultazione web, ed al fine di poterli esportare assieme ai dati nei files csv di output.

In questa fase avviene la trascodifica degli errori secondo la tabella fornita dal committente B: ogni errore rilevato dal validatore e da esso codificato secondo il suo standard, andra' mappato secondo la codifica definita nella tabella (quindi, ad esempio, si passa da un errore “cvc-datatype-valid.1.2.1” sull'attributo “ora_scontrino” del flusso vendite, ad un codice errore “10” da associare alla foglia dell'XML che l'ha generato).

2.6.7 Applicazione delle regole di business

Oltre agli errori “non bloccanti” rilevati in fase di validazione ed in seguito integrati con i dati, andranno inoltre applicate delle regole di business stabilite dal committente al fine di verificare la consistenza dei dati elaborati, e tutte le eventuali discrepanze andranno integrate nei dati, sempre al fine di poterle visualizzare lato web e di poterle esportare nei files csv di output.

Le regole di business vengono applicate in questa fase poiche' si tratta di vincoli da applicare all'intero insieme di dati estratti e non sul singolo elemento, quindi e' necessario aver prima estratto tutti i dati, e solo in seguito e' possibile valutare le regole definite.

2.6.8 Esportazione dei dati in formato csv

I dati estratti andranno poi esportati in formato csv, al netto di eventuali errori di validazione “non bloccanti” e di eventuali errori dovuti alle regole di business precedentemente applicate; la cardinalita' dei files csv dovra' rispecchiare quella dei files XML sorgenti, in particolar modo creando un file csv per ogni file XML processato, contenente tutte le tipologie di dato estratte dal file sorgente, identificabili da un carattere di tipo record che precedera' ogni riga del file csv, ed avente nomenclatura uguale al nome file sorgente, con l'aggiunta di un suffisso identificante l'instante di generazione del csv, ed ovviamente l'estensione .csv.

2.6.9 Compressione in formato tar.gz

Al termine dell'elaborazione, tutti i files csv creati andranno compressi in un unico file tar.gz, il quale dovra' presentare nel nome l'istante di generazione dello stesso, e dovra' essere identico al timestamp di generazione dei files csv in essi contenuti, riportato come suffisso su ogni file.

Ogni file compresso tar.gz dovra' contenere solamente un file per negozio, in particolar modo solamente il file piu' “vecchio” (con istante di generazione minore) di ogni negozio presente nel giro di elaborazione; questo requisito serve per poter garantire la sequenzialita' dell'elaborazione dei dati per negozio da parte del sistema di Data Warehouse del committente B (in particolar modo per gestire correttamente le cancellazioni, sia esplicite mediante invio di una cancellazione di vendita, sia implicite mediante il reinvio dello scontrino o del movimento).

2.6.10 Invio dei tar.gz al committente B

Dopo aver creato il file tar.gz, bisognera' provvedere ad inviare il file al committente B; per la comunicazione fra i sistemi, il committente B ha deciso di avvalersi di un tool chiamato XFB, il quale verra' installato sui server IBC e provvedera' automaticamente a trasferire i files depositati si di una cartella definita di comune accordo, che verra' da esso monitorata.

L'invio del file si riduce quindi semplicemente al deposito su di una cartella su file system, il trasporto viene gestito automaticamente dal tool XFB.

2.7 I requisiti tecnici

Dal punto di vista tecnico, e' stata effettuata un analisi del carico del sistema a regime, il quale dovra' sopportare l'elaborazione dei dati inviati da parte di circa 4000 senders, i quali effettueranno una o piu' trasmissioni al giorno, con maggior afflusso in orario serale, dopo la chiusura del punto vendita, fra le ore 00:00 e le ore 03:00, e dovra' elaborare i dati entro le ore 07:00 di mattina, in modo tale da renderli disponibili al reparto marketing e business del committente B sin dalla prima mattinata; la stima e' stata effettuata sviluppando un pseudo-programma che si occupi della sola validazione ed estrazione dei dati dai files in modo molto “grezzo”, e simulando l'applicazione di alcune logiche di business ritenute “credibili” su dei dati di esempio forniti dal committente B, oltre a calcolare i tempi macchina per spostamento e de-compressione dei files sulla base di una simulazione in ambiente di test (utilizzando un sistema server con specifiche hardware intermedie, in modo da lasciar spazio ad eventuali upgrade futuri, se necessario).

Oltre ai requisiti temporali dell'elaborazione, si e' stimato anche l'ingombro dei dati su file system (in termini di files XML mantenuti come storico cautelativo), su FTP e su database, per il periodo di retention stabilito, pari a 2 anni (2 settimane solamente per i files XML su file system); inoltre e' stata eseguita una stima di occupazione di banda nei momenti di maggior afflusso, ovverosia al momento della chiusura dei punti vendita nella zona con maggior densita' (l'Italia).

Per tutte le stime si e' applicata una logica di worst-case, quindi pensando sempre al caso peggiore, in cui l'ingombro dei files sia massimo, ed essi vengano spediti tutti assieme e di conseguenza elaborati nello stesso periodo temporale; inoltre in questa

prima fase di prototipo si e' implementata una logica di elaborazione per lo piu' seriale, in modo da lasciar spazio ad eventuali ottimizzazioni future mediante parallelizzazione dei processi.

Vengono di seguito riassunti i requisiti stimati sulla base di quanto sopra esposto, e proiettati su un periodo pari a due anni:

- *Storage FTP*: dato che l'FTP dovra' mantenere solamente i dati di una giornata (a meno di imprevisti nelle elaborazioni), si e' stimato che lo spazio richiesto sia pari a circa 1 GB (sovrastimato)
- *Database Storage*: considerando che molte ridondanze tipiche dei files XML vengono eliminate in fase di estrazione dati e caricamento su database, anche grazie all'utilizzo di strutture normalizzate, e considerando anche lo spazio aggiuntivo occupato dagli indici che verranno creati sui dati, si e' stimato che l'occupazione di spazio su database sia pari a circa 500 GB
- *File System Storage (per files XML)*: considerando che i files verranno storicizzati in formato ZIP (cosi' come vengono ricevuti via FTP), si e' stimato un ingombro di circa 10 GB per tutti i negozi della rete di vendita
- *Potenza di calcolo (CPU)*: 4 cores dedicati sin da subito all'ETL (principalmente per gestire le future validazioni dei files in parallelo, che sono le operazioni piu' intensive sulla cpu, trattandosi di parsing XML), 2 cores dedicati al Database Server, 1 core dedicato al sistema operativo, globalmente 7 cores; commercialmente si trovano solamente cpu con numero di cores multiplo di 2, quindi si e' scelto un sistema ad 8 cores.
- *Memoria RAM di sistema*: 1 GB dedicato al sistema operativo (sovrastimato), 3 GB dedicati al Database Server, 6 GB dedicati all'ETL; in totale 10 GB di RAM
- *Banda richiesta*: al fine di poter permettere un servizio continuativo e tempi di attesa ridotti nei confronti dei sender, si e' stimato che la banda richiesta per gestire i momenti di maggior afflusso sia verosimilmente pari a circa 4 Mbit dedicati

In aggiunta alle specifiche sopra indicate, le quali vanno a concorrere alla preparazione della macchina principale di gestione ETL, e' stata inoltre prevista una seconda macchina, dalle specifiche inferiori, dedicata ad application server, per la sola consultazione delle maschere web; tale macchina avra' 2 cores dedicati, 4 GB di ram, ed uno storage ininfluente ai fini delle esigenze (lo stretto indispensabile per installare il sistema operativo, l'application server ed il software web, ossia circa 10 GB).

3 La scelta del software e degli strumenti di sviluppo

In questo capitolo verra' considerata l'analisi tecnica effettuata al fine di stabilire i requisiti in termini di software da adottare per il supporto del progetto, e degli strumenti da adoperare per lo sviluppo del software; ci si concentrera' in particolar modo sullo strumento Kettle, utilizzato per lo sviluppo dell'ETL principale di elaborazione del flusso dati.

3.1 La memorizzazione dei dati: PostgreSQL

PostgreSQL e' un database relazionale ad oggetti completo, open source e rilasciato con licenza BSD, derivato da un progetto inizialmente sviluppato dell'Universita' di Berkley; l'installazione e la configurazione sono relativamente semplici e veloci, e la gestione del database e' molto accessibile, anche per neofiti nel settore (questo anche grazie a tool grafici come pgAdmin).

3.1.1 Caratteristiche e vantaggi

Fra le principali caratteristiche ed i punti di forza di PostgreSQL possiamo elencare:

- Supporto al linguaggio SQL ed al PL/pgSQL (un linguaggio nativo simile a PL/SQL di Oracle)
- Compatibilita' con la maggior parte dei linguaggi di programmazione (fra cui Java, C, Python, Perl, Ruby, etc) mediante Driver o Wrapper
- Supporto per lo sviluppo di procedure native scritte in linguaggio C / C++
- Gestione degli indici ottimizzata e flessibile
- Sistema di caching configurabile
- Planning delle query configurabile

- Supporto a procedure di replicazione automatizzate
- Grande stabilita' ed ottime performances
- Ottima documentazione esaustiva e buon supporto da parte della community / dei vari forum specializzati

3.1.2 Motivazione della scelta

La scelta per lo strumento di memorizzazione dei dati e' ricaduta su PostgreSQL principalmente in quanto e' uno dei database certificati a livello aziendale fra i piu' utilizzati, inoltre non richiede costi aggiuntivi (in quanto open source) ed e' gia' molto integrato con altri software aziendali, fra cui JStore (utilizzato anche in questo progetto, come di seguito meglio approfondito).

La versione adottata e' la 9.2.

3.1.3 Configurazione

La configurazione del server e' gestibile principalmente mediante due files di configurazione, presenti nella cartella di installazione del server:

- *postgresql.conf*: configurazione generale del server, fra cui:
  - numero di connessioni massime e concorrenti accettate
  - dimensione dei files di buffer
  - numero di transazioni che possono esser aperte dall'utente
  - configurazione del sistema di vacuum (per la “pulizia” dei dati)
  - configurazione del planning delle query
  - configurazione del sistema di logging (log level, percorsi dei files di log, etc)
  - gestione del sistema di lock
  - varie altre
- *pg_hba.conf*: configurazione delle policy di accesso al database, fra cui:
  - filtraggio hosts
  - sistema di autenticazione
  - permessi di accesso ai vari schemas / databases per utente
  - varie altre

3.2 L'elaborazione dei dati: Pentaho Data Integration (Kettle)

Pentaho Data Integration (chiamato anche “Kettle”) e' uno strumento di ETL (Extract Transform Load) open source, basato sui metadata, sviluppato in Java, con supporto all'interfacciamento verso la maggior parte dei database moderni, possibilita' di parallelizzazione e gestione del multi threading nativa, sviluppo visuale/grafico “drag and drop”, possibilita' di sviluppo di plugin, cloud computing, clustering e molte altre funzionalita' avanzate.

Kettle e' uno strumento che viene utilizzato aziendalmente da circa 3 anni con ottimi risultati, ed ha portato all'ottimizzazione di svariati ETL preesistenti, oltre allo sviluppo veloce e prestazionalmente valido di nuovi ETL richiesti dai clienti di IBC.

3.2.1 Motivazione della scelta

La scelta per lo strumento di elaborazione dei dati e' ricaduto su Kettle in quanto il know-how aziendale relativo allo stesso e' piuttosto alto; inoltre, vista la mole di dati da elaborare, dopo alcuni test di performance si e' dimostrato essere all'altezza delle elaborazioni richieste, restando entro le tempistiche massime ammesse dai requisiti.

La versione adottata e' la 4.3.0 GA.

3.2.2 Funzionalita' dello strumento utilizzate nel progetto

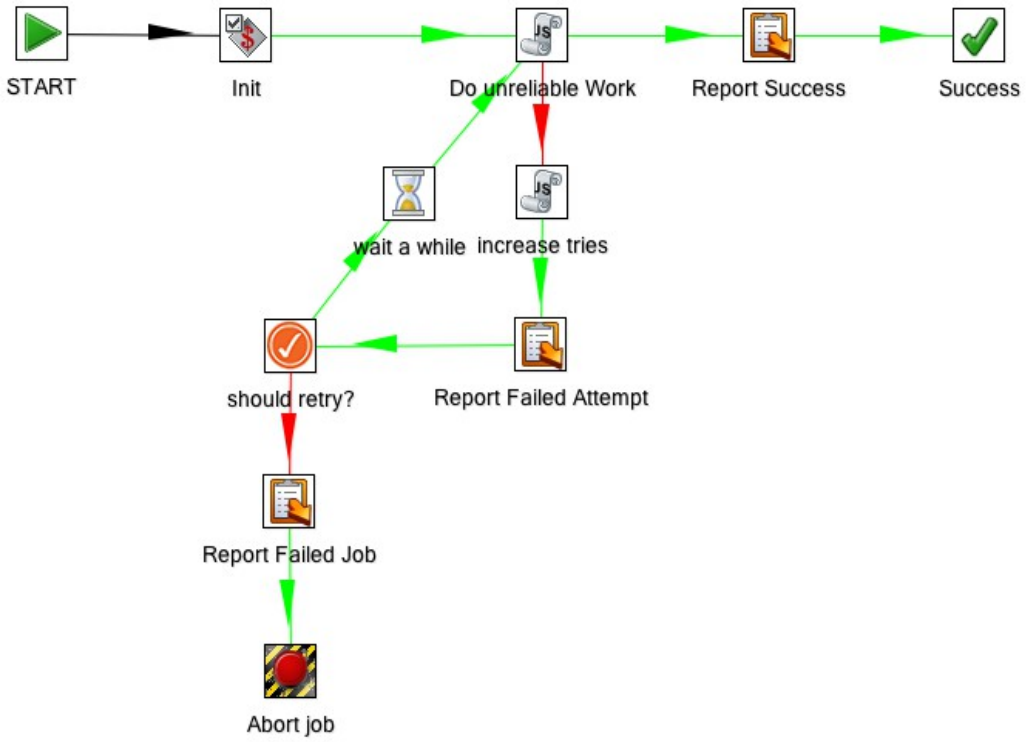
Per quanto di interesse al progetto, Kettle include il supporto a tutte le funzionalita' richieste dal software che si andra' a sviluppare:

- lettura e scrittura di dati su tabelle del database
- supporto ai lookup (utilizzati ad esempio per i controlli anagrafici)
- lettura da file XML (mediante libreria integrata XPath)
- validazione XSD “semplice”, con possibilita' di customizzazione mediante step che permettono di iniettare codice Java
- Supporto alle RegEx (regular expressions), utilizzate per il pattern matching sulle stringhe di errore del validatore (riconoscimento e categorizzazione degli errori)
- Supporto per le sequences su database (utilizzate per la generazione degli id delle righe)
- Supporto alla gestione di files compressi
- Supporto per la scrittura di files CSV
- Possibilita' di iniettare codice shell (utilizzato per il controllo di esecuzione concorrente)

- Accesso a file system (copia files, spostamento, cancellazione, etc)
- Filtraggio righe e switching sulla base di condizioni
- Manipolazione valori (ad esempio troncamento stringhe sulla base delle dimensioni dei campi specificate su db)
- Invio mail
- Grouping, sorting e controllo univocita' valori in memoria

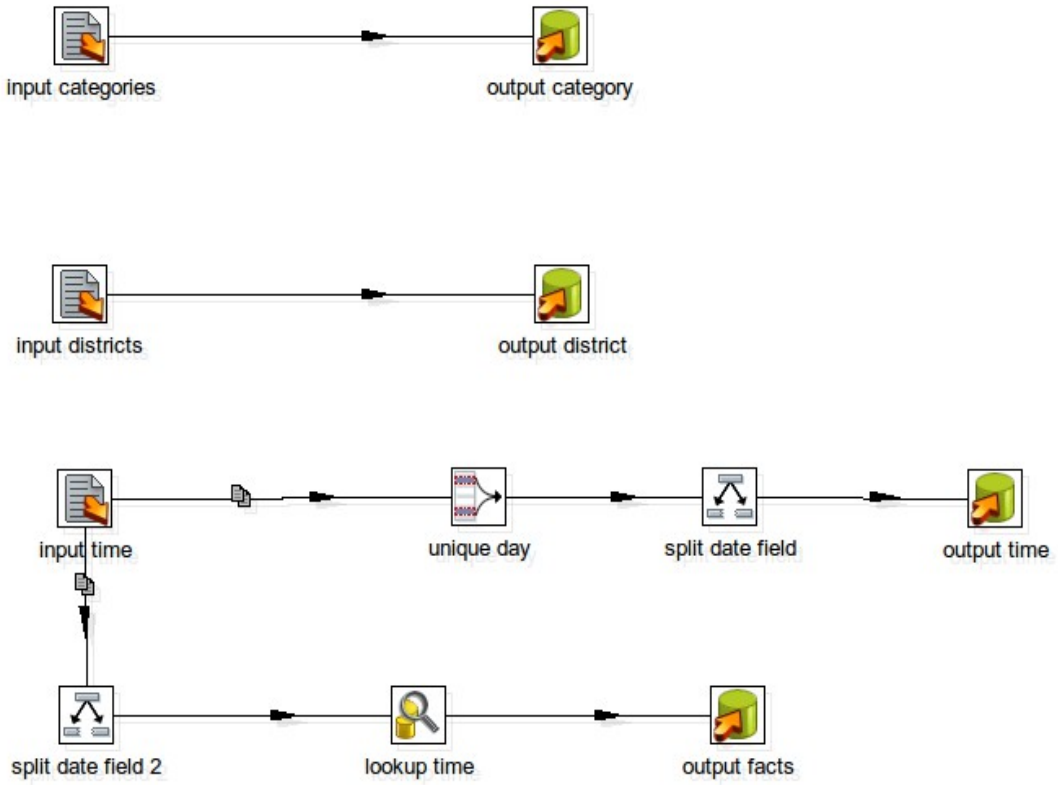
3.2.3 Elementi di cui si compongono i programmi

In Kettle sono presenti principalmente due tipologie diverse di macro elementi:



Jobs: i jobs permettono di stabilire il workflow a livello di macro elementi, e permettono di decidere le azioni da intraprendere a seconda dell'esito dei vari elementi di cui sono composti; essi permettono l'impiego di cicli e condizioni, e sono composti da elementi atomici, inoltre qui viene gestito il parallelismo fra i macro processi (se si decide di adottarlo), seppur ogni ramo di elaborazione venga poi eseguito serialmente.

Ogni job ha sempre un inizio ed una fine (quest'ultima puo' essere, come nell'esempio sopra riportato, sia un successo che porta ad un esito del job complessivamente positivo, sia un fallimento che porta ad un esito complessivamente negativo), e puo' seguire da 1 ad N rami di elaborazione.



Trasformate (Transformations): le trasformate servono a disegnare graficamente le micro elaborazioni che devono essere effettuate sui dati, quindi si occupano principalmente di gestire il flusso dati a livello di righe movimentate, da una o piu' sorgenti, verso una o piu' destinazioni; le trasformate sono parallele di natura, ed ogni suo elemento puo' esser visto come un thread che "vive di vita propria" (al contrario degli elementi che compongono i jobs), la cui esecuzione viene lanciata con l'avvio della trasformata, e termina solamente quando finisce di ricevere righe da elaborare da parte degli step precedenti.

Entrambi gli elementi introdotti sono composti da:

- *Steps*: sono gli elementi grafici rappresentati da un quadrato contenente un immagine rappresentativa della sua funzionalita'; essi rappresentano la singola operazione che viene eseguita in un determinato punto del flusso di elaborazione, e sono configurabili mediante pannelli appositi che vengono aperti da interfaccia grafica. Nei job sono semplici operazioni "prefatte" da eseguire atomicamente (oppure trasformate sviluppate dall'utente che vengono devono essere richiamate durante l'esecuzione), mentre nelle trasformate rappresentano un operazione da eseguire per ogni singola riga di dati, e si suddividono principalmente in tre tipologie:

    - Steps che generano dati (ad esempio step di lettura da database)
    - Steps che manipolano dati (ad esempio step di elaborazione su stringhe)
    - Steps che consumano dati (ad esempio step di scrittura su file csv)
- *Hops*: gli hops sono semplicemente le frecce di collegamento fra i vari steps di cui sono composti i jobs oppure le trasformate; a seconda del contesto (job o trasformata), si suddividono in diverse categorie:
    - Job / hops incondizionali (vengono sempre seguiti dal flusso)
    - Job / hops condizionali, a condizione vera (vengono seguiti solamente se lo step sorgente da cui parte l'hop ha esito positivo)
    - Job / hops condizionali, a condizione falsa (vengono eseguiti solamente se lo step sorgente da cui parte l'hop ha esito negativo)
    - Trasformata / hops normali (seguiti solitamente durante l'elaborazione dei dati)
    - Trasformata / hops di error handling (seguiti solamente nel caso di errori nello step che li ha generati, per i soli step che lo supportano; permettono di "redirigere" le singole righe che hanno creato problemi per una gestione custom, ad esempio inserimento dati errati in una tabella apposita)

3.3 La presentazione dei dati: JStore

JStore e' un software aziendale multi tier, multi database e multi platform, sviluppato in Java e specializzato nella gestione del business nell'ambito della GDO; dal punto di vista tecnico, si occupa principalmente di:

- gestione della persistenza dei dati mediante l'utilizzo della libreria di ORM "Hibernate", la quale utilizza JDBC per l'accesso al database, e mappa le entita' su database mediante utilizzo di EJB (Enterprise Java Beans)
- gestione della presentazione dei dati all'utente mediante interfaccia web, quest'ultima sviluppata appoggiandosi al framework "Wicket", con supporto HTTPS
- esposizione di web services mediante utilizzo della libreria Apache AXIS, la quale implementa il protocollo di interscambio dati SOAP
- Comunicazione interna fra software IBC mediante socket

mentre dal punto di vista pratico, si occupa di:

- Integrazione dati fra sistemi del cliente e/o sistemi esterni
- Data collect e processing

- Controllo dati e segnalazione automatizzata di eventuali anomalie al cliente e/o all'help desk IBC mediante mail
- Presentazione machere web funzionali alle esigenze, ad esempio con funzionalita' statistiche, di semplice consultazione, oppure operative (fatturazione in negozio, emissione bolle, gestionale, etc)

3.3.1 Motivazione della scelta

La scelta e' ricaduta su JStore in quanto e' il software principale proposto ai clienti da IBC per la gestione della sede, e poiche' implementa gia' un architettura client/web ottimale per la presentazione dei dati agli utenti, oltre ad avere le caratteristiche di espandibilita' e flessibilita' richieste per implementare quanto richiesto dal progetto.

JStore non ha una vera e propria “versione”, poiche' il software deployato dal cliente puo' essere aggiornato anche relativamente spesso (in base alle richieste di modifiche / nuove implementazioni); presenta invece una suddivisione in moduli funzionali, i quali vengono utilizzati per definire il contesto di utilizzo in termini di anagrafiche abilitate e maschere web accessibili agli utenti.

Per il committente B e' stato sviluppato un modulo funzionale apposito, in quanto le sue richieste sono molto specifiche e particolari del suo peculiare modello di business (prima fra tutte la gestione dell'articolo come combinazione di style/collection/size/color), per cui sono state sviluppate delle anagrafiche specializzate e delle maschere web di consultazione customizzate.

3.4 L'ambiente server: CentOS Linux

Per quanto riguarda l'ambiente server, si e' deciso di affidarsi alla soluzione CentOS (Community Enterprise Operating System), sistema operativo Enterprise-Class, open source, basato su Red Hat.

3.4.1 Caratteristiche e vantaggi

Fra le caratteristiche principali di CentOS figurano:

- piena compatibilita' con i binari RHEL (Red Hat Enterprise Linux)
- utilizzo di pacchetti RPM (Red Hat Package Manager) e del gestore di pacchetti YUM (Yellow Dog Updater Modified), quest'ultimo molto comodo e potente, con gestione automatica delle dipendenze (un po' come APT ed apt-get di Debian)
- supporto esteso di ben 10 anni per ogni release (a partire dalla versione 5)
- utilizzo esclusivamente di software libero secondo licenza GNU GPL

- completamente gratuito
- buona disponibilita' di software sui repository e sui canali ufficiali
- discreto supporto nei forum ufficiali
- possibilita' di adottare varianti minimal / server (senza interfaccia grafica) semplicemente specificandolo in fase di installazione
- molto stabile e conservativo (il passaggio a nuova major release del software installato non avviene finche' non si decide di aggiornare l'intero sistema all'ultimo rilascio, scelta peraltro facoltativa; viene comunque dato un adeguato supporto al software mediante rilascio continuativo di bugfix)

3.4.2 Motivazione della scelta

Si e' deciso di adottare CentOS in quanto aziendalmente certificato e molto usato, anche per via supporto esteso di 10 anni (fattore molto importante in ambiente enterprise), dalla grande stabilita' e della presenza di tutto il software necessario nei repository ufficiali (solo per installare l'ultima versione di PostgreSQL e' stato necessario aggiungere un repository, comunque mantenuto ufficialmente dal progetto di PostgreSQL).

La versione utilizzata e' l'ultima disponibile, ossia la 6.4.

3.5 Il sistema di gestione FTP: JScape Secure FTP

Il software JScape Secure FTP e' un prodotto commerciale omni comprensivo per la gestione del trasferimento via FTP client/server in ambiente enterprise; fra le sue caratteristiche possiamo elencare:

- supporto multipiattaforma Linux / Windows / Solaris / UNIX / AIX / Mac OSX
- supporto ai protocolli FTPS / SFTP / SCP / AFTP / HTTPS / AS2
- supporto ad autenticazione mediante LDAP, Active Directory, NTLM, PAM, database
- supporto alla creazione di triggers custom basati sugli eventi
- modulo dedicato per la prevenzione della perdita di dati
- virtual file system
- supporto per HA (high availability) e bilanciamento di carico
- gestione chiavi secondo standard PGP, SSL, SSH
- monitoraggio delle cartelle

- API di interfacciamento al server per utilizzo e configurazione compatibili Java e standard REST
- controllo di integrita' mediante checksum e supporto al resume dei trasferimenti

3.5.1 Motivazione della scelta

La scelta del software FTP e' stata delegata al reparto IT dell'azienda, ed esso ha optato per un software commerciale come JScape Secure FTP per avere stabilita' ed affidabilita', requisito fondamentale del progetto, in modo da poter garantire un servizio continuativo verso i senders e poter assicurare la consistenza nella gestione dei files ricevuti, oltre ad avere un supporto dedicato da parte di un azienda competente in caso di necessita'.

La versione adottata e' la 9.1 beta.

3.5.2 Modalita' di utilizzo

Il sistema e' stato deployato in un server dedicato dalle caratteristiche relativamente modeste (non erano necessari requisiti particolari per l'installazione), e viene utilizzato in modalita' FTPS, ossia FTP con supporto ai protocolli di crittografia TLS (Transport Layer Security) ed SSL (Secure Sockets Layer).

Per l'integrazione lato Java, sia lato software di negozio (per l'invio dei files), sia lato ETL di sede (per la ricezione e presa in carico dei files) viene utilizzata la libreria fornita in bundle con il software stesso.

4 La progettazione della base di dati

In questo capitolo verranno analizzate le varie tabelle che sono state implementate nel database per memorizzare i dati di interesse e per gestire le varie fasi operative dell'ETL.

4.1 Le categorie di informazioni memorizzate

Procediamo innanzitutto con una caratterizzazione delle informazioni memorizzate per gruppi, per poi analizzarli nel dettaglio nei prossimi paragrafi; possiamo suddividere i dati nelle seguenti macro categorie:

- *Dati Anagrafici*: in questa categoria rientrano le informazioni statiche, come ad esempio le anagrafiche di Negozi / Clienti / Senders / Software Houses; si tratta per lo piu' di dati "statici" (soggetti a poche variazioni), a bassa cardinalita' e che non dovrebbero occupare molto spazio
- *Data Collect:* in questa categoria abbiamo tutti i dati estratti dai files XML; si tratta di informazioni ad alta' cardinalita', soggette a crescere in numero con il tempo (almeno fino al raggiungimento del limite temporale di retention stabilito) e che non saranno mai soggette a modifiche (una volta caricati, i dati non vengono piu' modificati)
- *Errori del Data Collect:* le tabelle appartenenti a questa categoria servono per raccogliere tutti gli errori di validazione non bloccanti che sono direttamente riconducibili ad una foglia del file XML, e quindi ad un singolo elemento contenuto in una delle tabelle di data collect
- *Tabelle di utilita'*: si tratta di tabelle create per esigente operative, come ad esempio aggregare lo stato dei dati per giorno negozio (riportando se sono globalmente sono presenti errori, e se si, di quale categoria) oppure fungere da trascodifica degli errori a partire dai messaggi del validatore (in modo tale

da esser modificabile se in futuro si decidesse di cambiare la codifica degli errori)

Note salienti: ogni tabella presenta un campo “id”, chiave primaria, numerico da 19 cifre, gestito automaticamente da Hibernate mediante una sequence, ed utilizzato per i vincoli di integrita' referenziale fra tabelle.

Tutti i vincoli di univocita' aggiuntivi (sulla chiave logica / funzionale della tabella) vengono gestiti mediante Unique Key (le quali generano implicitamente anche un indice sui campi che le compongono).

Le tipologie dei campi verranno indicate mediante la specifica del loro tipo Java (con riferimento al Java Bean che mappera' la tabella), sottointendendo la seguente mappatura Java-to-SQL (applicata poi indirettamente da Hibernate):

- String → VARCHAR
- BigDecimal → NUMERIC
- Integer → INTEGER

Per tutte le tabelle si e' adottata la convenzione del naming al singolare (quindi STORE e non STORES), mantenendo il nome in inglese, sia per le tabelle, sia per i campi che le compongono, come definito dal committente.

4.2 Tabelle per dati anagrafici

In questo paragrafo verranno elencate le tabelle adibite alla memorizzazione di dati anagrafici, che sono nell'ordine:

- CUSTOMER (Cliente)
- STORE (Negozio)
- SENDER
- SOFTWARE_HOUSE

Come indicati nei capitoli precedenti, tali anagrafiche verranno alimentate da un flusso CSV proveniente giornalmente dai sistemi del committente B, e verranno utilizzate per alcuni controlli di business sulla consistenza dei dati, oltre ad esser consultabili mediante maschere di visualizzazione (questo principalmente per permettere all'Help Desk di ricercare velocemenete i recapiti telefonici / email di un Negozio / Cliente / Software House nel caso debba contattarlo).

4.2.1 CUSTOMER (Cliente)

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
code	String	1~10	Si	Unique Key
description	String	1~70	No	
address_id	BigDecimal	19,0	No	*1
email	String	1~255	No	
phone	String	1~20	No	
fax	String	1~20	No	
language_id	BigDecimal	19,0	No	*2
timezone_id	BigDecimal	19,0	No	*3

*1 – Foreign Key verso la tabella anagrafica di JStore "ADDRESS" (qui non riportata), la quale permette la memorizzazione di Via/Piazza, Localita', Cap, Provincia, Nazione ed altre informazioni di recapito (alcuni di questi campi solo a loro volta foreign keys verso altre anagrafiche, e.g. Nazione verso l'anagrafica delle nazioni)

*2 – Foreign Key verso la tabella anagrafica di JStore "LANGUAGE" (qui non riportata)

*3 – Foreign Key verso la tabella anagrafica di JStore "TIMEZONE" (qui non riportata)

4.2.2 STORE (Negozio)

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
code	String	1~10	Si	Unique Key
description	String	1~70	No	
sender_id	BigDecimal	19,0	Si	*1
customer_id	BigDecimal	19,0	Si	*2

address_id	BigDecimal	19,0	No	*3
email	String	1~255	No	
phone	String	1~20	No	
fax	String	1~20	No	
language_id	BigDecimal	19,0	No	*4
timezone_id	BigDecimal	19,0	No	*5
channel_id	BigDecimal	19,0	No	*6
active	String	1	Si	Flag S/N
activation_date	Integer		No	*7
deactivation_date	Integer		No	*7
date_first_sent	Integer		No	*7
date_last_sent	Integer		No	*7

*1 – Foreign Key verso la tabella “SENDER” riportata nel paragrafo successivo

*2 – Foreign Key verso la tabella “CUSTOMER” riportata nel paragrafo precedente

*3 – Foreign Key verso la tabella anagrafica di JStore “ADDRESS” (qui non riportata)

*4 – Foreign Key verso la tabella anagrafica di JStore “LANGUAGE” (qui non riportata)

*5 – Foreign Key verso la tabella anagrafica di JStore “TIMEZONE” (qui non riportata)

*6 – Foreign Key verso la tabella anagrafica di JStore “CHANNEL” (qui non riportata)

*7 – Date in formato numerico yyyyMMdd

4.2.3 SENDER

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
code	String	1~10	Si	Unique Key

description	String	1~35	No	
software_house_id	BigDecimal	19,0	Si	*1
email_to	String	1~255	Si	
email_cc	String	1~255	No	
files_prefix	String	1~20	No	
active	String	1	Si	Flag S/N
activation_date	Integer		No	*2
deactivation_date	Integer		No	*2
date_first_sent	Integer		No	*2
date_last_sent	Integer		No	*2

*1 – Foreign Key verso la tabella "SOFTWARE_HOUSE" riportata nel paragrafo successivo

*2 – Date in formato numerico yyyyMMdd

4.2.4 SOFTWARE_HOUSE

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
code	String	1~10	Si	Unique Key
description	String	1~35	No	
application_description	String	1~35	No	
email_to	String	1~255	Si	
email_cc	String	1~255	No	
address_id	BigDecimal	19,0	No	*1
phone	String	1~20	No	
fax	String	1~20	No	

language_id	BigDecimal	19,0	No	*2
timezone_id	BigDecimal	19,0	No	*3
active	String	1	Si	Flag S/N
activation_date	Integer		No	*4
deactivation_date	Integer		No	*4

*1 – Foreign Key verso la tabella anagrafica di JStore “ADDRESS” (qui non riportata)

*2 – Foreign Key verso la tabella anagrafica di JStore “LANGUAGE” (qui non riportata)

*3 – Foreign Key verso la tabella anagrafica di JStore “TIMEZONE” (qui non riportata)

*4 – Date in formato numerico yyyyMMdd

4.3 Tabelle di data collect

In questo paragrafo verranno elencate le tabelle adibite alla memorizzazione di dati estratti dai files XML sorgenti, che sono nell'ordine:

- XML_FILE (anagrafica di riferimento per tutte le seguenti)
- STOCK (Giacenza)
- INTERNAL_MOVEMENT (Movimento Interno)
- EXTERNAL_MOVEMENT (Movimento Esterno)
- SALE_HEADER (Testata di vendita)
- SALE_ROW (Riga di vendita)
- SALES_AMOUNT (Ammontare giornaliero)
- SALE_DELETE (Cancellazione di vendita)

Queste anagrafiche verranno alimentate dalle varie esecuzioni dell'ETL, e sono destinate a crescere in dimensione fino al raggiungimento del range temporale impostato per la retention dei dati (2 anni).

In tutte le tabelle, si e' deciso di riportare le informazioni riguardanti gli articoli (style/collection/size/color) in modo piatto (quindi senza references ad una tabella anagrafica esterna), poiche' la cardinalita' delle possibili combinazioni delle stesse avrebbe portato alla creazione di una tabella dedicata avente un numero di elementi

troppo elevato (e conseguentemente sarebbe stata poco funzionale, oltre che molto lenta in fase di lookup / aggiornamento durante l'estrazione dei dati dai files XML).

4.3.1 XML_FILE

La tabella XML_FILE e' stata creata per gestire ad alto livello tutte le informazioni presenti all'interno di un singolo file XML, e viene poi referenziata da tutte le tabelle seguenti (come fosse una tabella di "master").

Essa contiene informazioni utili sul file (ad esempio l'istante di generazione), oltre ad alcuni flag utili al software (come il flag per indicare se tutti i dati contenuti al suo interno sono stati esportati sui files CSV di output).

Le informazioni riguardanti la data/ora di generazione vengono estratte direttamente dal nome del file (e non sono obbligatorie appositamente per gestire files che presentano naming errato), mentre data/ora di ricezione vengono gestite dal programma (e sono state memorizzate in due campi separati per facilitare le ricerche).

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	*1
file_name	String	1~255	Si	
date_generated	Integer		No	*2
time_generated	Integer		No	*3
date_received	Integer		Si	*2
time_received	Integer		Si	*3
last_modified	BigDecimal	14,0	Si	*4
import_timestamp	BigDecimal	14,0	Si	*4
export_timestamp	BigDecimal	14,0	No	*4
exported	String	1	Si	Flag S/N
errors	String	1	Si	Flag S/N

*1 – Foreign Key verso la tabella "STORES" riportata nei paragrafi precedenti

*2 – Date in formato numerico yyyyMMdd

*3 – Orari in formato HhmmSS

*4 – Timestamp in formato numerico yyyyMMddHHmmss

4.3.2 STOCK (Giacenza)

La tabella degli stocks riporta le quantita' in giacenza per ogni articolo presente nel punto vendita, ed e' l'unica tabella ad esser sempre popolata per ogni file XML sorgente (a causa del vincolo di obbligatorieta' per quanto riguarda la presenza di questo flusso nei files).

Dato che ogni file XML contiene i dati di giacenza di un solo giorno, e considerando che l'articolo e' in chiave per la giacenza di ogni giorno, la chiave complessiva e' stata definita sui campi che compongono l'articolo + la reference al file XML sorgente.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	*1
stock_date	Integer		Si	*2
article_type	String	1	No	
ean_code	String	1~13	No	
style	String	1~18	Si	Unique Key
collection	String	1~3	Si	Unique Key
size	String	1~3	Si	Unique Key
color	String	1~3	Si	Unique Key
stock_quantity	BigDecimal	7,3	Si	
item_medium_cost	BigDecimal	10,3	No	
suggested_price	BigDecimal	10,3	No	
xml_file_id	BigDecimal	19,0	Si	Unique Key *3

*1 – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Foreign Key verso la tabella “XML_FILE” riportata nei paragrafi precedenti

4.3.3 INTERNAL_MOVEMENT (Movimento Interno)

Trattandosi di movimenti interni, si puo' notare che sia il negozio sorgente, sia il negozio destinazione sono references alla tabella anagrafica “STORE”; questo poiche' sono entrambe entita' note, a differenza dei movimenti esterni, in cui il negozio sorgente non e' noto (e' un entita' esterna, appunto).

Come indicato nei primi capitoli, il negozio che ha generato il file XML, e quindi quello principale di riferimento, e' sempre il negozio di destinazione.

La chiave e' stata definita sulla combinazione di campi negozio sorgente + negozio destinazione + data + campi che definiscono l'articolo + file XML sorgente, coerentemente con quanto visto nei capitoli precedenti.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
source_store_id	BigDecimal	19,0	Si	Unique Key *1
destination_store_id	BigDecimal	19,0	Si	Unique Key *1
movement_date	Integer		Si	Unique Key *2
shipping_date	Integer		No	*2
document_code	String	1~30	No	
article_type	String	1	No	
ean_code	String	1~13	No	
style	String	1~18	Si	Unique Key
collection	String	1~3	Si	Unique Key
size	String	1~3	Si	Unique Key
color	String	1~3	Si	Unique Key
movement_quantity	BigDecimal	7,3	Si	
item_medium_cost	BigDecimal	10,3	No	
suggested_price	BigDecimal	10,3	No	
xml_file_id	BigDecimal	19,0	Si	Unique Key *3

*1 – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

*2 – Date in formato numerico yyyyMMdd

*3 – Foreign Key verso la tabella “XML_FILE” riportata nei paragrafi precedenti

4.3.4 EXTERNAL_MOVEMENT (Movimento Esterno)

A differenza dei movimenti interni, qui il negozio sorgente non appartiene all'anagrafica dei Negozi, quindi e' rappresentato semplicemente dal suo codice alfanumerico.

Per il resto la struttura e' identica a quella dei movimenti interni (si e' voluto creare una tabella dedicata per rispettare la struttura del file XML), chiave compresa.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
source_store_id	String	1~20	Si	Unique Key
destination_store_id	BigDecimal	19,0	Si	Unique Key *1
movement_date	Integer		Si	Unique Key *2
shipping_date	Integer		No	*2
document_code	String	1~30	No	
article_type	String	1	No	
ean_code	String	1~13	No	
style	String	1~18	Si	Unique Key
collection	String	1~3	Si	Unique Key
size	String	1~3	Si	Unique Key
color	String	1~3	Si	Unique Key
movement_quantity	BigDecimal	7,3	Si	
item_medium_cost	BigDecimal	10,3	No	
suggested_price	BigDecimal	10,3	No	
xml_file_id	BigDecimal	19,0	Si	Unique Key *3

*1 – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

*2 – Date in formato numerico yyyyMMdd

*3 – Foreign Key verso la tabella "XML_FILE" riportata nei paragrafi precedenti

4.3.5 SALE_HEADER (Testata di vendita)

La chiave e' stata definita su negozio + data + id scontrino + file xml, in modo tale da permettere la ritrasmissione degli scontrini su diversi files.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	Unique Key *1
sale_date	Integer		Si	Unique Key *2
sale_time	Integer		No	*3
sale_id	String	1~18	Si	Unique Key
sale_amount	BigDecimal	9,3	Si	
vat_amount	BigDecimal	9,3	No	
discount_amount	BigDecimal	9,3	No	
xml_file_id	BigDecimal	19,0	Si	Unique Key *4

*1 – Foreign Key verso la tabella "STORES" riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Ora in formato HHmmSS

*4 – Foreign Key verso la tabella "XML_FILE" riportata nei paragrafi precedenti

4.3.6 SALE_ROW (Riga di vendita)

Questa tabella si lega direttamente alla testata di vendita, ereditandone quindi i riferimenti al file XML sorgente, data, negozio, etc.

La chiave e' composta come combinazione del riferimento alla testata + identificativo riga.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
sale_header_id	BigDecimal	19,0	Si	Unique Key *1
row_id	String	1~3	Si	Unique Key
article_type	String	1	No	
ean_code	String	1~13	No	
style	String	1~18	Si	
collection	String	1~3	Si	
size	String	1~3	Si	
color	String	1~3	Si	
sold_quantity	BigDecimal	3,3	Si	
sold_amount	BigDecimal	9,3	Si	
discount_amount	BigDecimal	9,3	No	
item_medium_cost	BigDecimal	9,3	No	
vat_percentage	BigDecimal	5,2	No	
vat_amount	BigDecimal	9,3	No	

*1 – Foreign Key verso la tabella “SALE_HEADER” riportata nel paragrafo precedente

4.3.7 SALES_AMOUNT (Ammontare giornaliero)

Anche qui, la chiave e' stata estesa al file XML per poter gestire eventuali ritrasmissioni (rettifiche).

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	Unique Key *1
sales_date	Integer		Si	Unique Key *2

sales_total_amount	BigDecimal	11,3	Si	
xml_file_id	BigDecimal	19,0	Si	Unique Key *3

*1 – Foreign Key verso la tabella "STORES" riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Foreign Key verso la tabella "XML_FILE" riportata nei paragrafi precedenti

4.3.8 SALE_DELETE (Cancellazione di vendita)

Questa tabella non e' stata legata alle testate di vendita poiche' la testata relativa ad una determinata cancellazione potrebbe non esser presente, e per specifiche di analisi bisogna comunque elaborare le cancellazioni senza testata di riferimento, passandole poi al sistema di Data Warehouse del committente specificando un codice di errore apposito per identificare tale anomalia.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	Unique Key *1
sale_date	Integer		Si	Unique Key *2
sale_time	Integer		No	*3
sale_id	String	1~18	Si	Unique Key
xml_file_id	BigDecimal	19,0	Si	Unique Key *4

*1 – Foreign Key verso la tabella "STORES" riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Ora in formato HHmmSS

*4 – Foreign Key verso la tabella "XML_FILE" riportata nei paragrafi precedenti

4.4 Tabelle per errori del data collect

Le tabelle degli errori del data collect servono per memorizzare in maniera strutturata tutti gli errori non bloccanti che vengono rilevati dal validatore XSD; esse presentano tutte la stessa struttura, ossia una reference alla tabella di data collect di afferenza (che permette di identificare l'elemento che ha generato l'errore), il codice dell'errore rilevato ed un flag che indica se l'errore e' valido oppure no, assieme all'eventuale motivazione della chiusura (invalidazione) dell'errore.

La chiave univoca di tutte le tabelle e' estesa su entrambi i campi che le compongono, in quanto e' possibile avere solo una volta lo stesso errore su di una singola riga di dati.

Viene di seguito riportata la struttura generica:

4.4.1 ERROR_[NOME_TABELLA_DATA_COLLECT]

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
[nome_tabella]_id	BigDecimal	19,0	Si	Unique Key *1
error_code	String	1~10	Si	Unique Key
valid	String	1	Si	Flag S/N
closure_reason	String	1~255	No	

*1 – Foreign Key verso la tabella di data collect “[nome_tabella]” riportata nei paragrafi precedenti

4.5 Tabelle di utilita'

In questa categoria abbiamo solamente tre tabelle:

- FLOW_STATUS (Stato flusso)
- BLOCKING_ERROR (Errore bloccante)
- ERRORS_TABLE (Tabella anagrafica degli errori)
- ERROR_TRANSCODING (Transcodifica errore)

Tali tabelle sono classificate come “di utilita'” in quanto non sono strettamente funzionali all'ETL (il quale, con un implementazione semplificata, potrebbe procedere comunque in loro assenza, poiche' si tratta di informazioni che non vengono trasmesse al sistema di Data Warehouse del committente mediante flusso CSV), ma servono piu' che altro per memorizzare informazioni utili ad altre funzionalita', quali le maschere web di consultazione dati oppure le mail di alert.

4.5.1 FLOW_STATUS (Stato flusso)

La tabella di stato flusso permette di memorizzare in modo aggregato lo stato dei vari flussi (intesi come vendite, movimenti, etc) suddiviso per giorno/negozio; tali informazioni sono utili per velocizzare le consultazioni mediante maschere web (le

quali sarebbero costrette altrimenti a consultare direttamente i dati caricati, che pero' hanno una cardinalita' assai maggiore).

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	Unique Key *1
reference_date	Integer		Si	Unique Key *2
flow	String	1~20	Si	Unique Key *3
status	String	1~15	Si	*4
blocking_errors	String	1	Si	Flag S/N
non_blocking_errors	String	1	Si	Flag S/N
last_modified	BigDecimal	14,0	Si	*5

*1 – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Puo' assumere come valore il nome di una qualsiasi delle tabelle di data collect

*4 – Puo' assumere come valore: OK (dati ricevuti senza errori), MISSING (dati mai ricevuti), OPEN_ERRORS (dati ricevuti ma presenti errori non ancora risolti), CLOSED_ERRORS (dati ricevuti ed errori presenti in passato, ora risolti)

*5 – Timestamp in formato numerico yyyyMMddHHmmss

4.5.2 BLOCKING_ERROR (Errore bloccante)

Questa tabella permette di memorizzare gli errori bloccanti rilevati in un file XML, informazione che altrimenti non verrebbe memorizzata da altre parti (dato che i dati non vengono estratti in caso di presenza di errori bloccanti).

La codifica dell'errore e' statica, cosi' se l'errore dovesse cambiare transcodifica, la modifica non sarebbe retroattiva.

Nel campo “additional_details” viene memorizzata la stringa “grezza” ricevuta in output dal validatore (la quale riporta anche riga e colonna dell'errore nel file XML), mentre i campi “valid” e “closure_reason” servono per l'eventuale invalidazione manuale dell'errore, gestita da maschera web.

La tabella non ha chiave univoca, in quanto e' possibile avere piu' volte lo stesso errore nello stesso file XML, ed alla pari delle tabelle di errori sui dati, prevede un flag di validita'.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
store_id	BigDecimal	19,0	Si	*1
reference_date	Integer		Si	*2
error_code	String	1~10	Si	
xml_file_id	BigDecimal	19,0	Si	*3
additional_details	String	1~500	No	
valid	String	1	Si	Flag S/N
closure_reason	String	1~255	No	
last_modified	BigDecimal	14,0	Si	*4

*1 – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

*2 – Data in formato numerico yyyyMMdd

*3 – Foreign Key verso la tabella “XML_FILE” riportata nei paragrafi precedenti

*4 – Timestamp in formato numerico yyyyMMddHHmmss

4.5.3 ERRORS_TABLE (Tabella anagrafica degli errori)

Tale tabella, fornita dal committente del progetto, serve a censire tutti i possibili errori che si vuole gestire, bloccanti o non, in modo da assegnare ad ognuno di essi una codifica, una descrizione, e dei flag che verranno poi utilizzati per stabilire se includere l'errore nelle specifiche mail di notifica.

Le entry presenti in questa tabella non verranno mai cancellate, ma al piu' modificate (principalmente per abilitazione/disabilitazione di flag), ed eventualmente ne verranno aggiunte in futuro nel caso dovesse rendersi necessario gestire nuove tipologie di errori; la manutenzione della tabella e' effettuata mediante apposita maschera web.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
code	String	1~10	Si	Unique Key
description	String	1~100	Si	
english_description	String	1~100	Si	

class	String	1~10	Si	
category	String	1~20	Si	
typology	String	1~20	Si	
source	String	3	Si	
type	String	1	Si	
gravity	String	1	Si	
email_customer	String	1	Si	Flag S/N
email_store	String	1	Si	Flag S/N
email_sender	String	1	Si	Flag S/N
email_software_house	String	1	Si	Flag S/N
last_modified	BigDecimal	14,0	Si	**1*
active	String	1	Si	Flag S/N

**1* – Timestamp in formato numerico yyyyMMddHHmmss

4.5.4 ERROR_TRANSCODING (Transcodifica errore)

La tabella di transcodifica errore e' una tabella di appoggio statica, manutenzionata manualmente (non sono previste maschere automatizzate in quanto non dovrebbe quasi mai esser modificata), e viene utilizzata dall'ETL per stabilire la codifica da assegnare agli errori che rileva in fase di validazione in base alla stringa di output fornita dal validatore XSD.

Nome colonna	Tipo campo	Dimensione	Obbligatorio	Note
flow_name	String	1~20	Si	Unique Key **1*
flow_field_name	String	1~50	Si	Unique Key **2*
xsd_error_code	String	1~50	Si	Unique Key
error_code	String	1~10	Si	
last_modified	BigDecimal	14,0	Si	**3*

**1* – Foreign Key verso la tabella “STORES” riportata nei paragrafi precedenti

**2* – Puo' assumere come valore il nome di una qualsiasi delle tabelle di data collect

**3* – Timestamp in formato numerico yyyyMMddHHmmss

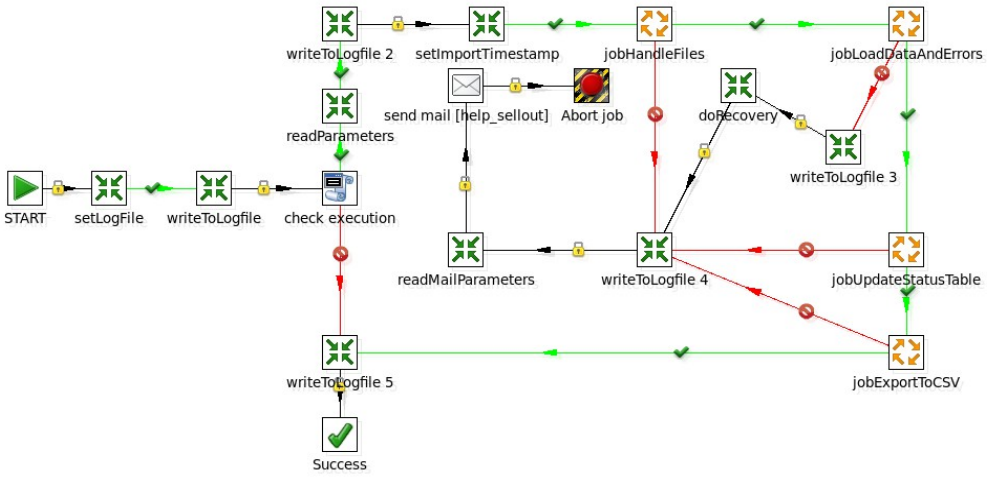
5 La progettazione e lo sviluppo dell'ETL

In questo capitolo verra' analizzato l'ETL che e' stato sviluppato per le esigenze del progetto, partendo da una visione di insieme, effettuando innanzitutto una prima suddivisione in macro job, per poi scendere piu' nel dettaglio e nelle specifiche operazioni eseguite dal singolo job e di alcune trasformate ritenute importanti nel contesto generale (in particolar modo la trasformata di validazione ed estrazione degli errori).

5.1 Una visione d'insieme

Il compito dell'ETL e' quello di eseguire, nell'ordine, tutte le operazioni indicate nel primo capitolo (quando si e' accennato alle esigenze funzionali), quindi di prendere in carico un insieme di files presenti sul server FTP nel momento dell'esecuzione e di processarli, applicando le logiche definite.

Se apriamo il Job principale che gestisce l'elaborazione complessiva, ci troviamo di fronte al seguente schema, apparentemente complicato:



Provando a dare un significato agli elementi ivi presenti, abbiamo:

- un punto di partenza univoco, identificato da un icona rappresentante una freccia verde rivolta verso destra, ed una didascalia con la descrizione “START”
- un punto di uscita nel caso l'elaborazione vada a buon fine, identificato da un icona rappresentante un segno di spunta verde, ed una didascalia con la descrizione “Success”
- un punto di uscita nel caso l'elaborazione NON vada a buon fine, identificato da un icona rappresentante un cerchio riempito di rosso, con uno sfondo di righe oblique alternate nere e gialle, ed una didascalia con la descrizione “Abort Job”
- un operazione di invio mail, preposta all'icona di Abort Job
- un operazione di shell scripting che effettua un check sull'esecuzione di un altra istanza del programma, identificata da un icona rappresentante un papiro azzurro con l'icona di una shell in sovrimpressione, ed una didascalia con la descrizione “check execution”
- vari job e trasformate richiamate durante l'esecuzione, identificate da un icona con delle frecce arancioni disposte in cerchio in senso orario (job) oppure da un icona con delle frecce verdi che puntano verso il centro dell'icona (trasformate)
- varie freccie di collegamento, alcune di color verde con una spunta verde a meta', alcune di color rosso con un divieto rosso a meta', ed alcune di color nero con un lucchetto giallo a meta'

Tutte le icone sono degli steps, mentre tutte le freccie sono degli hops; richiamando quanto spiegato nel capitolo riguardante Kettle, abbiamo che ogni step (a livello di job) compie una diversa operazione atomica, come fare da punto d'ingresso nel workflow (step di START) oppure terminare il programma con stato di errore (step di Abort Job), mentre gli hops mettono in collegamento i vari steps a seconda dell'esito dello step da cui partono: in particolar modo, gli hops verdi vengono seguiti solamente se lo step di partenza ha avuto esito positivo, gli hops rossi vengono seguiti solamente se lo step ha avuto esito negativo, mentre gli hops neri vengono seguiti incondizionatamente dal risultato dello step che li precede.

Seguendo il flusso di un elaborazione che va' a buon fine, possiamo vedere che vengono eseguite nell'ordine le seguenti operazioni:

- impostazione file di log
- scrittura di un messaggio su file di log (avvio del programma in corso)
- controllo dell'esecuzione di un altra istanza del programma; questa operazione puo' avere due esiti:

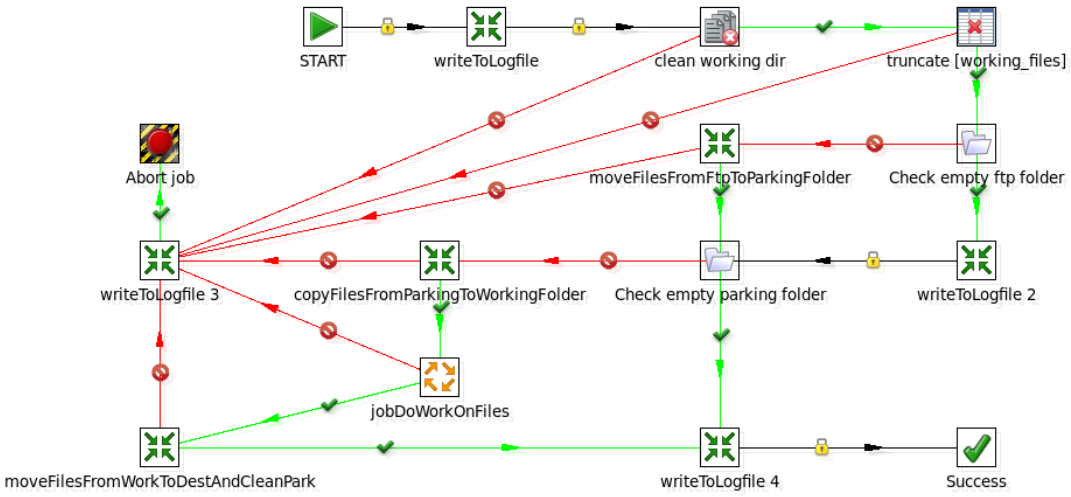
    - programma gia' in esecuzione, porta al termine del programma con esito positivo, dopo aver scritto un messaggio su log (programma gia' in esecuzione)
    - programma non in esecuzione, continua normalmente con le altre operazioni
- lettura parametri di esecuzione
- scrittura di un messaggio su file di log (lettura parametri effettuata con successo)
- impostazione di una variabile rappresentante l'istante di import dei dati (tale informazione verra' poi condivisa da tutti i dati caricati durante questa esecuzione, ed in particolar modo verra' riportata su tutte le entry che verranno inserite nella tabella XML_FILE, vedi capitolo precedente)
- esecuzione del job di “Handle Files”; questo si occupa di prendere in carico i files dal server FTP
- esecuzione del job di “Load Data And Errors”; questo e' il Job piu' corposo dell'ETL, e si preoccupa di estrarre dati ed errori dai files XML, di collegarli, di gestire le casistiche di ritrasmissione e di cancellazione (vedremo poi il motivo di tale necessita'), oltre ad applicare le regole di business definite dal committente ed infine spostare i files nelle cartelle di storico su file system
- esecuzione del job di “Update Status Table”; questo effettua l'aggiornamento della tabella “FLOW_STATUS” definita nel capitolo precedente in base ai dati ricevuti
- esecuzione del job di “Export To CSV”; questo effettua l'esportazione dei dati nei files CSV di export per l'invio al Data Warehouse del committente, compresa la compressione in archivio compresso tar.gz
- scrittura di un messaggio su file di log (elaborazione terminata con successo)

Nel caso in cui si verifichi un problema adurante l'esecuzione di un qualsiasi passo del Job, il problema viene scritto su file di log, e poi si provvede a segnalarlo via mail al gruppo di Help Desk (i parametri per l'invio della mail vengono letti da file di configurazione), in modo tale che quest'ultimo possa fare le verifiche necessarie ed eventualmente avvertire il gruppo di sviluppo, nel caso in cui le problematiche riscontrate vadano oltre le loro competenze.

L'unico caso in cui e' necessario provvedere ad effettuare delle azioni aggiuntive e' quello in cui si verifichi un errore durante il job di “Load Data And Errors”; in tal caso viene richiamata una trasformata specializzata che effettua delle operazioni di recovery al fine di mantenere l'integrita' dei dati.

Il dettaglio delle azioni compiute dai vari job verra' analizzata nel corso nei prossimi paragrafi.

5.2 jobHandleFiles



Il Job di "Handle Files" (gestione files) ha il compito di prendere in carico i files presenti sul server FTP nel momento in cui viene lanciato l'ETL ed effettuare le prime elaborazioni (controllo naming, de-compressione, etc).

Esso utilizza una tabella temporanea di appoggio (non documentata) di nome "working_files", in cui viene inserita ad ogni esecuzione una lista dei files presi in carico, la quale verra' poi utilizzata come riferimento per ogni sotto trasformata e sotto job (in modo da non dover controllare ogni volta il file system).

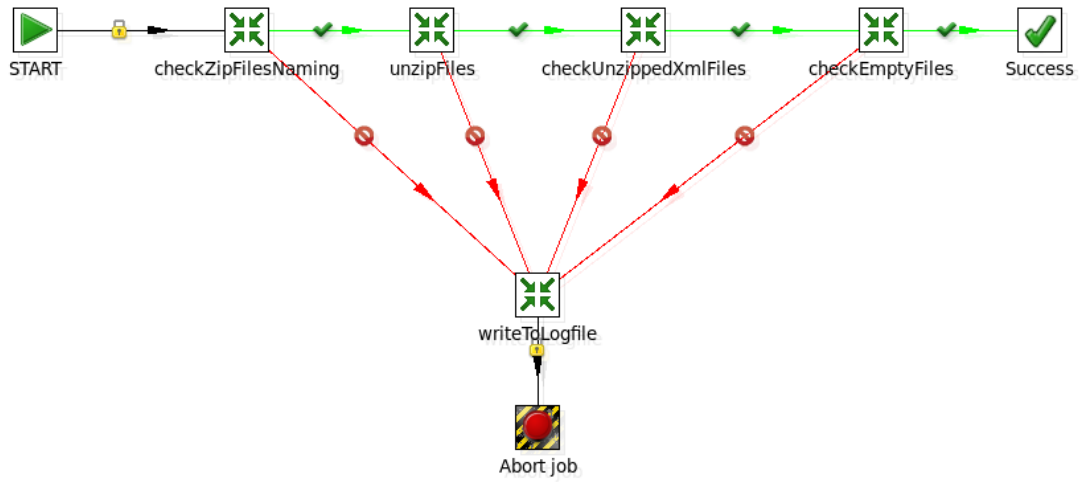
Oltre alla tabella temporanea "working_files", il job si appoggia a tre cartelle su file system, di cui due utilizzate solo come cartelle "di transito" ed una cartella "definitiva":

- *parking folder*: la cartella di "parking" (parcheggio) e' la prima delle tre in cui transitano i files; il suo scopo e' semplicemente ospitare i files nella prima fase del job, in modo tale da rimuoverli quanto prima dal server FTP (i files vengono quindi SPOSTATI)
- *working folder*: la cartella di "working" (lavoro) e' la seconda cartella in cui transitano i files; tale cartella viene ripulita ad ogni elaborazione, e contiene temporaneamente i files (che vengono quindi COPIATI in questa cartella a partire da quella di parcheggio) in modo tale da permettere l'esecuzione delle operazioni del job sui files, come ad esempio la de-compressione
- *destination folder*: la cartella di "destination" (destinazione) e' per l'appunto l'ultima cartella in cui transitano i files, ed e' anche la cartella da cui inizieranno tutte le elaborazioni da parte dei jobs successivi; i files vengono

quindi SPOSTATI in questa cartella e CANCELLATI dalla cartella di parcheggio solamente al termine di tutte le operazioni previste dal job

A livello di job principale abbiamo quindi la gestione “macro” di queste operazioni di transito dei files fra le varie cartelle e la gestione implicita della tabella di “working_files” mediante le trasformate chiamate direttamente, mentre il lavoro “vero e proprio” sui files viene eseguito dal sotto job di “Do Work On Files”.

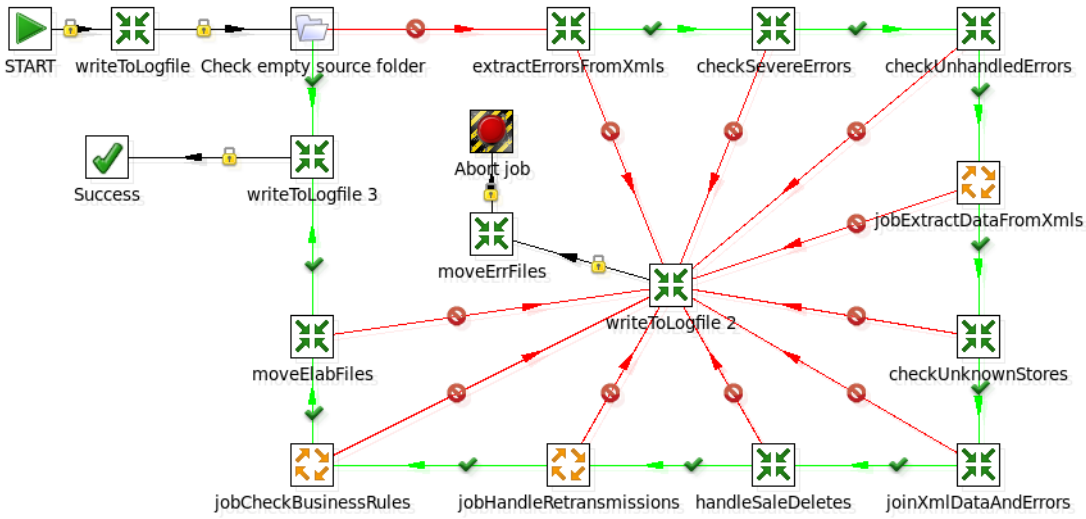
5.2.1 jobDoWorkOnFiles



Il Job di “Do Work On Files” esegue nell'ordine le seguenti operazioni:

- *controllo naming files*: per ogni file preso in carico viene controllato che rispetti la naming convention stabilita, ed inoltre che le entita' referenziate implicitamente dal nome del file (Software House, Negozio) siano presenti in anagrafica e siano correttamente legate mediante un sender
- *de-compressione files*: effettua l'unzip dei files controllandone l'integrita', oltre ad accertarsi che ognuno di essi contenga solamente un file con naming coerente al file ZIP che lo contiene (come stabilito in analisi)
- *controllo files scompressi*: verifica che i files XML scompressi siano ben formati, e che la loro dimensione non superi singolarmente i 50 MB (anche questa specifica come stabilito in analisi)
- *controllo files vuoti*: controlla se sono presenti files “vuoti business”, in modo tale da scartarli sin da subito ed evitare lavoro inutile in seguito da parte dell'ETL; con il termine “vuoto business” si intende un file da zero bytes oppure con dimensione maggiore di 0 e contenente delle tag XML, ma che non contiene nessun dato utile ai fini pratici (e.g. Tag aperte e subito dopo chiuse)

5.3 jobLoadDataAndErrors



Il Job di "Load Data And Errors" viene richiamato subito dopo l'Handle Files, e quindi si trova a lavorare con dei files XML che hanno gia' superato i primi controlli.

Esso provvede a:

- estrarre gli errori dai files XML
- controllare la presenza di errori bloccanti
- estrarre i dati dai files XML
- controllare se i files XML contengono dati relativi a negozi non codificati
- unire gli errori rilevati ai dati estratti, eseguendo la transcodifica degli stessi secondo la mappatura definita nella tabella ERROR_TRANSCODING
- gestire casistiche particolari nei dati, quali cancellazioni di vendita e ritrasmissioni
- applicare le regole di business

Al termine dell'elaborazione i files vengono spostati nelle cartelle di storico, una per i files elaborati correttamente, ed una per i files che hanno presentato degli errori durante l'elaborazione.

5.3.1 extractErrorsFromXml

Questa trasformata (di cui non viene riportata la struttura per motivi di spazio, essendo abbastanza corposa) si occupa della validazione e categorizzazione degli errori rilevati nei files XML.

Per rilevare gli errori viene utilizzata uno step di “User Defined Java Class”, in cui e' stata iniettata una classe Java customizzata per eseguire la validazione richiesta, la quale si appoggia agli oggetti di parsing XML standard delle librerie Java (SAXParser), con utilizzo di un Handler sviluppato anch'esso “ad-hoc” e che permette di tener traccia del livello di profondita' raggiunto nel corso della navigazione dell'albero XML, assieme a tutti i riferimenti per identificare il ramo preciso in cui si e' arrivati (memorizzando anche i valori di alcuni attributi dei nodi attraversati).

La stringa di risultato restituita dallo step di validazione (anch'essa customizzata per facilitare l'identificazione dell'errore ed il recupero dei riferimenti) passa poi attraverso una catena di step che applicano delle regular expressions di pattern matching sulla stringa (per stabilire se l'errore e' di loro competenza) e quindi inserire l'errore (a questo punto ancora “grezzo”, non transcodificato) in delle tabelle temporanee, assieme ai riferimenti dell'errore estratti dalla stringa mediante capturing group definiti nelle regex.

Nel caso in cui non si riesca a classificare correttamente l'errore, le destinazioni possibili per lo stesso sono due:

- una tabella di “errors uncategorized”, ossia errori riconosciuti ma volutamente non gestiti
- una tabella di “errors unhandled”, ossia errori che non si e' proprio riusciti a riconoscere

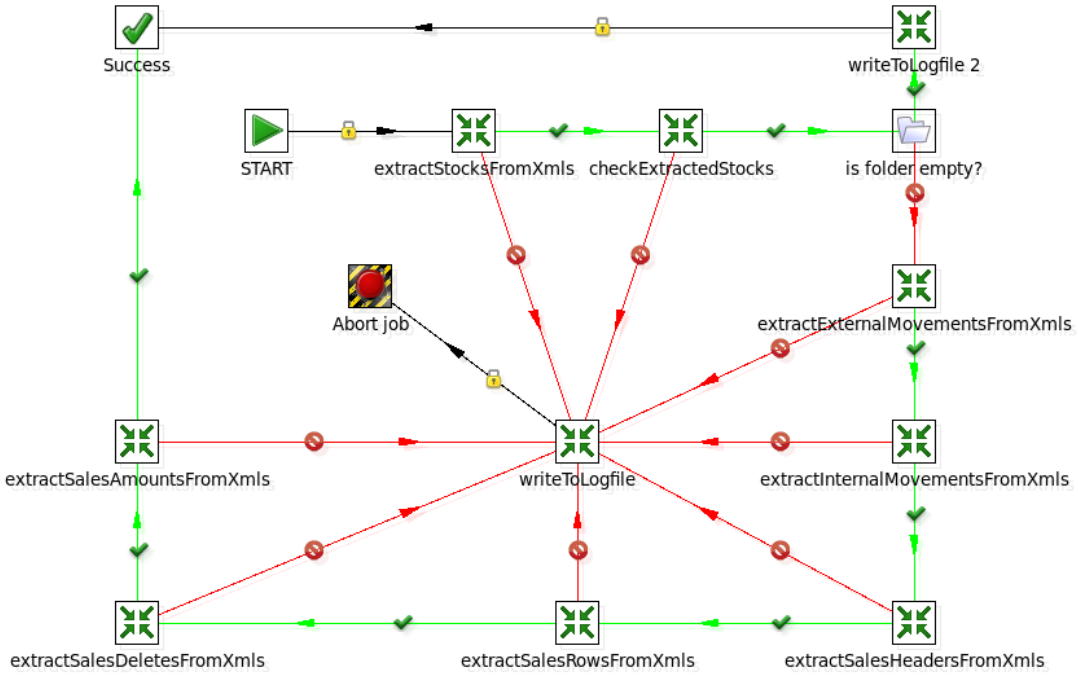
Tutti gli errori inseriti in queste due ultime tabelle, verranno poi caricati nella base di dati definitiva come “errori bloccanti”, mentre tutti gli errori inseriti nelle altre tabelle (una per tipo dato, come nelle tabelle definitive) verranno caricati come “errori non bloccanti” nelle rispettive tabelle di pertinenza.

5.3.2 checkSevereErrors / checkUnhandledErrors

Queste due trasformate molto semplicemente gestiscono gli eventuali errori inseriti nelle due tabelle sopra indicate di “errors_unhandled” ed “errors_uncategorized”; l'unica differenza fra le due tipologie e' che gli errori “non categorizzati” non sono stati gestiti volutamente, ma presentano comunque una codifica di errore specializzata per ognuno di essi, quindi la trasformata di “checkSevereErrors” si occupa, oltre che di caricare l'errore come “bloccante”, anche di recuperare ed assegnare il codice errore appropriato, mentre per gli errori “non gestiti” viene semplicemente inserito un errore bloccante con codifica generica di errore non gestito.

Le trasformate non vengono riportate nel dettaglio in quanto sono piuttosto banali.

5.3.3 jobExtractDataFromXmls



Il job di "Extract Data From Xmls" si occupa dell'estrazione dei dati relativi ai vari flussi dai files XML sorgenti; l'estrazione avviene mediante 7 iterazioni successive dei vari files per estrarre i diversi tipi di dati (poiche' lo step che effettua l'estrazione dati da XML permette l'iterazione di un solo sotto albero alla volta, mediante XPath), nell'ordine:

- Giacenze (STOCK)
- Movimenti esterni (EXTERNAL_MOVEMENT)
- Movimenti interni (INTERNAL_MOVEMENT)
- Testate di vendita (SALE_HEADER)
- Righe di vendita (SALE_ROW)
- Cancellazioni di vendita (SALE_DELETE)
- Ammontare giornaliero (SALES_AMOUNT)

L'elaborazione avviene sequenzialmente, e l'unico punto di interesse in cui viene effettuato un controllo particolare e' subito a seguito dell'estrazione delle giacenze dai files; in questo passaggio viene controllato se sono state effettivamente estratte delle giacenze da tutti i files sorgenti, e tutti quelli che non ne contenevano vengono scartati (per essi viene inoltre registrato un errore bloccante sulla tabella BLOCKING_ERROR).

5.3.4 checkUnknownStores

La trasformata di “checkUnknownStores” si occupa di controllare l'eventuale presenza di negozi non codificati in anagrafica all'interno dei dati appena estratti dai files XML; nel caso in cui ne dovesse riscontrare la presenza, l'intero file che li contiene viene marcato come errato, e di conseguenza i dati in esso contenuto non verranno esportati (restano comunque disponibili alla consultazione).

Il controllo viene effettuato in questa fase poiche' non e' possibile effettuare il controllo prima di aver effettivamente estratto i dati dai files XML.

Anche questa trasformata non viene riportata nel dettaglio in quanto piuttosto banale.

5.3.5 joinXmlDataAndErrors

La trasformata di “joinXMLDataAndErrors” si occupa del caricamento e della transcodifica degli errori dalle tabelle temporanee in cui vengono momentaneamente inseriti durante la fase di validazione verso le tabelle definitive collegate ai dati estratti.

Per la transcodifica ci si appoggia alla tabella ERROR_TRANSCODE definita nel capitolo precedente, sulla quale viene effettuato un lookup (con caching abilitato per ridurre gli accessi al database) in modo da stabilire la corretta codifica da attribuire all'errore.

La transcodifica viene effettuata in parallelo per tutte le tabelle coinvolte.

Il dettaglio della trasformata di joinXMLDataAndErrors non viene riportato per ragioni di spazio.

5.3.6 handleSalesDeletes / jobHandleRetransmissions

Rispettivamente questa trasformata e questo job si occupano di gestire due casistiche particolari che possono presentarsi nei files XML: le cancellazioni di vendita e le ritrasmissioni.

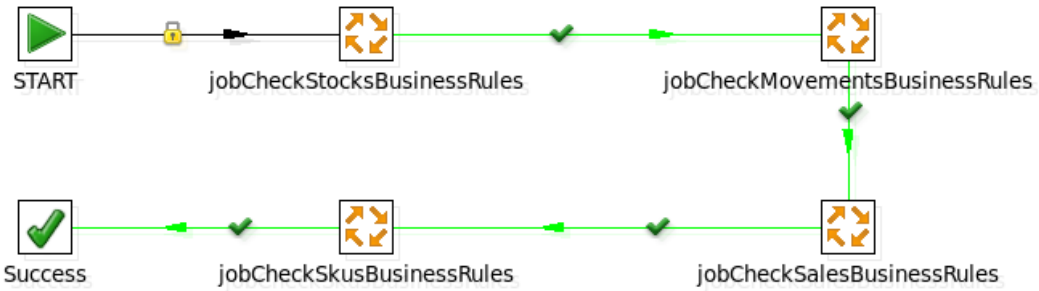
Per quanto riguarda le cancellazioni di vendita, sappiamo che esse sono esplicite negli XML e sono identificate dai dati di tipo SALE_DELETE; la presenza delle stesse porta alla bonifica (annullamento) di tutti gli errori collegati alla testata di vendita indicata nella cancellazione di vendita, a prescindere dalla data in cui e' stata trasmessa, ed ai fini pratici si traduce nell'impostazione a “N” del flag “valid” sulle tabelle degli errori coinvolti (ERROR_SALE_HEADER ed ERROR_SALE_ROW), ovviamente solo per i dati precedenti alla trasmissione corrente.

Per quanto riguarda invece le ritrasmissioni, sappiamo che esse sono implicite e riconoscibili solamente dalla presenza nel database di righe aventi lo stesso identificativo dei nuovi dati appena caricati; il trattamento e' identico a quello

previsto per le cancellazioni, e comporta l'invalidazione di tutti gli errori legati ai dati che rispettano il vincolo sopra esposto.

Il dettaglio della trasformata di handleSalesDeletes e del job di handleRetransmissions non viene riportato per ragioni di spazio.

5.3.7 jobCheckBusinessRules



Il Job di "Check Business Rules" si occupa di controllare le regole di business che sono state definite per le varie tipologie di dati estratte; come si puo' vedere, non fa' altro che richiamare, uno alla volta, tutti i sotto job specializzati nell'applicazione delle regole di business del peculiare tipo dato.

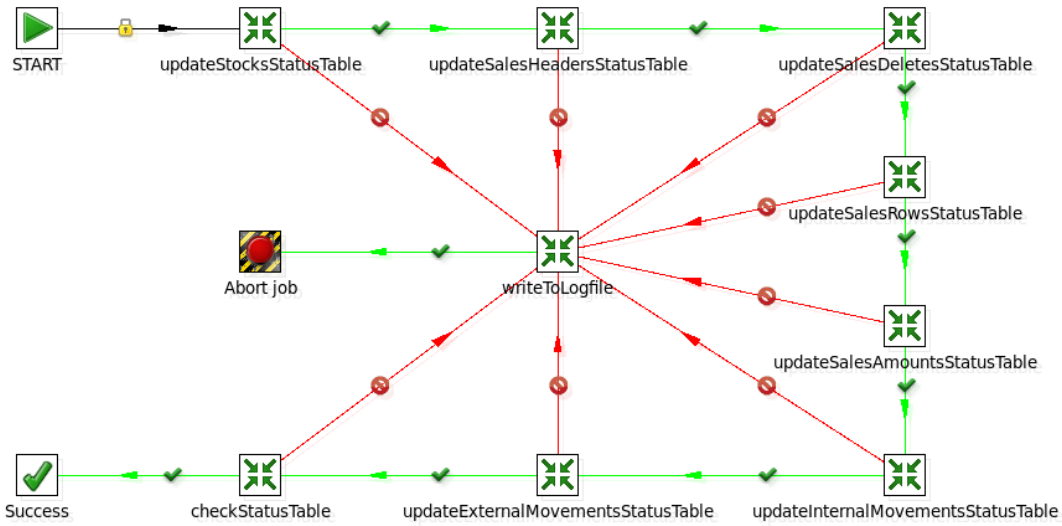
Citiamo a titolo d'esempio alcuni dei controlli che vengono effettuati sui dati:

- *GIACENZE / controllo taglia e colore articoli*: viene verificato che, per tutti gli articoli di tipo 'A', la taglia ed il colore siano diversi dalle stringhe '???'; in caso contrario, le righe che non rispettano la condizione vengono marcate con un codice di errore apposito
- *MOVIMENTI / data movimento e data consegna*: viene verificato che, per tutti i movimenti, la data del movimento sia >= della data di consegna; nel caso non venga soddisfatta la condizione, le righe vengono marcate con un apposito codice d'errore
- *VENDITE / valore totale testata*: viene verificato, per ogni testata di vendita, che il valore totale di vendita in essa riportato, al netto di eventuali sconti, sia pari alla somma dei valori riportati nelle singole righe di vendita; in caso contrario, tutte le righe (sia testata che dettagli) vengono marcate con un apposito codice d'errore
- *SKUS (EANS) / codice tipo ean*: viene verificato, per tutte le anagrafiche che riportano fra i loro campi l'ean, che il codice tipo ean sia coerente per i != 'A'; in particolar modo vengono recuperati, un'anagrafica per volta, tutti gli ean con codice != 'A' e viene verificato su tutte le altre anagrafiche che essi non

compaiano con codice = 'A', ed in caso contrario vengono marcati con un apposito codice d'errore

Come si puo' dedurre dalla descrizione delle operazioni effettuate, si tratta esclusivamente di logiche applicate sui valori dei dati estratti.

5.4 jobUpdateStatusTable



Il compito del Job di “Update Status Table” e' quello di aggiornare lo stato dei vari flussi riportato nella tabella di stato “FLOW_STATUS”, la quale, come accennato nel capitolo precedente, memorizza per ogni giorno / negozio / flusso lo stato dello stesso (se presente o mancante, se presenta errori o ne ha presentati in passato, etc).

L'aggiornamento avviene considerando una tabella di dati alla volta, ed al termine viene effettuata un operazione di “check” della tabella di stato, operazione che verra' meglio approfondita in seguito.

Scendendo piu' nel dettaglio delle operazioni effettuate dalle varie trasformate di aggiornamento stato, possiamo dire che hanno tutte la stessa struttura e seguono i seguenti passaggi:

- recuperano le coppie giorno / negozio caricate nel giro di elaborazione corrente mediante query sui dati, integrando ad esse un flag aggiuntivo che indica se per tale coppia giorno / negozio sono presenti errori nel flusso appena caricato
- controllano la relativa entry sulla tabella di stato; se non era presente alcun dato, allora si procede semplicemente registrando una nuova entry che riporta

lo stato appena rilevato (OK se non ci sono errori, OPEN_ERRORS se invece ce ne sono), altrimenti procede come segue

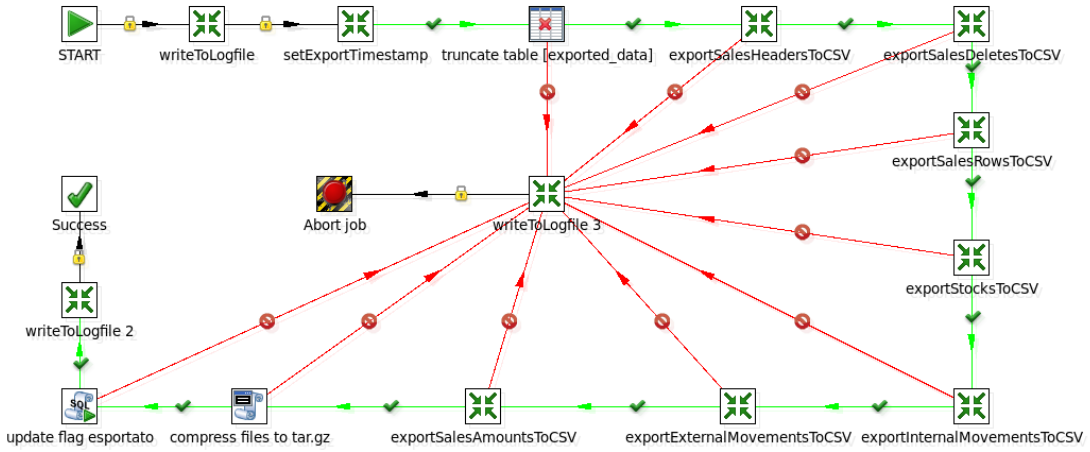
- nel caso in cui fosse gia' presente un entry nella tabella di stato, prende in considerazione lo stato precedente; nel caso in cui fosse OK, CLOSED_ERRORS oppure MISSING, viene semplicemente aggiornato con il nuovo stato (quindi viene commutato in OK se diverso da OK, oppure in OPEN_ERRORS, a seconda del valore del nuovo flag), mentre per il solo caso in cui lo stato era OPEN_ERRORS ed il nuovo flag indica che non sono presenti errori, prosegue con quanto segue
- nel caso fosse gia' presente una riga riportante stato OPEN_ERRORS ed il nuovo stato indica che non sono presenti errori, recupera tutte le righe di dati precedenti alle ultime caricate che presentano errori, assieme ad esse recupera tutte le nuove righe che sono state caricate durante l'elaborazione corrente (le quali a questo punto non presentano errori) ed effettua il match fra i due insieme di righe per stabilire se tutte le righe “vecchie con errori” hanno una corrispondente riga “nuova senza errori”; in caso affermativo, lo stato viene portato a CLOSED_ERRORS, altrimenti resta OPEN_ERRORS se almeno una riga di dati “vecchi con errori” non “viene chiusa” da una nuova riga “pulita”

Al termine dell'esecuzione delle varie trasformate specializzate nell'applicare la logica sopra esposta per tutti i flussi, viene eseguita un ultima trasformata specializzata (la “checkStatusTable”), la quale opera la logica di seguito indicata:

- recupera tutte le coppie data / negozio presenti nei dati dell'ultima elaborazione in modo da stabilire chi ha inviato dati
- per ognuna di esse, controlla se sono presenti dei flussi mancanti sulla tabella di stato
- in base al risultato dei controlli precedenti, decide se aggiornare due errori particolari presenti nella tabella BLOCKING_ERRORS:
  - il primo dei due e' l'errore che rappresenta la mancanza di tutti i flussi, sempre presente di default in quanto inserito per tutti i negozio da un processo parallelo che viene eseguito ogni giorno (tutti i giorni / negozi / flussi sono mancanti fino a che non vengono trasmessi dati), ed eventualmente rimosso da questa trasformata
  - il secondo e' l'errore che rappresenta la presenza del solo flusso stock per il giorno / negozio, inserito e rimosso da questa trasformata

Cosi' facendo vengono aggiornati anche questi due particolari errori bloccanti, in modo tale da mantenere consistenti le segnalazioni riportate sulle maschere web e mediante mail di alert.

5.5 jobExportToCSV



Il Job di “Export To CSV” e' l'ultima operazione che viene eseguita dall'ETL, e consiste per l'appunto nell'esportazione dei dati elaborati, al netto di eventuali errori di validazione e/o derivanti da regole di business, in files in formato CSV, i quali vengono poi depositati su di una cartella di interscambio monitorata dall'applicativo XFB che si occupa del trasferimento verso i sistemi del committente.

Le operazioni eseguite dal job sono relativamente semplici; nell'ordine:

- dopo aver scritto un messaggio su log, imposta il timestamp di esportazione in una variabile, in seguito utilizzata per aggiornare il campo “exported_timestamp” su tutte le entry coinvolte nella tabella XML_FILE
- tronca una tabella temporanea “exported_data”, nella quale vengono momentaneamente “segnati” i files XML che sono stati esportati (in modo tale da poter aggiornarne il flag “esportato” al termine dell'elaborazione)
- effettua l'esportazione dei dati in formato CSV, elaborando un flusso alla volta (preponendo il carattere di tipo record relativo al flusso) ed andando in “append” sui files generati di volta in volta (per rispettare il requisito di cardinalita' dei files, esplicitato nel secondo capitolo, paragrafo riguardante i requisiti funzionali)
- comprime i files CSV relativi all'esportazione corrente in un unico file tar.gz, ed al termine lo sposta nella cartella di deposito / interscambio
- aggiorna il flag di “esportato” per tutti i files XML di cui ha esportato i dati
- segnala su log il termine dell'esportazione

Al termine del sotto job di esportazione, il job principale segnala su log il termine dell'elaborazione complessiva, ed esce dal programma con exit code positivo,

chiudendo cosi' l'elaborazione dei files che aveva trovato sul server FTP al momento dell'avvio.

6 Le maschere web per il controllo e l'analisi dei dati

In questo capitolo verranno presentate le maschere web implementate per le esigenze del committente, le quali si suddividono principalmente in due categorie: maschere di consultazione anagrafica, utili per lo piu' all'Help Desk (per funzioni di supporto al committente), e maschere funzionali, quali il Monitor Errori ed il Monitor Statistico.

6.1 Le maschere anagrafiche

Le maschere anagrafiche sono semplici maschere di visualizzazione delle anagrafiche caricate, quindi:

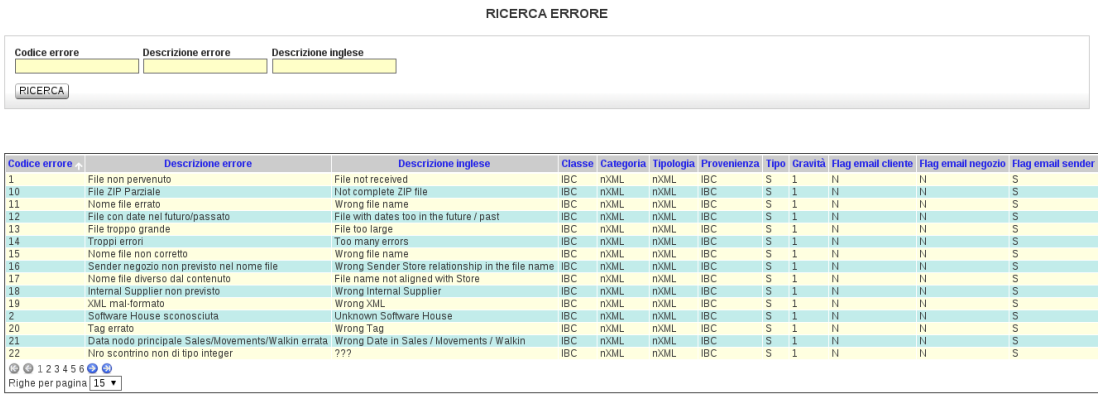
- Anagrafica “Negozio”
- Anagrafica “Cliente”
- Anagrafica “Sender”
- Anagrafica “Software House”

In aggiunta alle anagrafiche suddette (le quali vengono aggiornate giornalmente dal flusso CSV proveniente dai sistemi del committente), abbiamo una quinta maschera che viene utilizzata per visualizzazione e manutenzione della tabella anagrafica relativa agli errori (ERRORS_TABLE), la quale viene gestita solamente manualmente (non sono previsti flussi automatici).

Per quanto riguarda la prima categoria di anagrafiche, riportiamo a titolo d'esempio la maschera di visualizzazione anagrafica negozio:

La maschera prevede un semplice filtro per codice / descrizione e presenta il risultato della ricerca nella tabella sottostante (che nell'immagine e' riportata "tagliata" per ragioni di spazio); tale tipologia di maschere e' utile principalmente per l'Help Desk nelle situazioni in cui esso ha necessita' di contattare i negozi (ad esempio quando i files trasmessi presentano degli errori), per cui possono essere utilizzate le funzionalita' di ricerca offerte dalle maschere per recuperare i recapiti telefonici / fax / email del singolo negozio.

Viene riportata di seguito la maschera di visualizzazione anagrafica errori:



Anche questa maschera e' molto semplice: essa prevede un filtro di ricerca sugli attributi principali dell'anagrafica errori (codice e descrizione) e presenta il risultato della ricerca nella tabella sottostante

In aggiunta alle funzionalita' di ricerca, essa permette la modifica dei dati mediante doppio click sulla riga che si intende editare; cosi' facendo, viene aperto un popup contenente una form precompilata con i valori attuali della riga, la quale permette di modificare i singoli campi mediante inserimento di nuovi valori e pressione del tasto "Salva", oppure annullare le modifiche finora effettuate mediante pressione del tasto "Annulla".

6.2 Le maschere funzionali

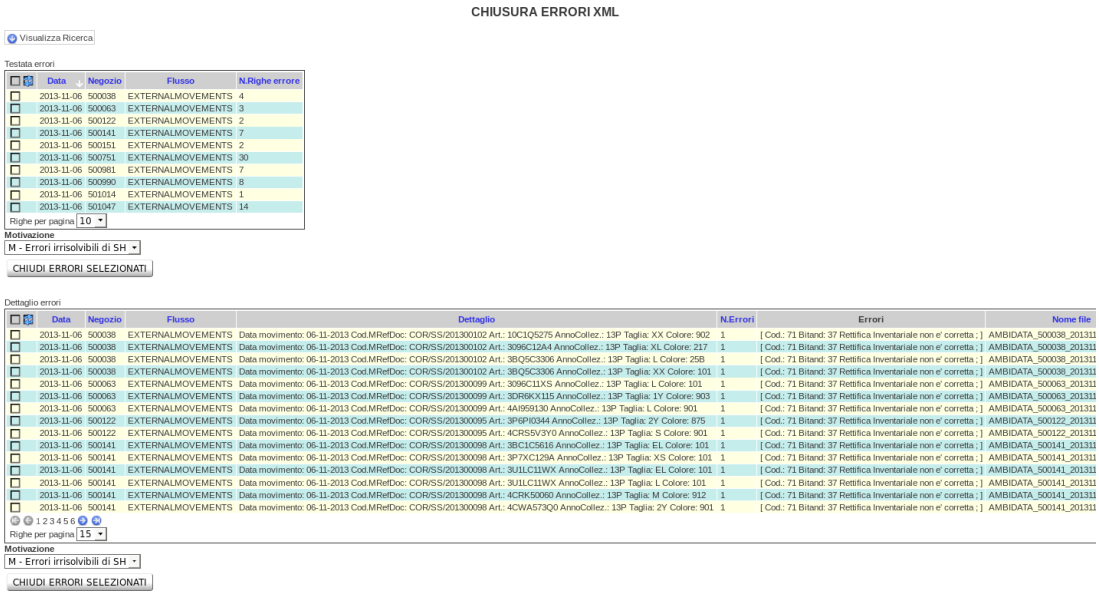
Le maschere funzionali sono quelle che permettono di analizzare piu' nello specifico i dati caricati dagli XML, ed eventualmente di modificare gli errori ad essi collegati.

Questa tipologia di maschere si puo' suddividere in due categorie:

- Maschere di chiusura errori, di cui fanno parte:
  - Maschera di chiusura errori XML
  - Maschera di chiusura errori bloccanti
- Maschere di "monitoraggio", di cui fanno parte
  - Maschera di monitor errori
  - Maschera di monitor statistico

Presentiamo quindi le varie maschere appartenenti alla categoria, nell'ordine in cui sono state elencate:

6.2.1 Maschera di chiusura errori XML



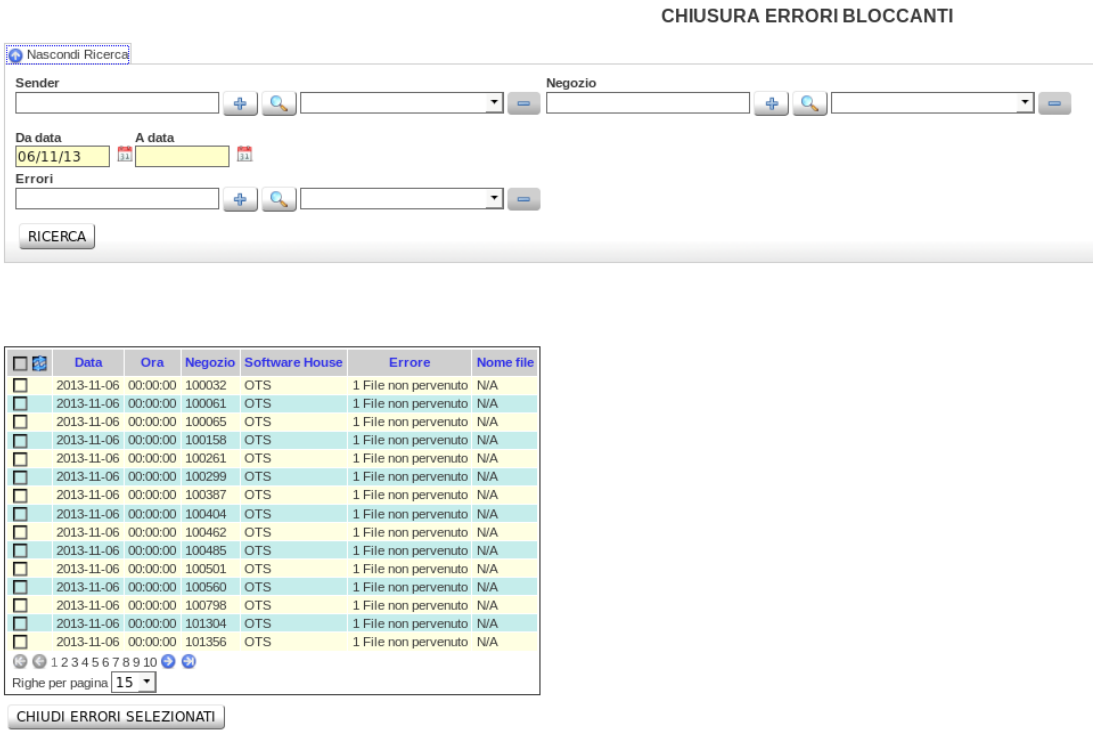
La maschera di chiusura errori XML permette la ricerca su di un range temporale, con eventuali filtri su negozio / cliente / software house, degli errori aperti sui vari flussi dati, presentandoli inizialmente raggruppati in una tabella riassuntiva (la prima tabella di "testata"), la quale riepiloga il numero di errori aperti per data / negozio / flusso.

E' poi possibile “esplodere” il dettaglio della riga selezionata nella prima tabella, andando cosi' ad aprire una seconda tabella nella parte inferiore dello schermo, la quale visualizza le singole righe che presentano degli errori, relative a data / negozio / flusso precedentemente selezionati.

E' possibile inoltre operare la chiusura degli errori, sia a livello di “testata” (andando quindi a chiudere tutti gli errori relativi ad una data / negozio / flusso), sia a livello di dettaglio (andando quindi a chiudere tutti gli errori relativi ad una singola riga di un flusso); entrambe le chiusure necessitano di una motivazione, selezionabile fra una lista statica riportata in una combobox, la quale verra' poi riportata su tutte le righe coinvolte dalla selezione.

Non e' possibile chiudere il singolo errore sul singolo elemento del flusso, sulla base delle specifiche di analisi definite inizialmente (gli errori legati ad un singolo elemento, o si chiudono tutti, oppure nessuno).

6.2.2 Maschera di chiusura errori bloccanti

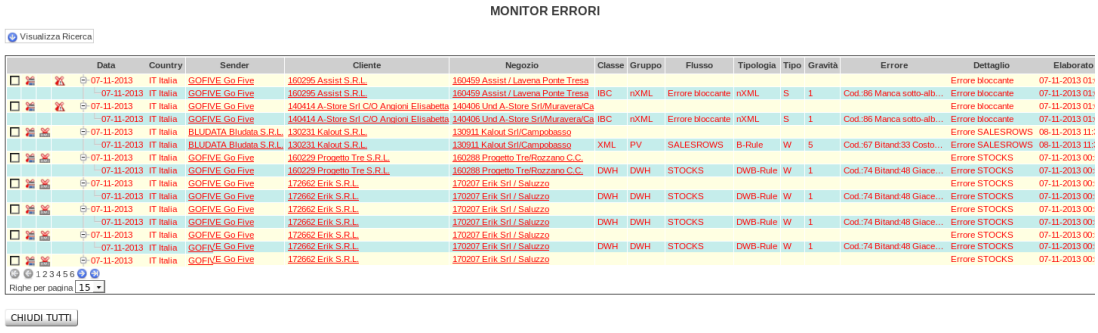


La maschera di chiusura errori bloccanti ha le stesse funzionalita' della maschera di chiusura errori XML, con l'unica differenza che la seconda delle due interroga le varie tabelle contenenti gli errori dei vari flussi, mentre questa si limita ad interrogare la tabella contenente gli errori bloccanti (BLOCKING_ERROR).

In fase di ricerca e' possibile impostare i filtri che si desidera applicare, poi premendo sul pulsante “Ricerca” comparira' il risultato nella tabella sottostante.

E' possibile poi procedere alla chiusura dei singoli errori, oppure ad una chiusura multipla selezionando le righe che si intende “chiudere” e premendo il pulsante “Chiudi errori selezionati”; cosi' facendo, tali errori non compariranno piu' nella maschera di monitor errori (di seguito illustrata), ne' nelle varie mail di alert.

6.2.3 Maschera di monitor errori



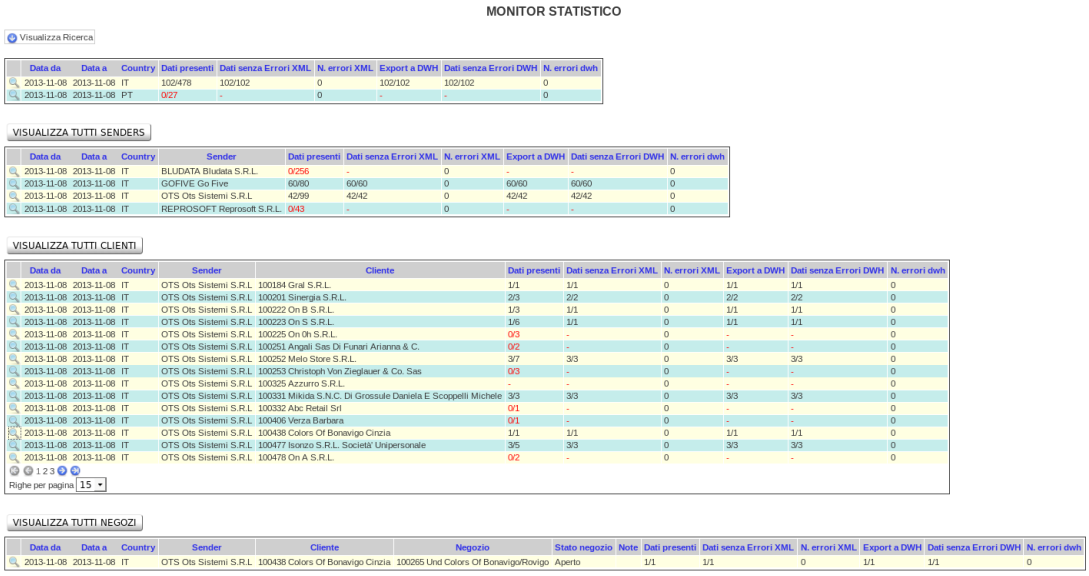
La maschera di monitor errori permette la consultazione degli errori aperti sui vari flussi per un range di date e/o per negozio, impostabili nel filtro di ricerca.

Per ogni riga di errore rilevata, viene mostrato lo storico di tutte le variazioni che la stessa ha subito nel corso delle varie elaborazioni (quindi vengono visualizzate tutte le eventuali ritrasmissioni dell'elemento); inoltre e' possibile procedere alla chiusura manuale degli errori direttamente da questa maschera, selezionando le righe che si intende chiudere.

Ogni riga puo' presentare da 1 ad N errori, riepilogati nella colonna “Errore”; posizionando il mouse su tale colonna, e' possibile visualizzare il dettaglio di tutti gli errori presenti sulla riga mediante un tooltip che viene aperto in sovrimpressione, dettagli altrimenti non visibili per intero nella tabella, ove viene applicato un troncamento degli stessi per motivi di spazio.

Di default, la maschera mostra i dati relativi agli errori del giorno corrente, ma e' possibile modificare i parametri di ricerca aprendo il pannello di apposito nella parte superiore della videata ed impostando le restrizioni che si preferisce applicare.

6.2.4 Maschera di monitor statistico



La maschera di monitor statistico riporta delle informazioni aggregate riguardanti lo stato di affidabilita' del servizio, visualizzando dati riguardanti:

- flussi presenti / flussi mancanti
- flussi con errori
- numero errori presenti
- dati esportati

Le informazioni vengono riportate su piu' livelli, a partire da quello piu' alto che rappresenta semplicemente un aggregazione per data / negozio, e scendendo poi, mediante click sull'icona a forma di lente all'inizio di ogni riga, a livello di informazioni raggruppate per sender, cliente ed infine per negozio.

Eventuali valori che superano le soglie di attenzione definite vengono riportati in colorazione rossa, in modo da attirare subito l'attenzione dell'utente a colpo d'occhio.

Come si puo' vedere, le maschere previste non sono in gran numero, poiche' le esigenze degli utenti web sono abbastanza esigue, ed inoltre gran parte dei controlli vengono effettuati direttamente sul Data Warehouse del committente mediante sistemi di consultazione dedicati implementati dallo stesso.

7 Conclusioni

Lo sviluppo del progetto ha portato a buoni risultati, soprattutto grazie ad un ottimo lavoro di analisi e di progettazione iniziale, permettendo di eseguire le elaborazioni richieste, rispettando le tempistiche massime ammesse.

Il rilascio del software ha seguito un processo graduale, per cui la prima installazione e' stata effettuata su di una macchina di test, basata su di un flusso attivato per alcuni negozi “pilota”, in modo da avere a disposizione dei dati “reali”, sulla base dei quali valutare la bonta' del software sviluppato.

In seguito ad un periodo di test di circa due settimane, durante le quali e' stato formato il gruppo di Help Desk, si e' dato inizio ad un progressivo roll-out della soluzione per vari negozi, attivando il flusso verso la nuova macchina di produzione; in tale occasione sono state eseguite le prime “elaborazioni controllate”, ossia dei processamenti dei files seguiti da vicino dal gruppo di sviluppo per controllare che non si verificassero dei problemi.

Una volta validato il flusso in produzione, si e' proceduto con lo startup di un numero di negozi sempre maggiore, ed al momento della stesura di questo documento sono ancora in corso le ultime attivazioni per quanto riguarda il flusso dati della regione europea.

Si pianifica di procedere con gli startup restanti nel corso del prossimo anno, ed entro la fine del 2014 si dovrebbe riuscire a gestire i dati provenienti dalle regioni asiatiche ed americane.

Sicuramente sono ancora presenti alcuni punti di miglioramento; in particolar modo, si sta valutando di aumentare il grado di parallelizzazione dei processi, oltre alla ristrutturazione dello schema del database, prevedendo dei nuovi indici per le tabelle piu' “corpose” (prima fra tutte quella delle giacenze).

Oltre a quanto sopra riportato, si sta pensando anche di creare delle nuove tabelle da utilizzare per raggruppare logicamente i dati, in modo da renderli piu facilmente consultabili da parte delle maschere web, e comunque da parte di tutti i processi che effettuano letture sui dati, ETL compreso.

Il software sviluppato ha ricevuto una buona accoglienza da parte del committente, il quale ha iniziato a sviluppare tutta una serie di software atti all'analisi approfondita dei dati ricevuti, i quali dovrebbero permettergli di razionalizzare ed ottimizzare il processo distributivo e di approvvigionamento della rete di vendita, al fine di evitare costi inutili e raggiungere un traguardo di maggiore fatturato a fine anno.

Bibliografia

[1] John C. Worsley, Joshua D. Drake, *Practical PostgreSQL*, O' Reilly, 2002.

[2] Matt Casters, Roland Bouman, Jos van Dongen, *Pentaho Kettle Solutions*, Wiley, 2010.

[3] Adrián Sergio Pulvirenti, *Pentaho Data Integration 4 Cookbook*, Packt, 2011.

[4] Jeffrey E.F. Friedl, *Mastering Regular Expressions, 3rd Edition*, O' Reilly, 2006.

[5] Martijn Dashorst, Eelco Hillenius, *Wicket in Action*, Manning, 2009.

[6] Karthik Gurumurthy, *Pro Wicket*, Apress, 2006.

[7] Frank Zammetti, *Practical JavaScript, DOM Scripting and AJAX Projects*, Apress, 2007.

[8] Christian Bauer, Gavin King, Gary Gregory, *Java Persistence With Hibernate*, Manning, 2006.

[9] Deepak Vohra, *Java EE Development with Eclipse*, Packt, 2012.

[10] Jonathan Hobson, *CentOS 6 Linux Server Cookbook*, Packt, 2013.