UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI SCIENZE STATISTICHE

TESI DI LAUREA TRIENNALE

IN STATISTICA E TECNOLOGIE INFORMATICHE

SpagoBI e PentahoBI: Due software Open Source per la Business Intelligence

Relatore: Prof.ssa Susi Dulli

Laureando: Roberto Marchetto

Anno accademico 2007/2008

Indice generale

1 Introduzione

Negli ultimi dieci anni si sono diffuse tecnologie di analisi e raccolta dati che costituiscono nel loro insieme il concetto di Business Intelligence. Se prima la disciplina trovava interesse tipicamente negli studi teorici o in organizzazioni con esigenze del tutto particolari, recentemente la diffusione di database gestionali e la crescente disponibilità di dati ha ampliato il numero di organizzazioni interessate all'argomento.

Parallelamente a questa diffusione sono nati software commerciali e non che riducono notevolmente l'investimento necessario, che rimane ancora significativo nei prodotti di punta delle grandi software house. L'introduzione di prodotti economici e la maturazione di progetti Open Source ha di fatto aperto le porte della Business Intelligence anche alle piccole imprese caratterizzate da esigenze e capitali limitati.

La tesi nasce dall'interesse suscitato da due progetti Open Source per la Business Intelligence, uno è SpagoBI di Engineering Ingegneria Informatica e l'altro PentahoBI della software house Pentaho. Entrambe le piattaforme verranno implementate in un contesto pratico, seguendo le effettive esigenze di una tipica PMI italiana. Non si vuole confrontare nel dettaglio le due alternative, cosa che peraltro richiederebbe delle valutazioni oggettive e una metodologia operativa troppo onerosa per due software così complessi. Piuttosto è obiettivo della tesi rilevare i loro pregi e difetti e il loro stato di evoluzione.

Una descrizione generale della Business Intelligence viene trattata nel capitolo 2. Il successivo capitolo 3 riporta una panoramica

generale, esaminando l'architettura e le funzionalità dei due software. Nel capitolo 4 si discute una implementazione di prova sia di PentahoBI che di SpagoBI, che verrà descritta passo a passo nel capitolo 5. Segue una comparativa generale fra le due piattaforme nel capitolo 6 per poi trarre le considerazioni finali nel capitolo 7.

2 La Business Intelligence

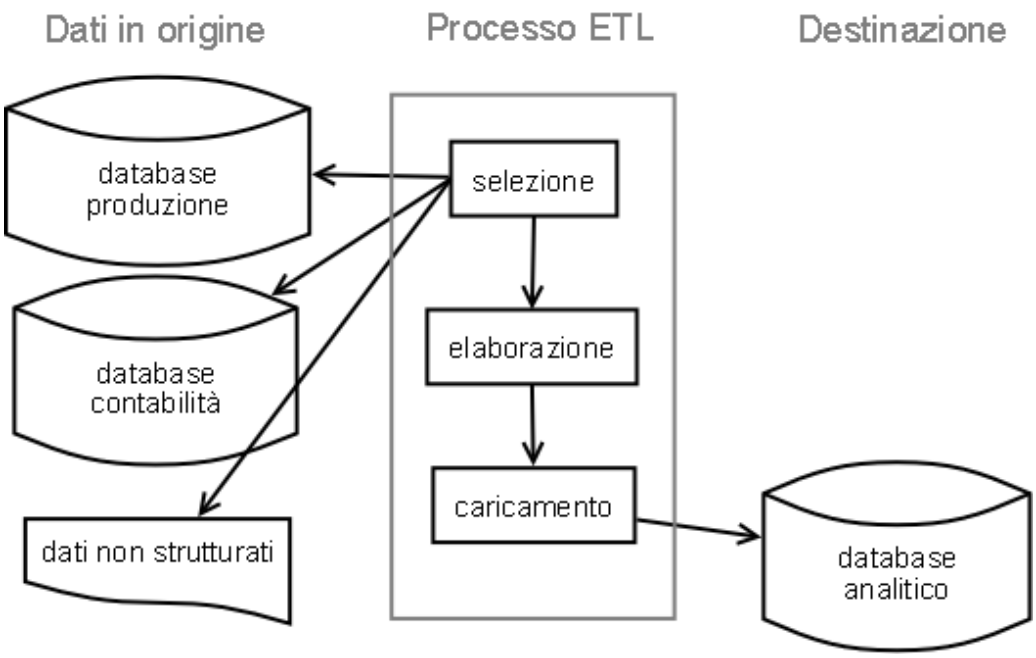
Se consideriamo la quasi totalità delle aziende che fanno uso di strumenti informatici troveremo almeno un database di supporto all'attività operativa. Più comunemente saranno presenti diversi database dedicati a funzioni specifiche come ad esempio alla fatturazione, al processo operativo, all'amministrazione, eccetera. Si tratta di database isolati l'uno dall'altro nel senso che lo scambio di dati è ridotto, la ridondanza delle informazioni elevata e i dati sono spesso disallineati. Poi ci possono essere informazioni provenienti da fonti esterne all'azienda, salvati su fogli di calcolo (Microsoft Excel, OpenOffice Calc, ecc) o più raramente provenienti da web services (SOAP).
La struttura informatica si complica maggiormente se l'azienda è di grandi dimensioni con filiali sparse per il mondo, ognuna delle quali con differenze organizzative.
Se i database operazionali sono adatti al loro campo specifico di utilizzo, nasce l'esigenza da parte dei manager aziendali di avere una visione sintetica e di insieme, pur potendo arrivare ad un certo grado di dettaglio. In pratica appaiono nuove problematiche di tipo tecnologico e nuove soluzioni legate al fatto che:

a) i dati provengono solitamente da sorgenti eterogenee come database, documenti di testo, file XML
b) questi dati vanno estratti e rielaborati per adattarli ad uno

schema comune

c) a parte l'aggiornamento del database, i dati raccolti sono acceduti in sola lettura e sono tipicamente molto numerosi
d) i dati sono perlopiù di tipo storico, organizzati per argomento tematico
e) gli utenti finali hanno bisogno di strumenti di analisi flessibili, semplici e sintetici, che consentano esplorazioni sui dati non completamente definite a priori

Estendere i database preesistenti è il più delle volte proibitivo per i vincoli imposti dalle case produttrici e risulta di gran lunga più flessibile separare i database operazionali da un insieme di strumenti software con fini puramente analitici. Queste problematiche erano piuttosto rare nei primi elaboratori elettronici, ma con la diffusione dell'informatica nelle aziende e l'accumulo di dati grezzi si sono dimostrate interessanti, tanto da costituire una disciplina che prende il nome di **Business Intelligence**. Usando la terminologia della Business Intelligence, i punti a) e b) vengono risolti nella fase di **ETL (Extraction Transfromation Loading)** che si occupa del prelievo dei dati dalle sorgenti, la loro elaborazione e pulizia, nonché del caricamento e aggiornamento del database analitico.



Il database di destinazione per la raccolta dei dati ha esigenze particolari rispetto ai comuni database relazionali, come si può vendere nei punti c) e d). Pur potendo riadattare allo scopo un classico RDBMS, si sono diffuse soluzioni specifiche per l'argomento che soddisfano il concetto di **DataWarehouse**. Più formalmente secondo la definizione di Bill Inmon un DataWarehouse è una collezione di dati in supporto al processo decisionale, dati che sono:

- orientati al soggetto, ossia l'utente finale
- nati dall'integrazione di fonti eterogenee
- dipendenti dal tempo
- non modificabili una volta inseriti

Particolarmente adatto alla memorizzazione è il modello multidimensionale, il quale rappresenta i dati come punti nello spazio definito da dimensioni. Così è possibile calcolare una informazione di sintesi come il fatturato totale (fatto) a seconda molteplici aspetti come il periodo, l'area geografica eccetera (dimensioni). Fatti e dimensioni sono tra loro collegati a formare il cosiddetto **cubo multidimensionale**.

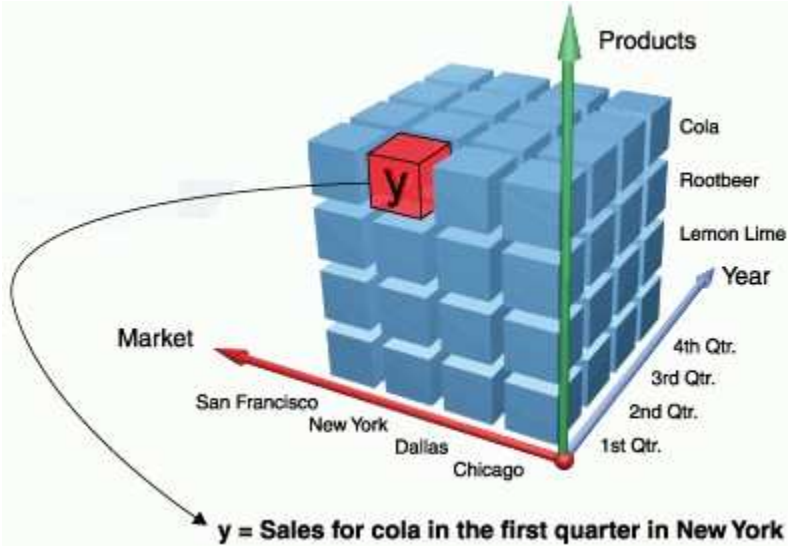


Illustrazione 1: Esempio cubo multidimensionale

Molto particolare è la gestione del tempo nei DataWarehouse, dove per ogni record inserito sono associate le informazioni temporali di registrazione. Dati aggiornati possono essere aggiunti alla base dati senza modificare quelli precedenti, così da avere a disposizione durante le analisi tutte le informazioni storiche fedeli a qualunque intervallo temporale considerato.

Uno schema multidimensionale può essere rappresentato da uno schema relazionale denormalizzato, dove i fatti sono raccolti in una tabella relazionata con le tabelle delle dimensioni seguendo uno schema a fiocco di neve o, più comunemente, a stella.

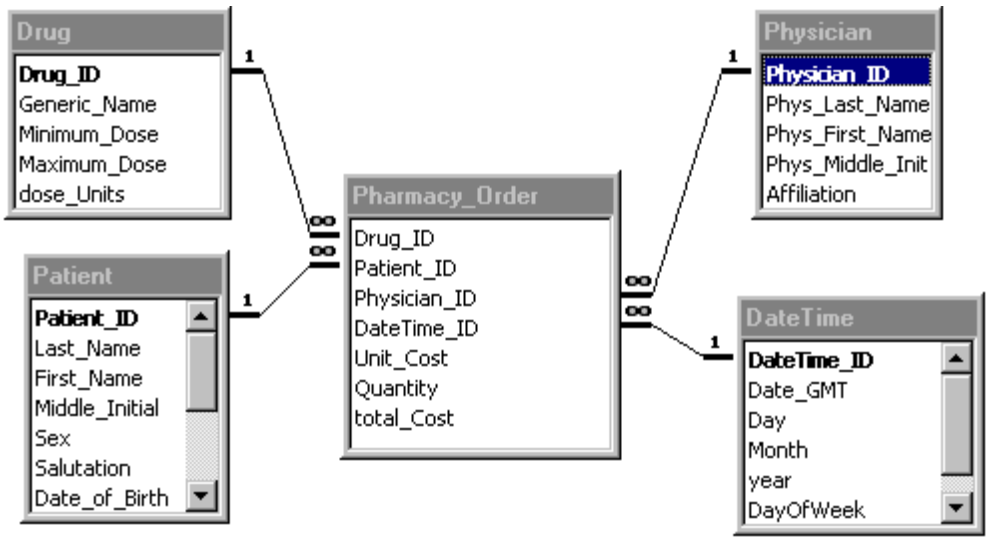


Illustrazione 2: Esempio di schema a stella

Ciò consente di immagazzinare cubi multidimensionali su comuni database relazionali sfruttando la diffusione e le capacità di archiviazione che li caratterizzano. Un modello alternativo a quello relazionale sfrutta le proprietà delle matrici sparse, più simili alla rappresentazione concettuale del cubo e con un miglioramento delle performance, a discapito di alcune caratteristiche come la capacità di archiviazione.

Il modello multidimensionale viene impiegato nella tecnologia **OLAP (Online Analytical Processing)** che consente una esplorazione dei dati seguendo le esigenze di flessibilità e semplicità rilevate nel punto e). Se il cubo è realizzato da un modello relazionale si parlerà più precisamente di Relational-OLAP, mentre si parla di Multidimensional-OLAP quando si utilizza la tecnologia a matrice sparsa e Hybrid-OLAP con una combinazione delle due tecniche.
L'analisi OLAP nasce come strumento per scoprire il comportamento generale dei fatti a seconda delle dimensioni, pur consentendo di andare nel dettaglio a isolare informazioni particolareggiate.
Completano le esigenze di analisi della Business Intelligence

strumenti quali cruscotti, grafici, report e, nei casi più avanzati, data mining.

3 Due soluzioni Open Source

3.1 Piattaforma Pentaho

La software house Pentaho è specializzata in strumenti completamente Open Source per la Business Intelligence e negli ultimi anni ha integrato vari progetti di terze parti, proponendo al momento attuale un'ampia offerta di software come:

- Mondrian (Server Relational-OLAP)
- JFreeReport (Software di reporting)
- kettle (Software di E.T.L.)
- Weka (Data Mining)
- Pentaho BI (Piattaforma per la Business Intelligence)

Pentaho BI in particolare è la piattaforma di Business Intelligence giunta alla versione 1.2 stabile e 1.5.4 beta, nata dall'integrazione dei progetti sopra elencati e il supporto a componenti di terze parti quali:

- BIRT (Software di reportistica realizzato per l'IDE Eclipse)
- JasperReports (Altro software di reportistica molto diffuso)
- JPivot (Interfaccia grafica via web per analisi OLAP)
- JFreeChart (Strumento per la generazione di grafici)

Il cuore di Pentaho BI gira in tecnologia JSP (Java Servlet Pages) su Application Server Java JEE a scelta tra JBoss e Tomcat[1]. Questo significa che la navigazione all'interno del portale avviene tramite tecnologia web.

Illustrazione 3: Interfaccia grafica di Pentaho BI

Pentaho non dispone di un proprio database per la persistenza[2] ma si appoggia via JDBC ai più diffusi database relazionali, il che rende teoricamente possibile utilizzare un qualsiasi database con supporto JDBC (ad esempio Oracle, Interbase, Firebird, Microsoft SQL Server,

1 La natura basata su Java di Pentaho e le scelte architetturali rendono possibile l'utilizzo di application server alternativi, non ancora implementati. Su Tomcat si noti che non è possibile utilizzare una particolare modalità visiva della piattaforma per l'assenza del supporto diretto allo standard JSR-168.

2 Per persistenza si intente il salvataggio dei dati su memoria secondaria non volatile, utilizzando sistemi quali database relazionali/multidimensionali/ad oggetti, file fisici, ecc.

MySQL, IBM-DB2, eccetera).

I documenti analitici[3] vengono salvati in una semplice repository su disco fisso. Nella versione Professional di Pentaho BI è previsto il supporto a repository su database relazione MySQL, migliorando le prestazioni e l'affidabilità generale del sistema.

L'architettura che ne risulta è ricca e complessa, come si può vedere nella Illustrazione 4.

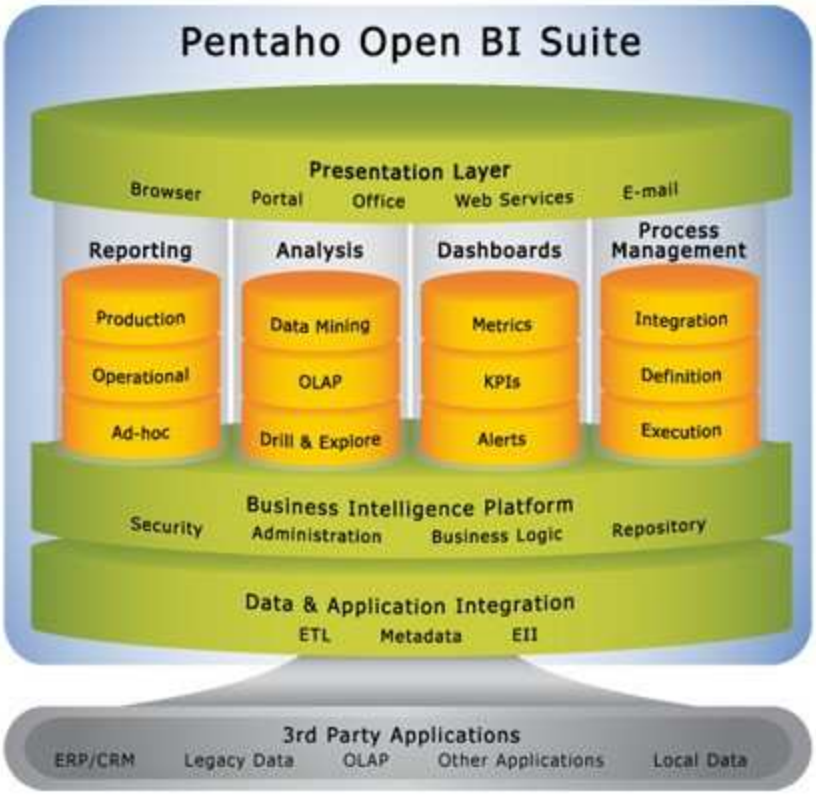


Illustrazione 4: Architettura di PentahoBI

Oltre alla visualizzazione classica Pentaho BI prevede una modalità di funzionamento a portale, basata sull'uso di Portlet JSR-168. La modalità a portale, di cui si ha un esempio nell'Illustrazione 5,

3 Per documenti analitici si intende gli applicativi che permettono di fare Business Intelligence come grafici, report, interfacce OLAP e altri strumenti accessori o di analisi

funziona al momento solo con JBoss Portal che è l'unico degli application server previsti da Pentaho aderente allo standard JSR-168. La scarsa documentazione e supporto della modalità a portale, unita alle competenze richieste nell'uso di JBoss Portal fanno sì che questa particolare interfaccia venga tralasciata nell'ambito della tesi perché richiederebbe uno studio finalizzato alla programmazione di JBoss e ai dettagli implementativi di Pentaho più che all'esame di una funzionalità utilizzabile da un tipico utente della piattaforma. Va inoltre aggiunto che la modalità completa e predefinita di Pentaho BI è l'interfaccia classica e non a portale, inoltre quest'ultima nasconde alcune funzionalità rilevanti come l'amministrazione e l'esplorazione delle repository degli applicativi.

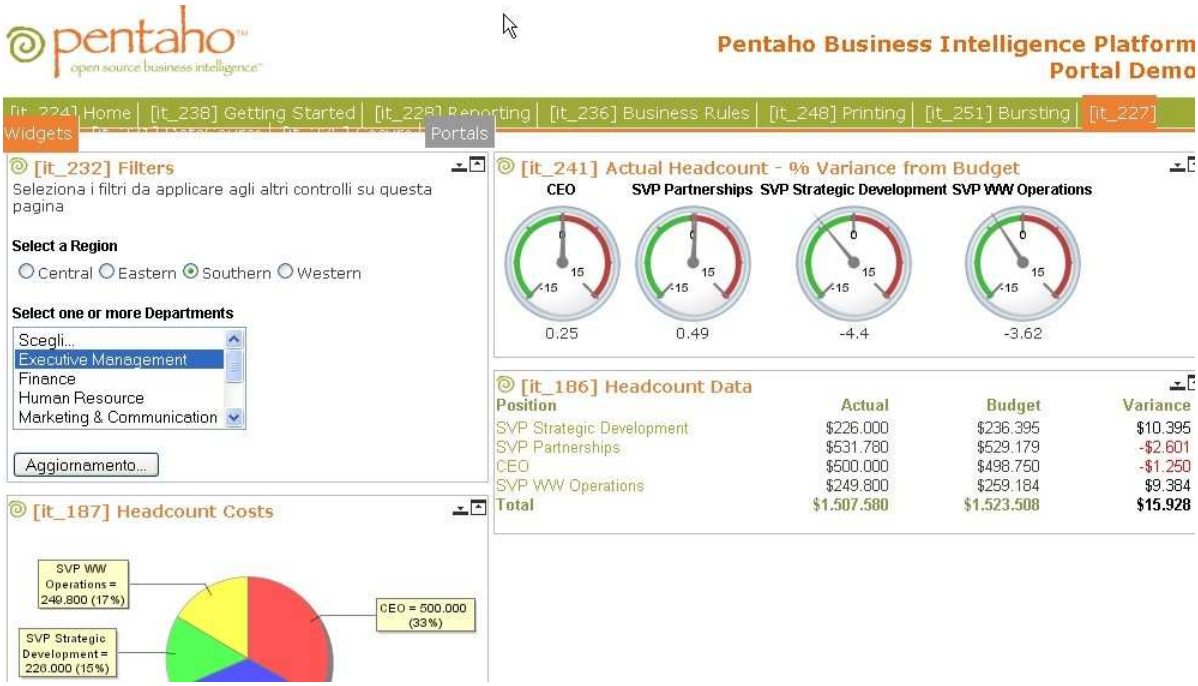


Illustrazione 5: Modalità di funzionamento a portale

Pentaho BI viene distribuito anche in versione non Open Source a pagamento con strumenti software aggiuntivi e il supporto tecnico diretto.

3.1.1 Sviluppo e amministrazione

I documenti analitici si basano sul formato Pentaho Action Sequence, ossia file XML che vengono caricati e interpretati dalla piattaforma. Una action sequence definisce i parametri di ingresso e le azioni da compiere, il tutto in una logica procedurale. Possono in tal modo venire eseguite azioni in sequenza come la visualizzazione di una form per la rischiesta di dati in ingresso e la seguente stampa di un report.

Per creare i documenti analitici è disponibile lo strumento Pentaho Design Studio, un plug-in per il noto IDE Eclipse che si propone come ambiente di riferimento per la parte di sviluppo della piattaforma.

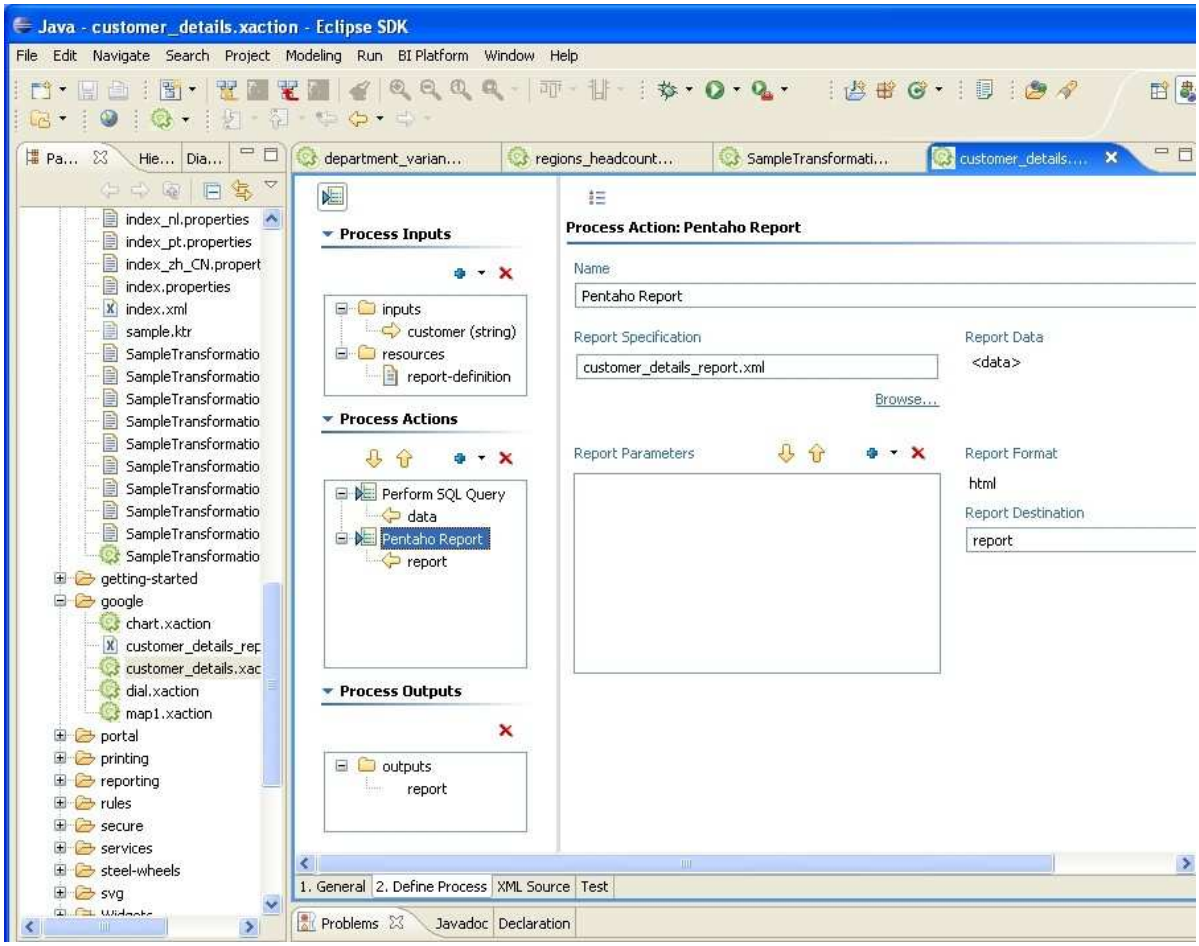


Illustrazione 6: Pentaho Design Studio

La logica funzionale è definita all'interno dell'area “Process Actions” del documento analitico elencando le operazioni che si vogliono eseguire. Tipiche azioni sono l'esecuzione di query, report, processi ETL, analisi OLAP, grafici eccetera. Sono disponibili inoltre un ampio set di funzionalità accessorie come stampa di messaggi di debugging, esecuzione di codice JavaScript, spedizione dell'output via email, eccetera.

Possono essere aggiunti parametri che provengono da diverse fonti, ad esempio passati per URL, inglobati nella sessione corrente o salvati nell'ambiente di esecuzione. Interessante è la possibilità di combinare Action Sequence in modo sequenziale così che l'output di una Action Sequence diventi l'input di un'altra producendo applicazioni di notevole flessibilità.

Oltre alla logica di funzionamento, una Action Sequence riporta anche le informazioni descrittive (nome, descrizione, icona) che appaiono all'utente finale all'interno della piattaforma.

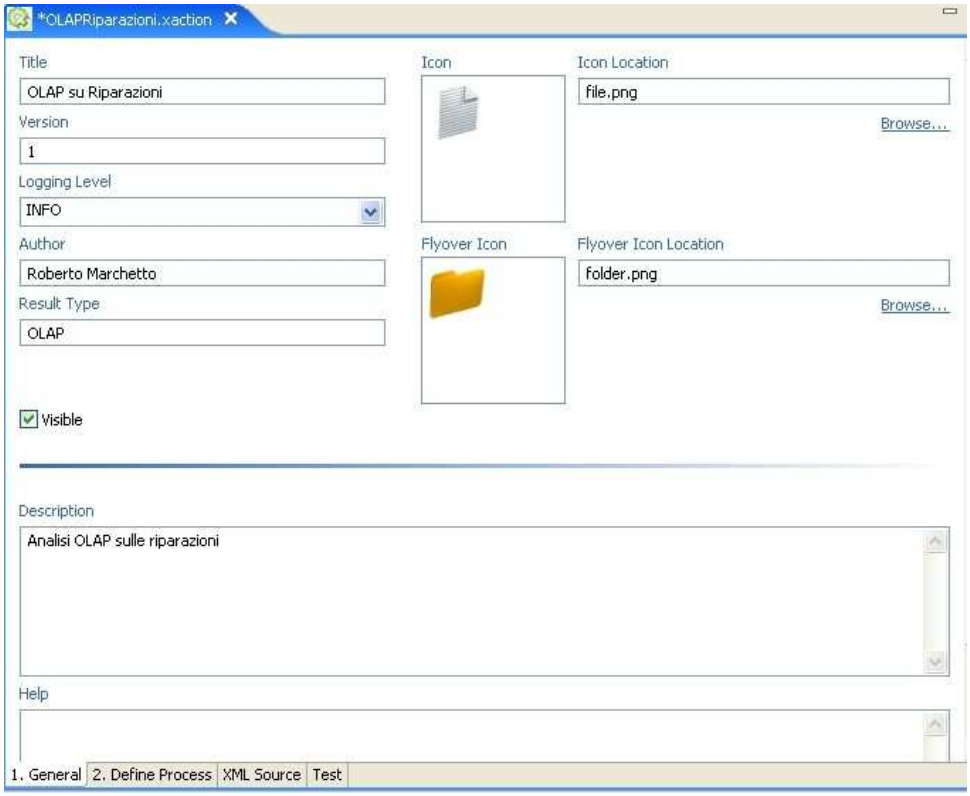


Illustrazione 7: Informazioni generali della Action Sequence

Per alcune applicazioni come report o schemi Mondrian è richiesta la definizione dei template con strumenti esterni, per poi inglobare il file di configurazione all'interno di una Action Sequence.

3.1.2 Reportistica

E' possibile utilizzare report generati dai software

- JasperReports
- BIRT
- JFreeReport

Tutti e tre permettono la stampa nei comuni formati HTML, PDF e XLS. C'è inoltre la possibilità di visualizzare grafici, eseguire funzioni di raggruppamento e parametrizzazione, opzione questa che permette di pilotare il report direttamente da un documento analitico.

3.1.2.1 JasperReport

JasperReports è una libreria scritta in Java che legge un file tracciato con estensione .jrxml e produce il rispettivo report. Il file .jrxml è in formato XML opportunamente strutturato, che può essere anche compilato garantendo la protezione del codice sorgente nel rilascio dei report.

Invece che scrivere a mano il file .jrxml viene in aiuto il completo strumento di design iReport.

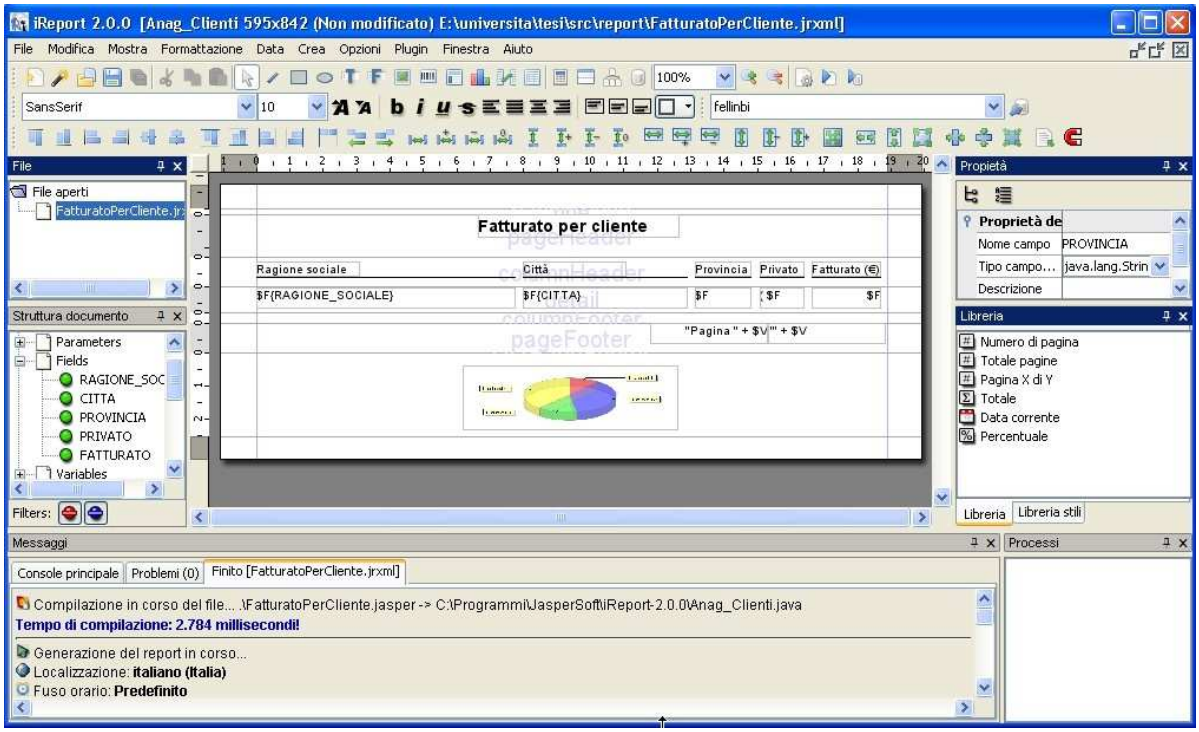


Illustrazione 8: iReport, lo strumento per il design di Jasper Reports

La realizzazione di report anche complessi con iReport diventa semplice e intuitiva. In pratica si tratta di definire la query dei dati sorgente e trascinare all'interno dell'area i vari componenti come campi di testo, valori di database, tabelle, grafici, eccetera usando uno stile di sviluppo WYSIWYG[4].

4 Acronimo di What You See Is What You Get, ossia quello che si vede è quello che si

La logica di funzionamento è a bande, dove ogni banda stampa i componenti che contiene al verificarsi della condizione associata, che può essere ad esempio l'arrivo dell'intestazione, l'inizio di un gruppo, una riga del risultato della query, eccetera.
JasperReport rispetto alle alternative per la reportistica disponibili in Pentaho è quello che offre maggiori formati di esportazione, contando oltre ai classici HTML, PDF e XML anche Excel (fogli multipli e singoli), CSV, RTF e TXT.

Come database può essere utilizzato qualunque database relazionale con supporto JDBC o anche Hibernate[5]. I valori dei campi di testo possono essere modificati dinamicamente usando espressioni Groovy, un linguaggio di scripting di compatibile con Java.

Il prodotto si dimostra molto completo, curato nei particolari, competitivo anche rispetto ai software commerciali più rinomati, forte anche della maturità e supporto acquisito nel tempo.
Nel paragrafo 5.4.1 si illustrerà la creazione pratica di un report con iReport.

3.1.2.2 BIRT

Molti probabilmente conosceranno già l'ambiente di sviluppo Eclipse, uno degli IDE (Integrated Development Enviroment) più popolari nella comunità Open Source. BIRT (Business Intelligence and Reporting Tool) è il progetto che conferisce ad Eclipse

ottiene. Si tratta di una tecnica di sviluppo di applicazioni con un approccio fortemente visuale, anticipando nella fase di design quello che risulta ad applicazione avviata.

5 Hibernate è uno dei framework Java più famosi per la persistenza degli oggetti (Object/Relational mapping). In pratica gli oggetti software vengono archiviati e quindi recuperati da database relazionali gestendo tecniche evolute di caching e ricerca.

funzionalità di reportistica.

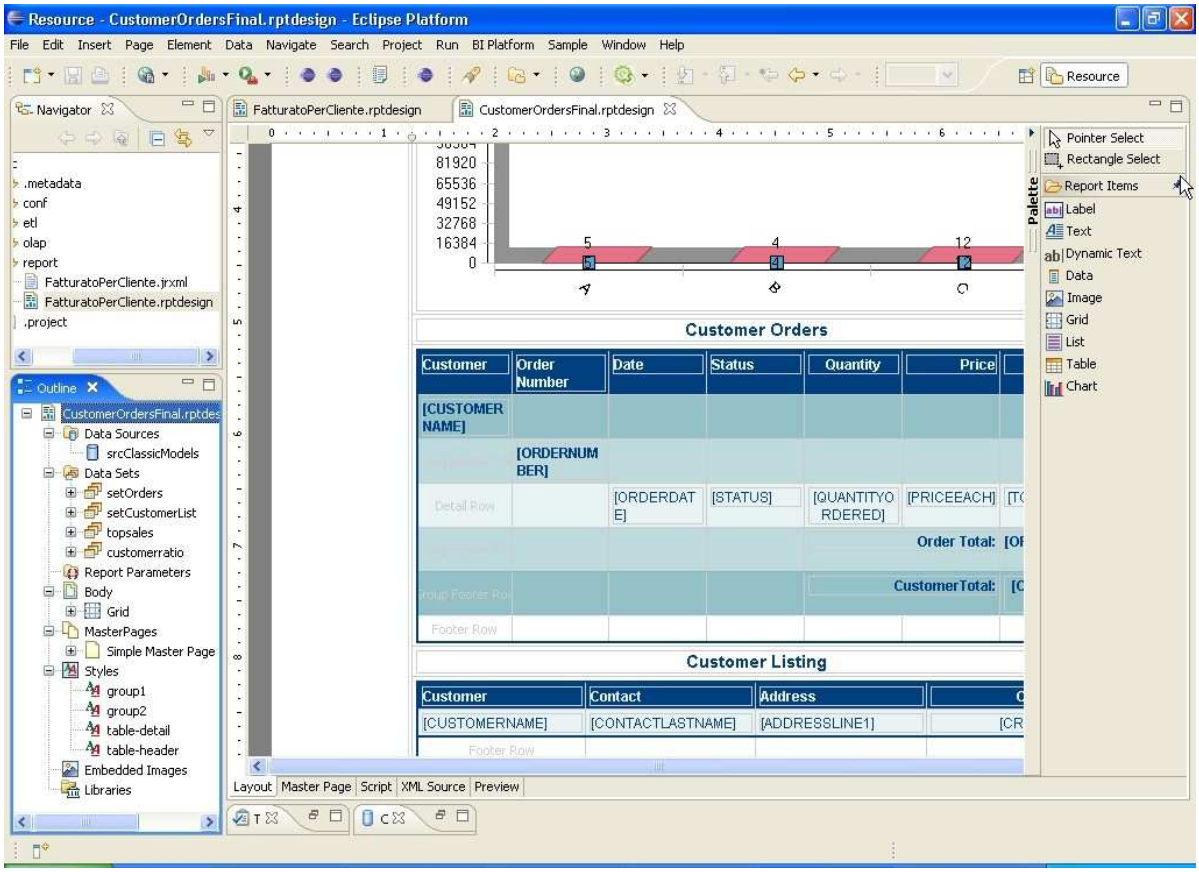


Illustrazione 9: BIRT all'interno dell'IDE Eclipse

A chi è abituato alla logica di design dei report a bande in BIRT noterà che prima di posizionare qualunque componente è necessario definire una griglia, concettualmente simile a una tabella HTML. L'utilizzo dei CSS (Cascading Style Sheets) per la formattazione grafica dei componenti e di un linguaggio per la generazione di valori dinamici simile a JavaScript rende BIRT un prodotto intuitivo ai programmatori web.

Le funzionalità rilevanti per un software di reportistica ci sono tutte, dai grafici ai campi calcolati e la parametrizzazione del comportamento del report.

Salvato il report dal designer quello che si ottiene è un file XML con estensione .rptdesign, interpretato dalla libreria Java di BIRT per

produrre report in formato HTML, PDF o CSV.

BIRT, progetto relativamente giovane rispetto a JasperReport, fa dei suoi punti di forza l'integrazione con Eclipse, la completezza di funzionalità e un notevole supporto da parte della comunità Open Source.

Nel paragrafo 5.4.2 verrà illustrato un caso pratico di utilizzo di BIRT.

3.1.2.3 JFreeReport

JFreeReport è il progetto acquisito dalla Pentaho e integrato all'interno della piattaforma PentahoBI per diventarne il software di reportistica di riferimento, pur consentendone un uso stand alone come le controparti JasperReport e BIRT.

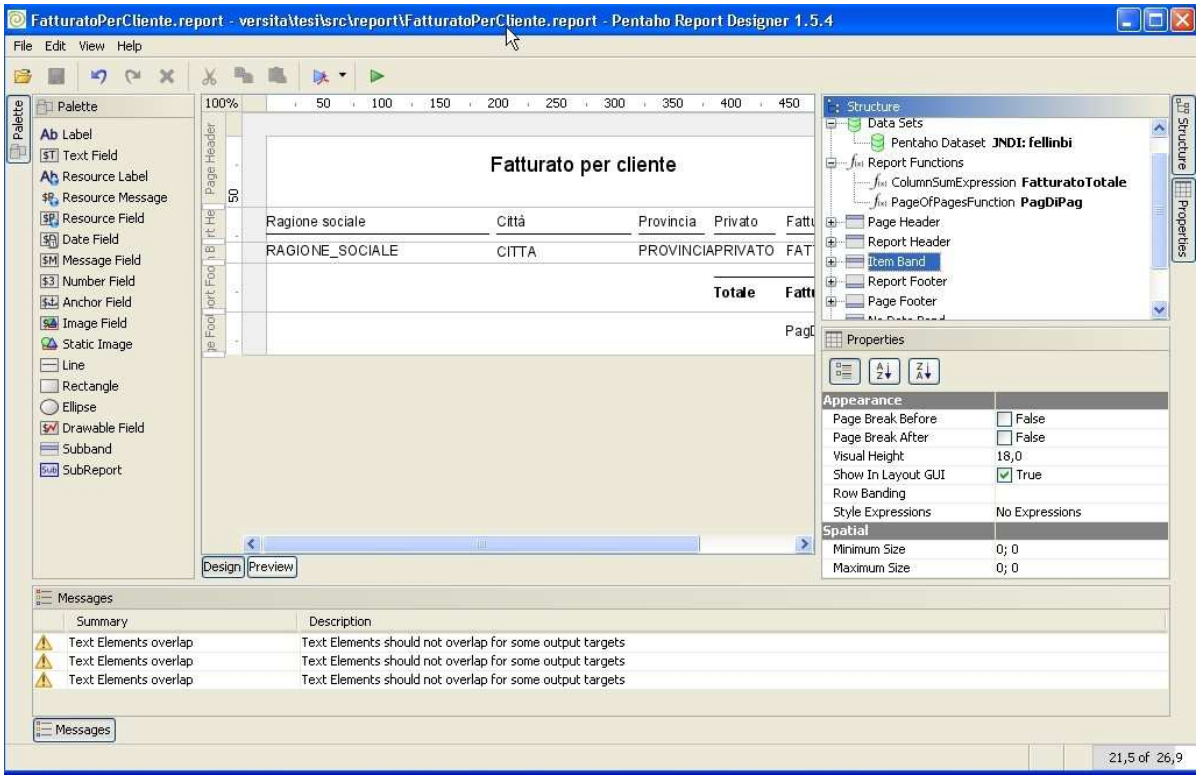


Illustrazione 10: Videata di JFreeReport

Dopo un po' di utilizzo si nota che siano facilmente disponibili le funzionalità più importanti, se si tralascia il supporto a grafici che richiede il passaggio dell'immagine da una libreria esterna come JFreeChart. Mancano invece e ne penalizzano la funzionalità gli strumenti accessori come l'editor assistito per la creazione di espressioni, la scelta dei valori delle proprietà da menù a tendina o comunque il controllo dei valori immessi. Tale lacuna unita alla scarsa documentazione di Pentaho Design Studio rallenta la creazione ed il collaudo dei report. Si tratta comunque di un progetto in stato di evoluzione che potrebbe presentare nei prossimi rilasci delle migliorie significative.

JFreeReport ha come vantaggio l'integrazione con PentahoBI consentendo l'utilizzo di connessioni e schemi Mondrian. Da PentahoBI versione 1.5.0 è possibile creare report JFreeReport direttamente dalle pagine web della piattaforma con una procedura assistita, come si vede nell'Illustrazione 11. Questa modalità agevola anche la creazione di grafici sul report.

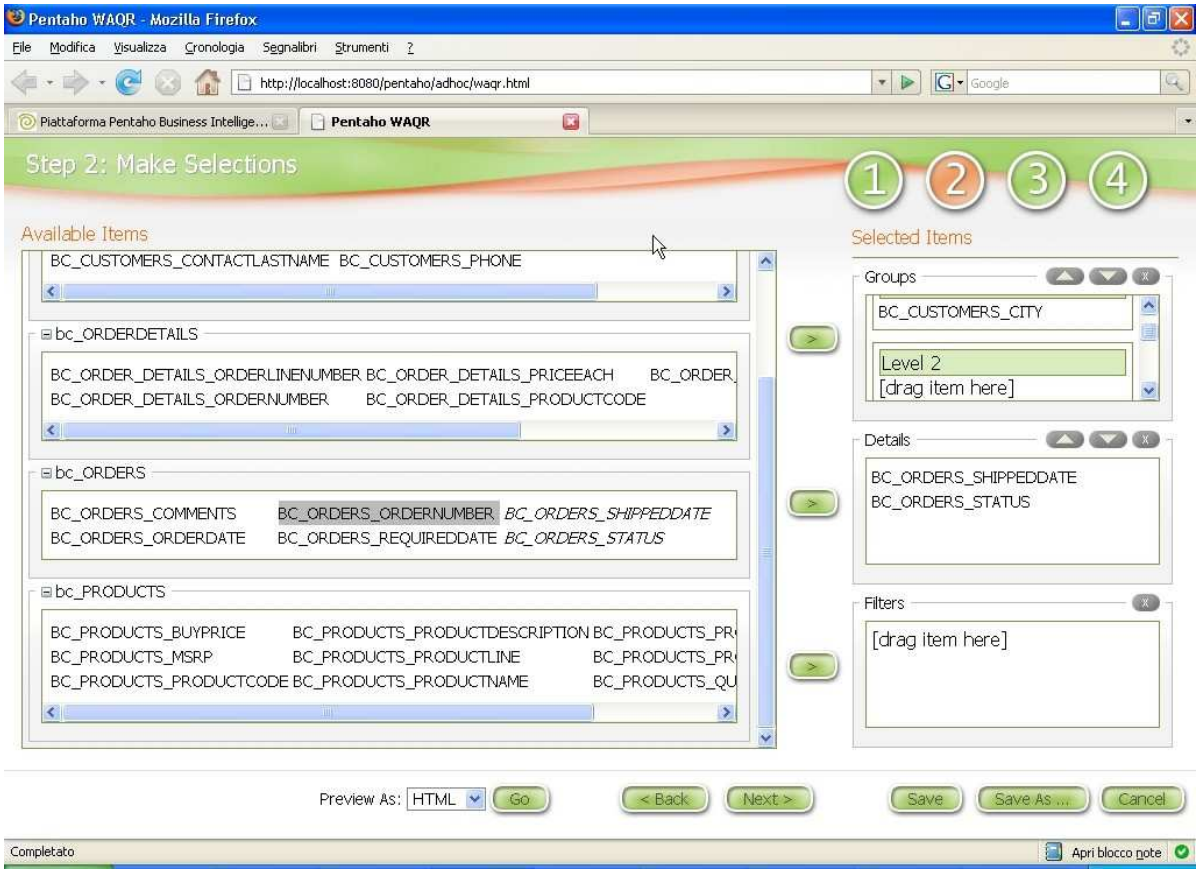


Illustrazione 11: Creazione di un report direttamente da PentahoBI

Il formato del file descrittore dei report è XML il che consente una notevole flessibilità implementativa, mentre il formato di stampa è a scelta tra XML, PDF, HTML, RTF, XLS e CSV.

3.1.3 Analisi OLAP

L'analisi OLAP si basa su due componenti, uno è Mondrian per la gestione del cubo multidimensionale, l'altro è JPivot per la parte di visualizzazione.

Mondrian è una libreria Java per la creazione di server Relational-OLAP usando database relazionali verso cui vengono mandate query SQL per generare il cubo multidimensionale. Trattandosi in pratica di un server OLAP virtuale su una base di dati relazionale le

prestazioni di Mondrian sono fortemente influenzate dalla capacità di aggregazione, caching e capienza del RDBMS sottostante.

Influiscono positivamente sulle prestazioni l'affinamento dei parametri della memoria e un buon design dello schema relazionale, che può essere a stella o a fiocco di neve. A livello di libreria Java è previsto un efficace meccanismo di caching, con la possibilità di invalidare regioni del cubo multidimensionale che andranno rigenerate dal RDBMS alla successiva query. Questa soluzione, causa la difficile implementazione tecnica, non è di fatto utilizzata in parecchie applicazioni Mondrian.

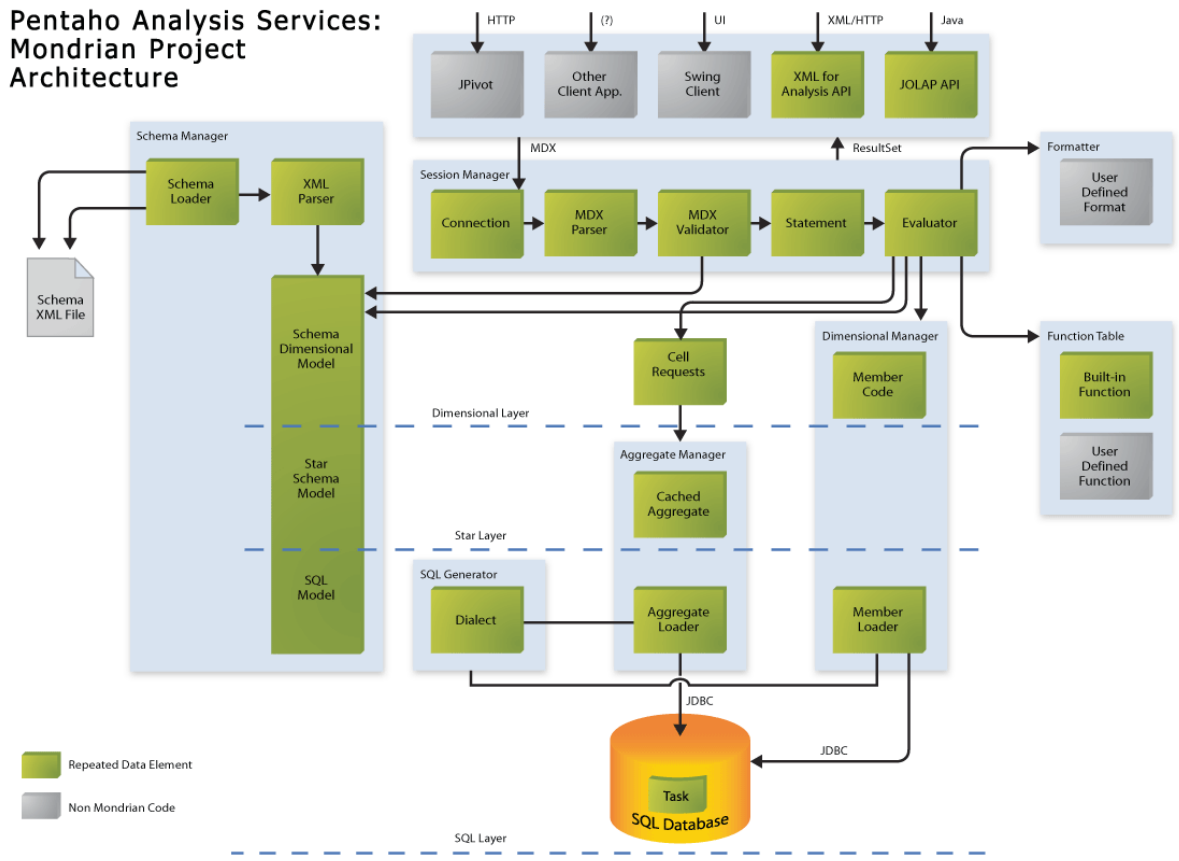


Illustrazione 12: Architettura di Mondrian

Il modo in cui Mondrian si interfaccia con l'RDBMS per caricare il cubo multidimensionale viene dapprima definito in un file XML di configurazione creato a mano oppure usando strumenti di sviluppo

quali Mondrian Schema Workbench. Recentemente è disponibile Pentaho Cube Designer, ancora privo di funzionalità importanti ma comunque utile per tracciare la base dello schema Mondrian.

L'interprete delle query per l'interrogazione del cubo multidimensionale è basato sul linguaggio MDX, pur consentendo l'utilizzo di XML for Analysis[6] e JOLAP[7] che vengono tradotte durante l'esecuzione nel linguaggio MDX. MDX, acronimo di Multidimensional Expressions, è stato introdotto dalla Microsoft in Microsoft SQL Server per OLAP nel 1998. Recentemente MDX è diventato parte di XML for Analysis ed è stato proposto da Microsoft come standard di riferimento. Mondrian, mancando la definizione completa del linguaggio MDX ai suoi albori, ha introdotto un dialetto che differisce in piccola parte dalla recente sintassi ufficiale.

6 XML for Analysis è uno standard che consente alle applicazioni di interrogare sorgenti dati OLAP. Usa MDX come linguaggio per le query e SOAP come protocollo di comunicazione.

7 JOLAP (Java Online Analytical Processing) è una interfaccia di programmazione Java che consente le operazioni di gestione e interrogazione di applicazioni OLAP

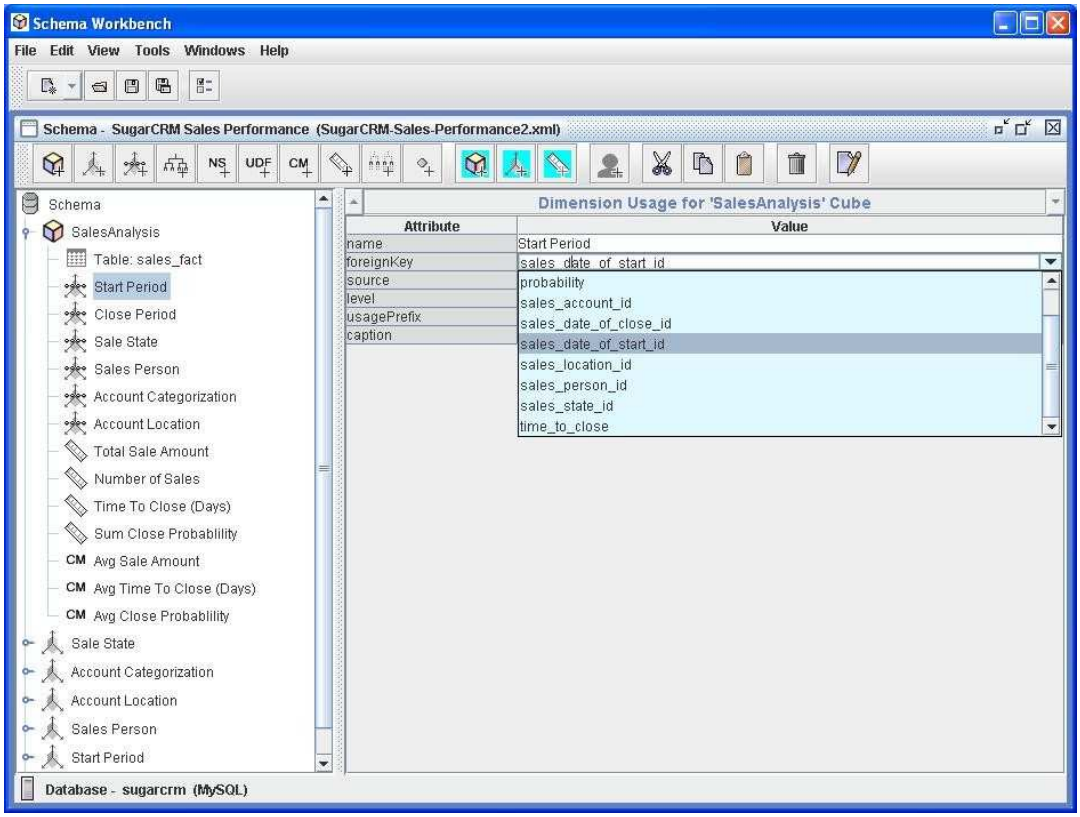


Illustrazione 13: Mondrian Schema Workbench

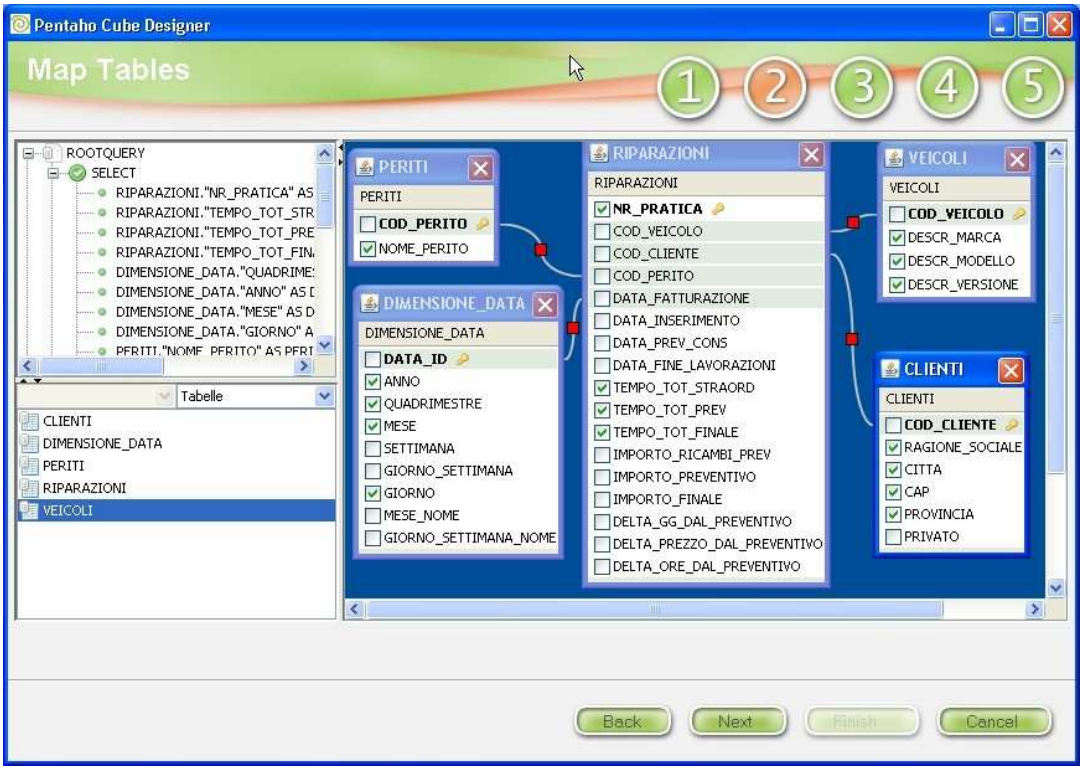


Illustrazione 14: Pentaho Cube Designer

Se Mondrian si occupa del reperimento dei dati, la parte di visualizzazione è demandata all'altro componente JPivot, che costituisce un binomio molto diffuso nelle applicazioni di analisi OLAP.

						Measures		
Region	My Image	P	P0	P1	P2	Measures[0]	Measures[1]	Measures[2]
−All Region[0]		0	V00			856.44	820.95	983.55
+Region[0]		0		V10		1,095.05	1,087.79	958.23
−Region[1]		1		V11		1,052.05	1,107.59	980.32
+City[0]	✔	0			V20	1,088.31	884.90	1,085.73
+City[1]	✔	1			V21	1,336.43	943.27	1,117.62
+City[2]	✔	2			V22	969.34	1,086.70	1,094.71
+City[3]	✔	3			V23	964.30	900.58	953.65
+City[4]	✔	4			V24	915.59	1,032.98	876.19
+City[5]	✔	5			V25	1,023.12	1,242.39	988.85
+City[6]	✔	6			V26	949.16	941.62	1,077.48
+City[7]	✔	7			V27	1,055.92	1,095.56	1,078.26
+Region[2]		2		V12		1,027.36	1,084.88	1,009.71
+Region[3]		3		V13		972.10	932.27	836.34
+Region[4]		4		V14		919.44	926.47	873.20

Illustrazione 15: Una immagine di JPivot

JPivot è una libreria JSP per interagire con cubi OLAP che consente di eseguire tipiche operazioni di navigazione quali:

- slice and dice
- drill down e roll up
- rotation

Viene impiegato un comune Application Server Java J2EE come ambiente di esecuzione e Mondrian come Server OLAP, pur supportando sorgenti dati aderenti allo standard XML for Analysis.

Alcune funzioni di layourt come la colorazione delle celle a seconda del loro valore vengono definite nella query MDX sorgente,

producendo tabelle con un buon impatto visivo.

Possono essere configurate dall'interfaccia web di JPivot la selezione delle dimensioni e dei fatti da visualizzare, come pure la query MDX.

Completano le funzionalità di JPivot la visualizzazione di grafici di epilogo, la stampa su file pdf e, seppure migliorabile, su Excel.

3.1.4 Grafici e dashboards (KPI)

Per la generazione di grafici Pentaho usa la libreria JFreeChart che prevede i classici formati a torta, barre, linee e area.

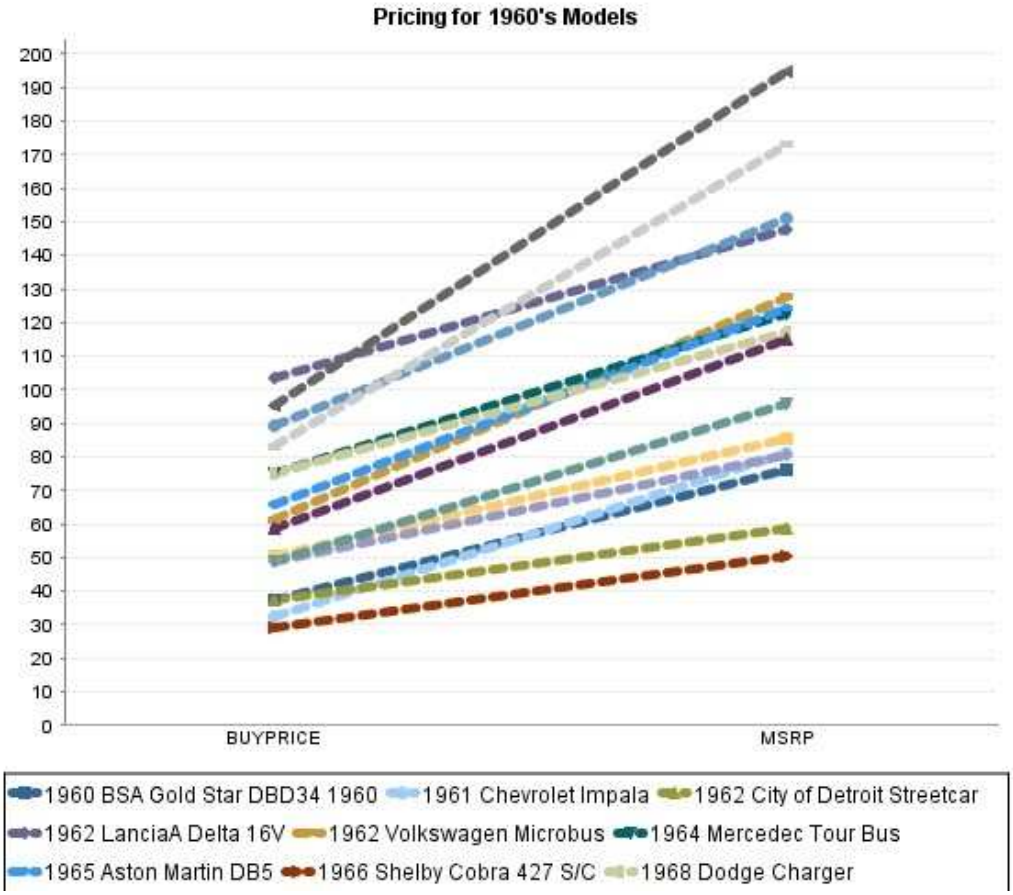


Illustrazione 16: Esempio di grafico con JFreeChart

Dalla versione 1.5 di PentahoBI è possibile utilizzare la libreria alternativa XML/SWF Charts, basata sul formato Adobe Flash e che

consente l'utilizzo di una più ampia gamma di grafici con eleganti effetti dinamici.

Manca in PentahoBI la possibilità di creare indicatori a lancetta in maniera agevole come pure le dashboards sono prerogativa di programmatori esperti. Chi volesse realizzare una dashboard anche non interattiva è richiesta una notevole dimestichezza con JSP,XML, HTML, Action Sequence e JavaScript dovendo creare una pagina JSP da zero.

Per rimediare alla mancanza è in cantiere il progetto Pentaho Dashboard Designer, con cui sarà possibile generare cruscotti direttamente da applicativi web.

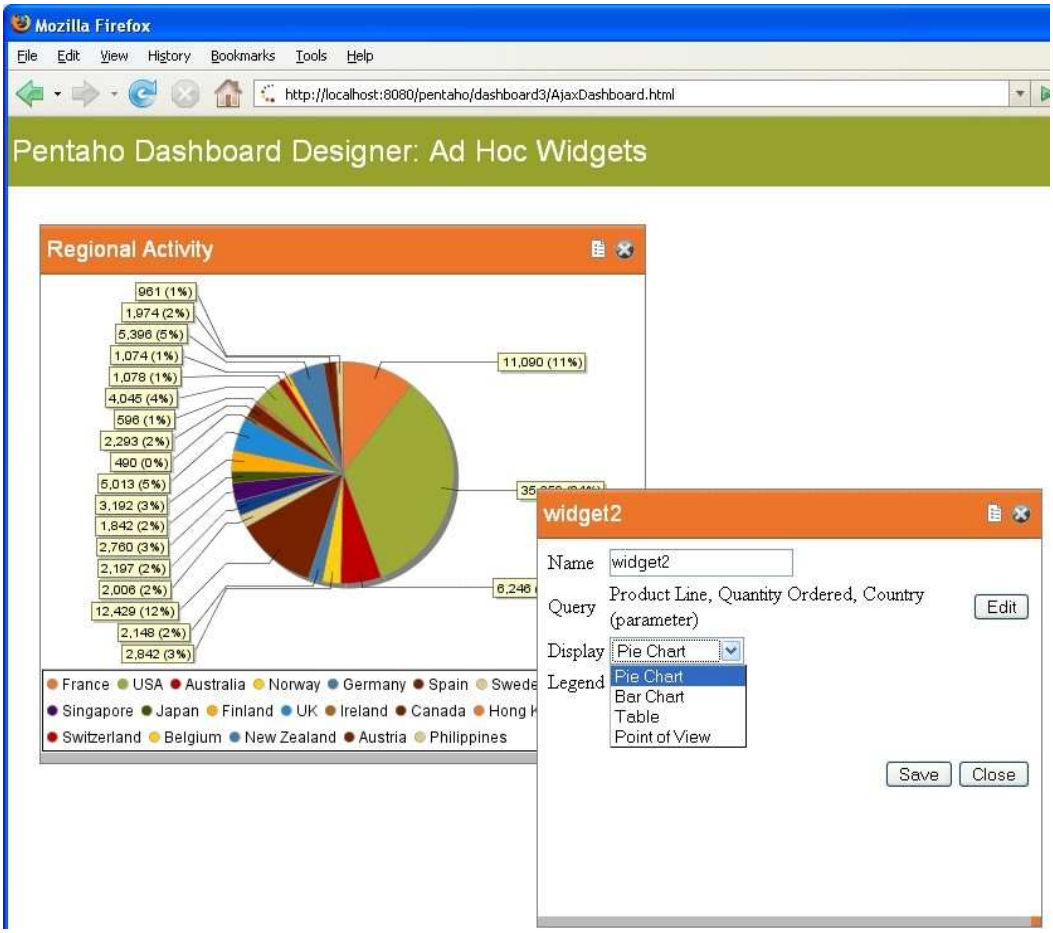


Illustrazione 17: Anteprima del progetto Pentaho Dashboard Designer

Pentaho Dashboard Designer al momento non è ancora rilasciato,

per cui chi volesse realizzare cruscotti senza troppe complicazioni tecniche dovrà orientarsi a uno strumento esterno oppure provare ad utilizzare la modalità a portale, scelta non del tutto consigliata per lo scarso supporto e per il fatto che non si tratta dell'interfaccia preferita di Pentaho BI.

3.1.5 ETL

L'ETL (Extraction Transformation Loading) è un componente essenziale nella Business Intelligence, come si è detto nel paragrafo 2. Pentaho propone per il processo di ETL il suo Pentaho Data Integration, maggiormente conosciuto con il nome di Kettle.

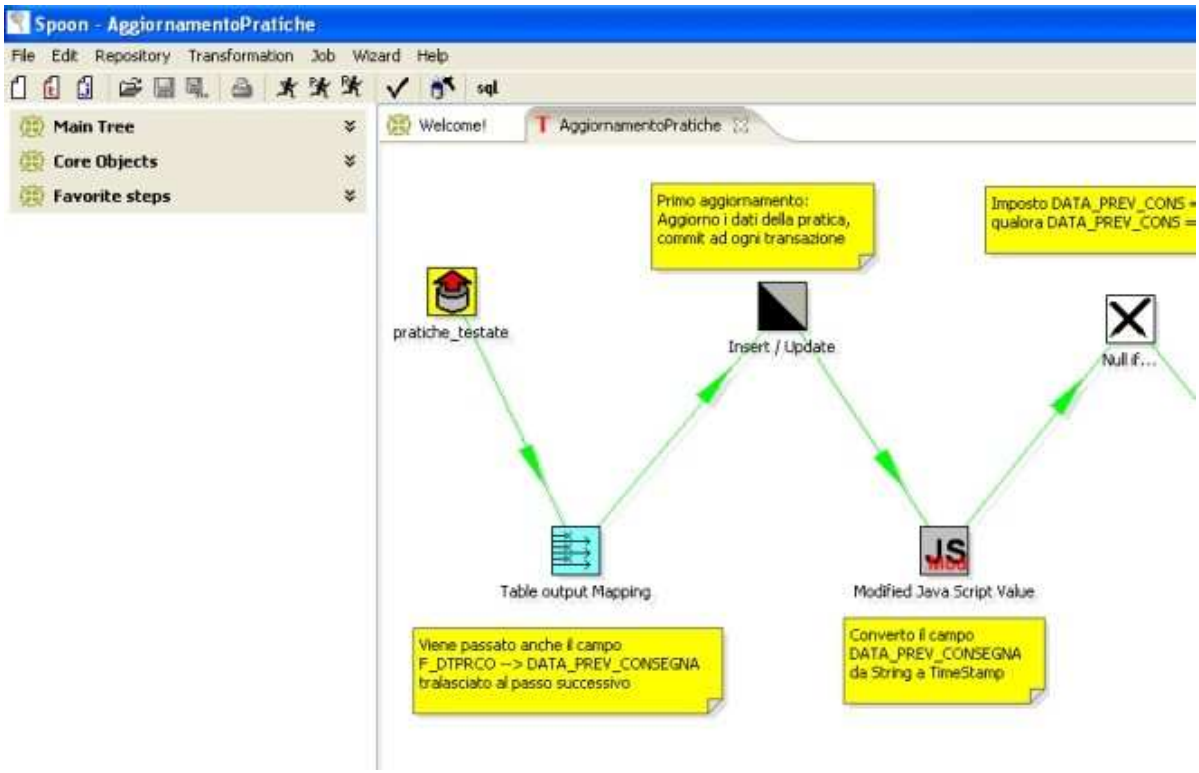


Illustrazione 18: Una schermata di Spoon, componente di Kettle

Una procedura ETL in kettle è in essenza un file XML prodotto dall'editor grafico Spoon (vedi Illustrazione 18) e mandato in esecuzione con Kitchen. Gli utenti possono inoltre demandare

l'avvio di procedure ETL ad altri computer nella rete usando il web service Carte.

Con Spoon definire una tipica trasformazione ETL è molto intuitivo e si compie in semplici passaggi come:

- definire le connessioni JDBC ai database
- trascinare i componenti sull'area del progetto, collegandoli in senso sequenziale di esecuzione
- testare l'applicativo realizzato
- salvare il tutto e mandare in esecuzione il progetto con Kettle

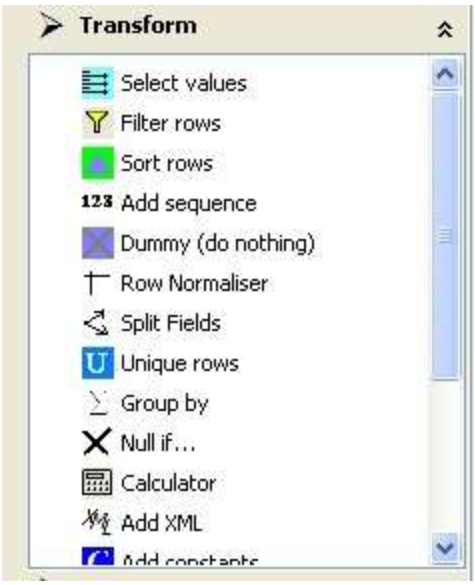
Le funzioni specifiche del processo ETL sono demandate a componenti specializzati, trascinati sull'area di lavoro e collegati in ordine di esecuzione. L'utente ha a disposizione del menù a tendina “Core Objects” un'ampia scelta di applicativi suddivisi in categorie tra cui le principali sono:

- **Input**

Sono componenti per definire flussi di dati in entrata, come il risultato di query su database relazionale ma anche file di testo, Excel o XML. Nel processo di ETL questi componenti si possono identificare nella fase di estrazione.

- **Transform**

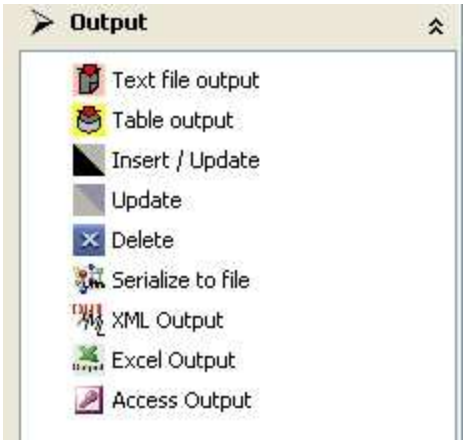
Comprende gran parte delle funzionalità di trasformazione di Kettle. Oltre a quelli della scheda Transform ci sono altri componenti per la manipolazione del flusso di dati come la modifica dei valori, il join di data set, il loockup dei campi, eccetera.



- **Output**

Permette di aggiornare una base dati di destinazione che può

essere un database relazionale oppure un file in formato testo, XML o Excel. Rappresentano le procedure di loading del processo ETL.



Oltre a questi componenti fondamentali Kettle mette a disposizione altre funzionalità per il passaggio di dati via socket, scripting con linguaggio JavaScript, gestione avanzata delle connessioni con partizione dei data set, connessioni multiple, clustering, logging dettagliato sia su file che su RDBMS.

Particolarmente interessante e completa è la funzionalità di debugging dei progetti. Una caratteristica che si potrebbe migliorare è il componente per lo scripting, ancora poco documentato e senza strumenti che agevolino la stesura del codice.

Le trasformazioni in Kettle hanno un flusso molto semplice, cioè prelevano dati da una o più sorgenti, eseguono delle elaborazioni e aggiornano una singola tabella di destinazione. Qualora si debbano aggiornare più tabelle di database si crea una trasformazione per ogni tabella e tramite un progetto job si accorpano le singole trasformazioni per darne un senso sequenziale di esecuzione. Il job viene realizzato in maniera simile a un progetto trasformazione, solo che i componenti in questo caso hanno funzioni più generali, come ad esempio lo scheduling.

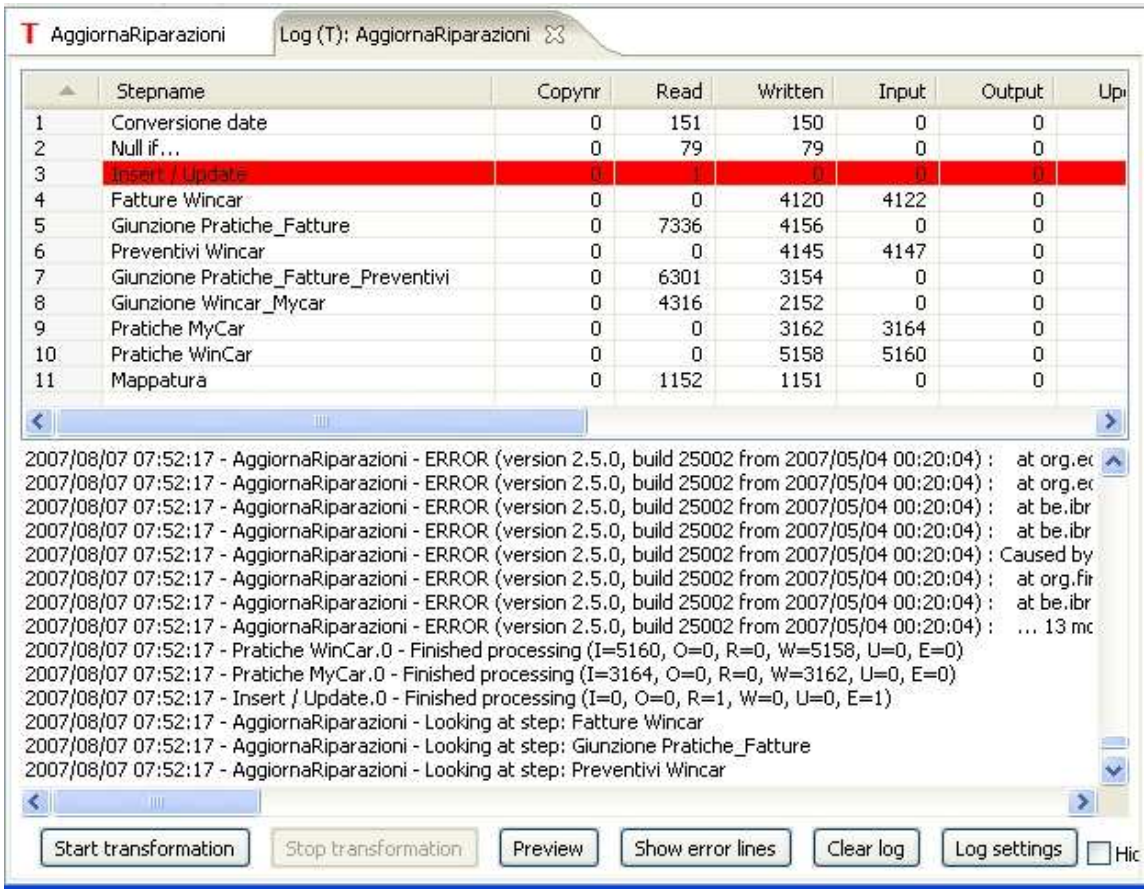


Illustrazione 19: Test della trasformazione in Kettle

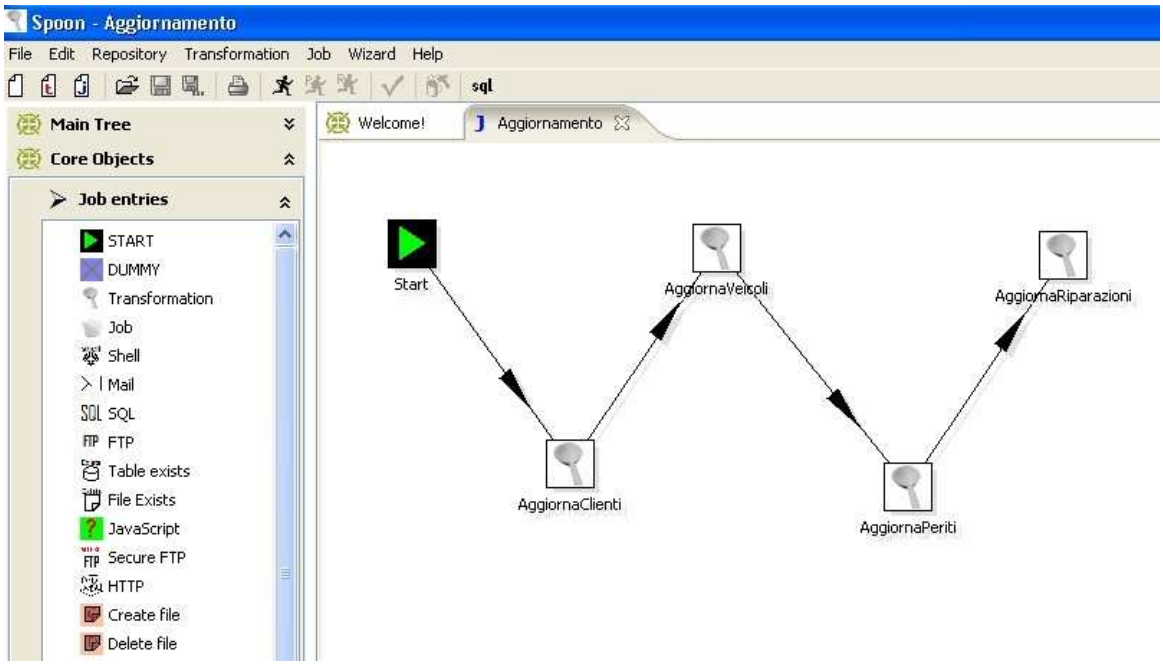


Illustrazione 20: Definizione di un progetto Job

3.1.6 Funzionalità aggiuntive

Completano la piattaforma le funzionalità GIS basate sulla grafica SVG (Scalable Vector Graphics) o in alternativa sull'integrazione con Google Maps.

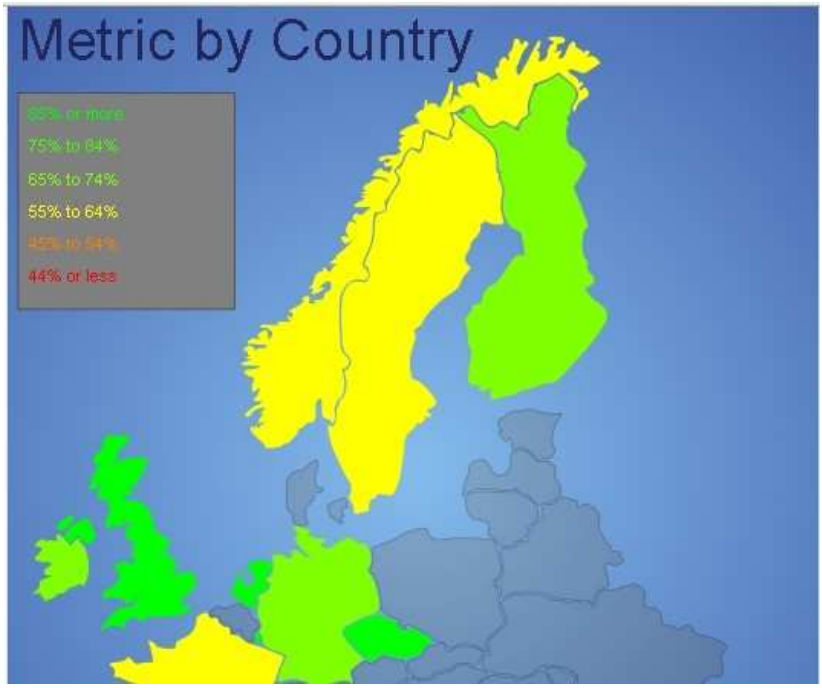


Illustrazione 21: Esempio di applicazione SVG

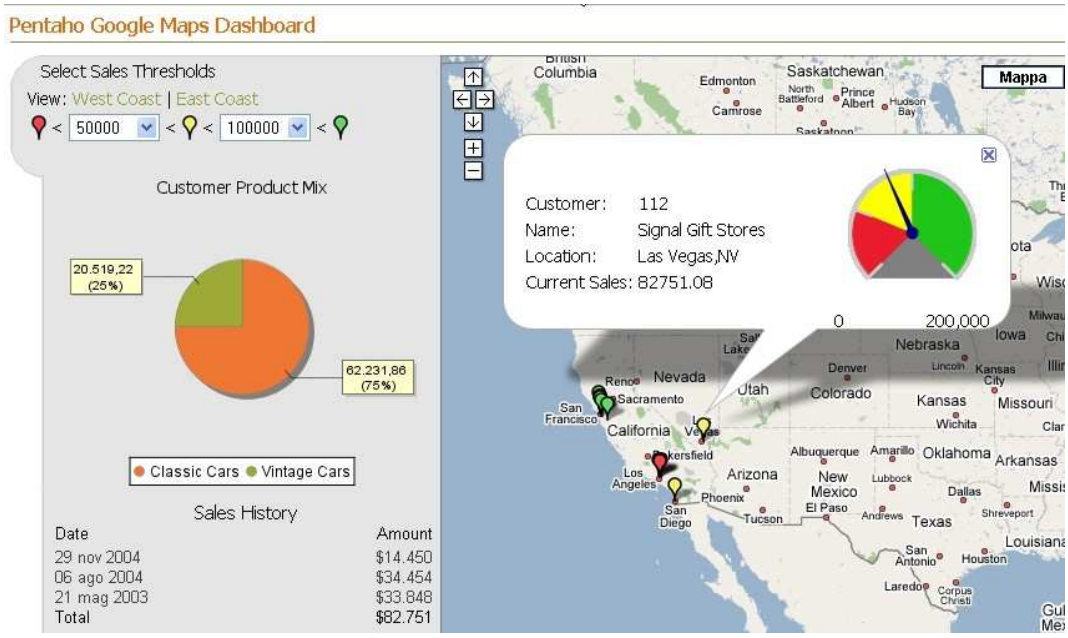


Illustrazione 22: Integrazione con Google Maps

Altre funzionalità come la schedulazione dell'esecuzione di applicativi, il motore di Workflow e il Data Mining con Weka, seppure pubblicizzate nel sito web di Pentaho sono in via di implementazione.

Manca nella versione 1.2 una gestione efficace dei diritti di accesso alle Action Sequence, come pure il login utente alla piattaforma. Se per il login si può ovviare installando nella rete aziendale un application proxy con tali funzionalità, per i diritti di accesso alle Action Sequence viene chiesto al programmatore uno sforzo tecnico notevole, situazione che migliora leggermente in Pentaho versione 1.5 ma ancora lontana dal definirsi una soluzione effettivamente completa.

3.1.6.1 Pentaho BI versione Professional

Pentaho BI è distribuito anche in una versione a pagamento. Le funzionalità aggiuntive in questo caso sono:

- Supporto tecnico
- Repository su database RBDMS, oltre che su directory di sistema
- Sottoscrizione alle Action Sequence: gli utenti possono schedulare le Action Sequence così da eseguirle a intervalli di tempo prestabiliti
- Possibilità di testare le Action Sequence in un ambiente di prova
- Report sulle performance di Pentaho BI e sull'utilizzo da parte degli utenti, particolarmente utili agli amministratori del sistema
- Vincoli sulle sottoscrizioni delle Action Sequence, in modo ad

esempio da proibire l'esecuzione di report in alcuni formati come Excel

- Single Sign-On con l'autenticazione a livello di Sistema Operativo usando LDAP, Kerberos o Active Directory
- Cancellazione dei contenuti obsoleti nelle Repository
- Clustering di certi tipi di Action Sequence come i Report

3.2 Piattaforma SpagoBI

Spago BI è la soluzione italiana per la Business Intelligence realizzata dalla Engineering Ingegneria Informatica in versione completamente “open” senza distribuzioni a pagamento.

A differenza di Pentaho BI dove i moduli software tendono ad essere fusi in un'unica soluzione, Spago BI si propone come una piattaforma di integrazione. Analisi OLAP, reporting, grafici e gli altri strumenti di Business Intelligence rimangono componenti esterni ma collegati alla piattaforma tramite driver software. Lo sviluppatore che volesse aggiungere funzionalità di un nuovo strumento software può creare il proprio driver che interagisca con le rispettive librerie.

Attualmente SpagoBI supporta tutte le funzionalità di Business Intelligence più rilevanti quali:

- Analisi OLAP (usando Mondrian/JPivot)
- Report (JasperReport e BIRT)
- Grafici e Cruscotti (OpenLaszlo)
- GIS (CartoWeb e MapServer)
- Data Mining (Weka)

- ETL (viene integrato Talend Open Studio)

Completano la gamma delle funzionalità le query QbE[8] (Query by Example), la generazione di presentazioni Power Point e lo strumento Booklet per la condivisione e discussione di documenti in gruppo.

L'interfaccia è interamente via web, compatibile con i browser di ultima generazione.

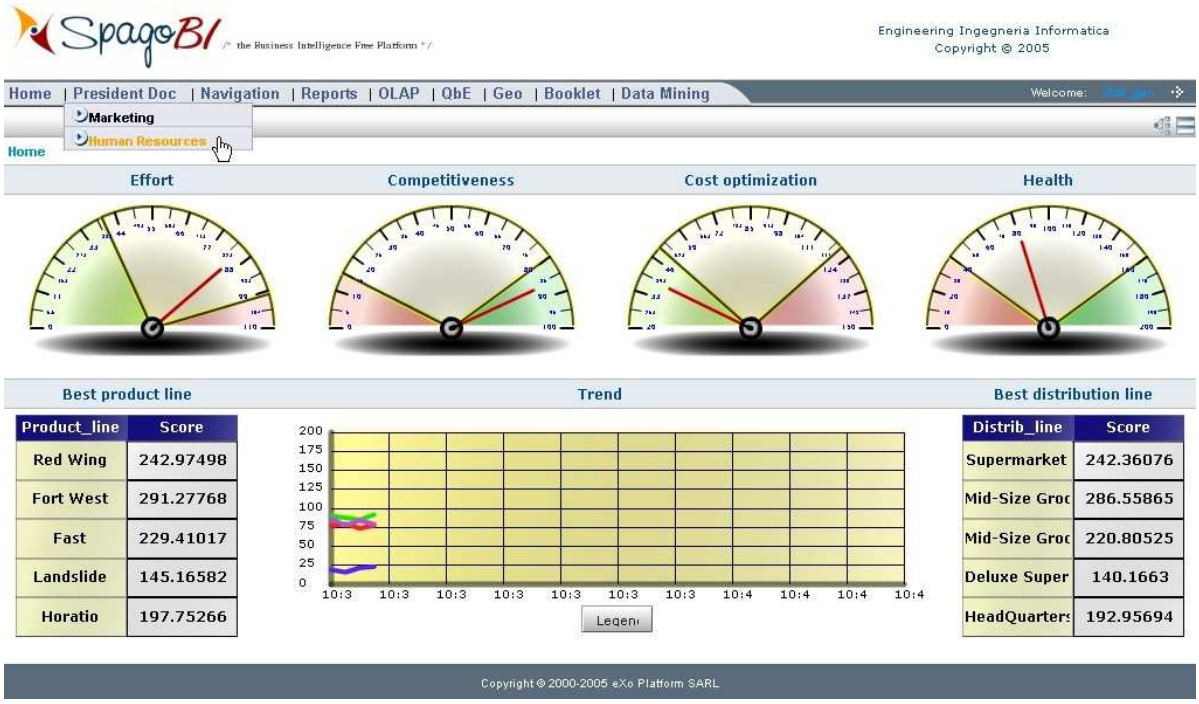


Illustrazione 23: Interfaccia utente SpagoBI

A livello implementativo la piattaforma fa uso del framework applicativo Spago, realizzato dalla Engineering Ingegneria Informatica per lo sviluppo di applicazioni mission-critical con tecnologia Java, ponendo particolare enfasi al pattern strutturale MVC (Model-View-Controller) e al paradigma SOA (Service Oriented Architecture). Spago BI si basa sull'utilizzo di Portlet JSR-168[9],

8 QbE consiste in un editor visuale per realizzare query senza usare SQL

9 E' il primo e più diffuso standard per l'interoperabilità fra portali e Portlet, dove per portale si intende un sito in cui l'utente con un singolo accesso ha a disposizione tutte

utilizzando l'ambiente di esecuzione EXO Portal e permettendo di scegliere dal programma di installazione tra i server JBoss, JonAS, IBM WebSphere e Apache Tomcat. Comunque l'adesione alle specifiche JSR-168 consente di installare SpagoBI teoricamente su qualsiasi portal container compatibile con tale standard.

SpagoBI rappresenta un concreto esempio di riutilizzo di soluzioni Open Source già esistenti per creare un prodotto specifico e complesso di elevato valore aggiunto.

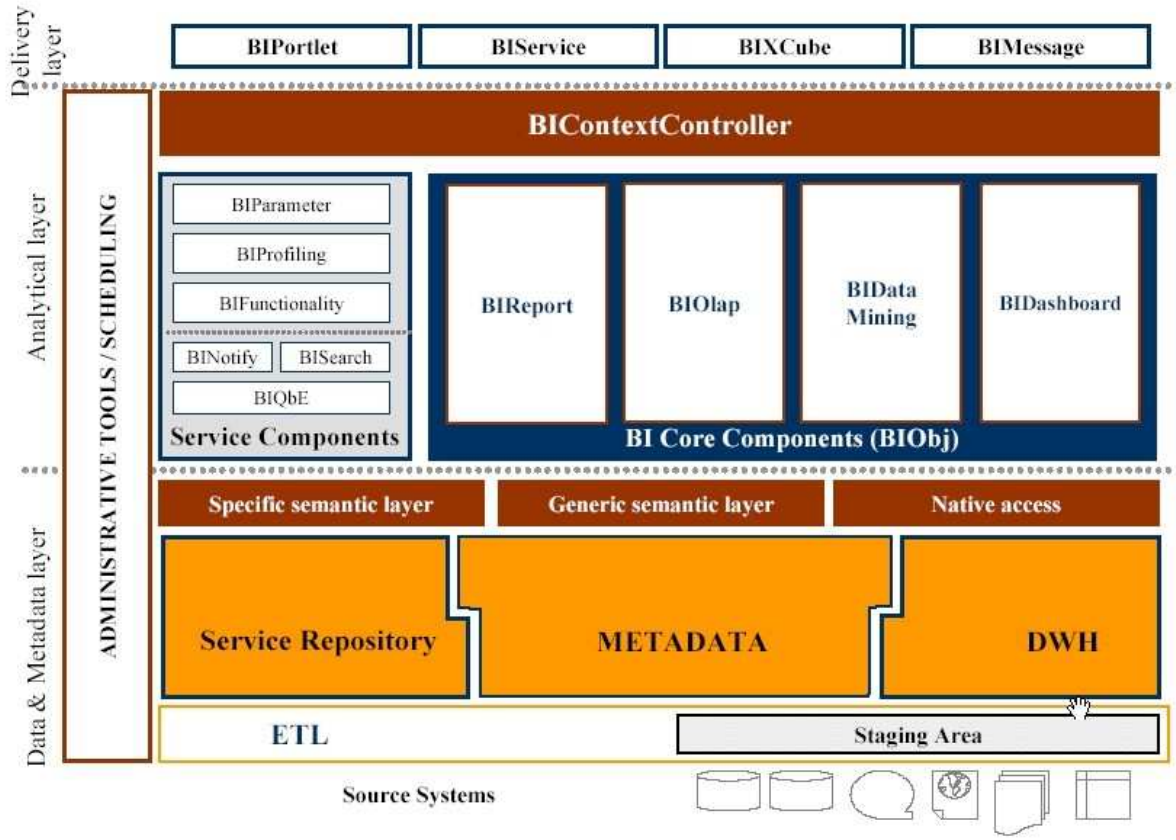


Illustrazione 24: Architettura di SpagoBI

3.2.1 Sviluppo e amministrazione

Grazie alla natura a portale della piattaforma, lo sviluppatore degli applicativi come pure l'amministratore di sistema possono accedere

le risorse aziendali di cui necessita e per Portlet si intende i vari componenti che vanno a costituire il contenuto del portale.

direttamente via web ai rispettivi strumenti di lavoro.

La registrazione di nuovi documenti analitici avviene con procedure guidate come quelle visibili nell'Illustrazione 25 dove l'utente il più delle volte fornisce il file progetto realizzato con i rispettivi strumenti di sviluppo come iReport o BIRT e completa le informazioni della videata per integrarli nella piattaforma.

In ambienti complessi e con molti utenti torna utile la gestione la procedura di approvazione dei documenti analitici, che possono essere lasciati in fase di test prima di pubblicarli agli utenti finali, il tutto nel rispetto dei diritti assegnati.

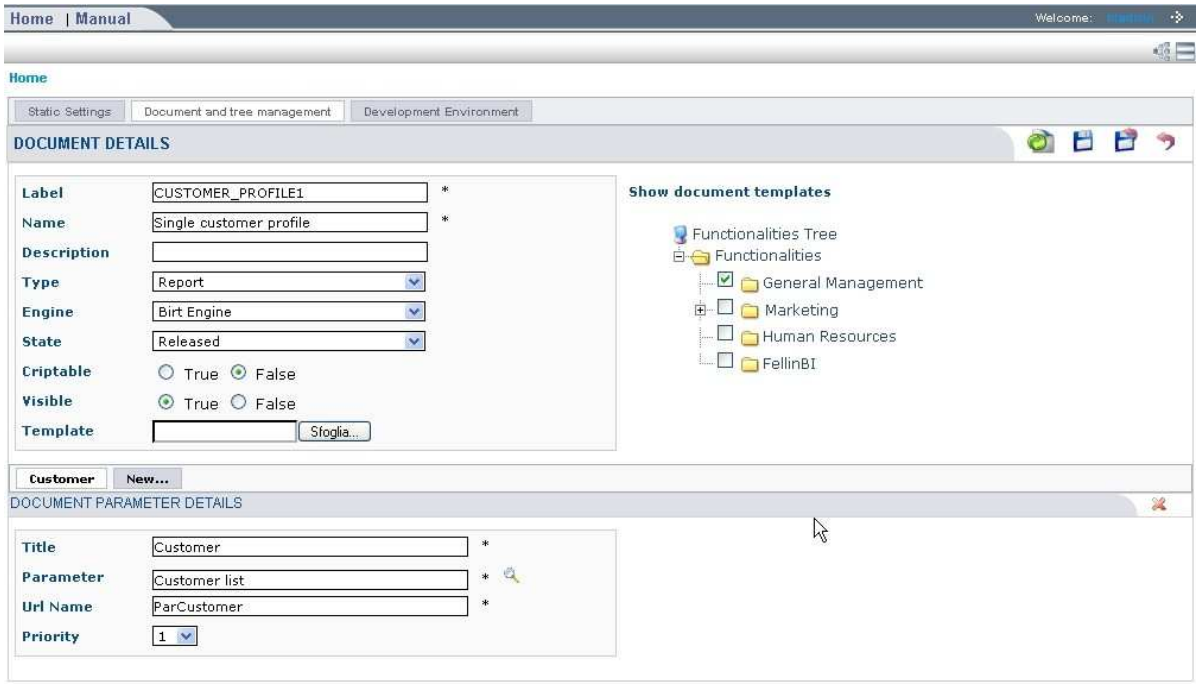


Illustrazione 25: SpagoBI: Importazione di un report creato con BIRT

Vengono gestite direttamente da web anche le pagine del portale e la navigazione per il singolo utente, come si può vedere dall'attività di posizionamento delle Portlet in una pagina di Illustrazione 26.

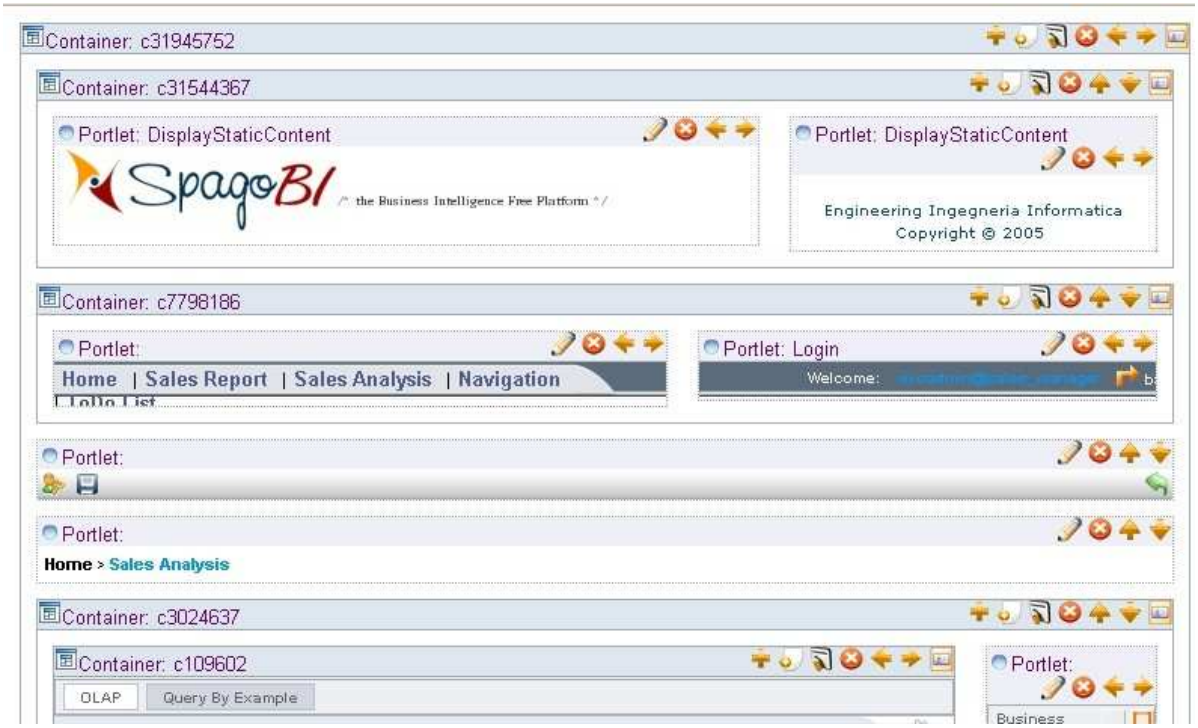


Illustrazione 26: Personalizzazione pagine del portale in SpagoBI

Spago BI grazie alla completa interfaccia web riduce notevolmente gli interventi al filesystem della piattaforma e i ai suoi file di configurazione, operazioni queste che avvengono solo in interventi specifici come l'aggiunta della connessione a un nuovo database.

Caratterizza lo sviluppo di documenti analitici in Spago BI la particolare struttura dei parametri. Un parametro può ricevere valori da uno specifico elenco di valori (List Of Values) generato da query SQL o script Java Script e può essere vincolato a particolari domini di valori (Predefined Values Constraints). L'elenco dei valori e i vincoli sono elementi indipendenti ma integrabili fra loro a formare nuovi parametri, consentendo un notevole riutilizzo dei singoli componenti.

3.2.2 Reportistica

SpagoBI utilizza come software per la reportistica i già citati JasperReport (vedi 3.1.2.1) e BIRT (vedi 3.1.2.2) oltre alla soluzione proprietaria WebI di Business Objects. Questi report, come nel caso di PentahoBI, vanno realizzati con i rispettivi strumenti esterni di design per poi registrare il file progetto nella piattaforma.

3.2.3 Analisi OLAP

Anche in questo caso l'analisi OLAP è principalmente demandata al binomio Mondrian/JPivot presentati nel paragrafo 3.1.3. Rispetto a PentahoBI la creazione di interrogazioni JPivot su schemi Mondrian è molto agevolata da una completa procedura assistita, che di fatto evita la scrittura a mano della query MDX. La definizione dello schema Mondrian rimane compito di strumenti appositi come Pentaho Cube Designer o Mondrian Schema Workbench. In alternativa a Mondrian si possono usare server con supporto allo standard XMLA (XML for Analysis), come Microsoft Analysis Server.

3.2.4 Grafici e dashboards (KPI)

L'approccio ai grafici di SpagoBI è una soluzione alternativa alle classiche librerie quali JFreeChart o XML/SWF Charts. Si utilizza infatti OpenLaszlo, una piattaforma Open Source per la creazione di applicazioni web usando file XML e codice JavaScript, che vengono compilati trasparentemente in Adobe Flash.

Per realizzare un grafico in Spago BI è necessario:

- preparare un nuovo grafico in formato Adobe Flash, o riutilizzare uno dei molti già predisposti

- creare un file XML che specifichi l'origine dei dati, la scala e le informazioni supplementari al grafico
- incorporare il file XML all'interno della piattaforma usando l'apposita procedura guidata

Come si vede nell' Illustrazione 23 l'uso della tecnologia Adobe Flash consente un effetto visivo gradevole. Lo sviluppatore può decidere di realizzare nuovi componenti oppure utilizzare quelli preesistenti che sono grafici a barre, a linee, a lancetta, a griglia e a torta.

SpagoBI sta progettando una gestione alternativa a OpenLaszlo per la generazione di cruscotti, meglio integrabile nella piattaforma.

3.2.5 ETL

Per la fase di ETL SpagoBI integra la soluzione Talend Open Studio, pur consentendo qualunque altro meccanismo di ETL che di fatto è una operazione indipendente dalla piattaforma. Il software è completamente Open Source, disponibile in binari precompilati per Windows o Linux. Al primo avvio si nota che il programma sia basato su Eclipse, come nel caso di Kettle.

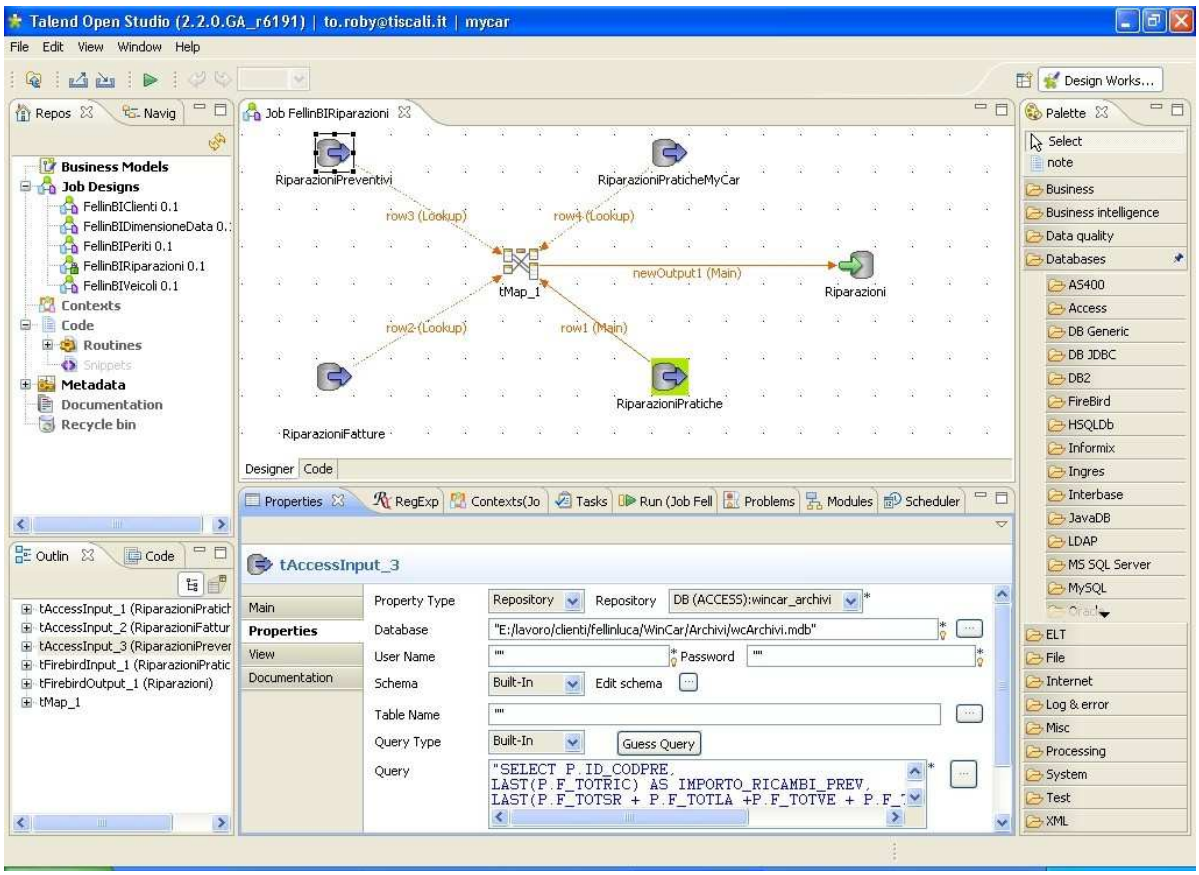


Illustrazione 27: Schermata di Talend Open Studio

Per realizzare un progetto con Open Studio va creato un job, in cui è possibile trascinare i componenti e darne una sequenza di esecuzione. A differenza di Kettle, dove il progetto viene salvato in file XML e quindi eseguito da un interprete, Open Studio provvede a generare direttamente il codice della procedura in Java o in Perl. Per eseguire il progetto si può utilizzare il comodo strumento integrato di debugging oppure compilare l'applicativo finale. Con questa ultima opzione verrà generata una cartella con il programma compilato che può essere mandato in esecuzione da Windows o Unix, previa l'installazione dell'ambiente Java o, a seconda dei casi, Perl.

Molto marcato è l'approccio basato sui meta dati, ossia lo schema delle sorgenti dati in entrata e uscita. Anche se così facendo si

vincola lo sviluppatore a definire lo schema delle base dati prima di usare i componenti di Open Studio, la manutenibilità del progetto a seguito dei cambiamenti viene molto agevolata perché gli schemi appartengono a una repository comune, riducendo le possibilità di incompatibilità fra flussi sorgente e destinazione.

Come ogni buon software di ETL i componenti a disposizione sono molti. Oltre ai classici strumenti di estrazione, mappatura e aggiornamento delle base dati sono previsti strumenti per il logging, connessione a server web o email, loop condizionale, filtraggio e aggregazione dei record e tanti altri. Passaggi complessi possono avvalersi del potente linguaggio di programmazione sottostante, che può essere Java o Perl a seconda di come viene inizializzato il progetto.

Se Kettle ha il vantaggio di essere più semplice da usare nei classici casi di aggiornamento di base dati, Open Studio è uno strumento che mostra tutta la sua flessibilità in progetti complessi, merito del linguaggio di programmazione Java o Perl e delle scelte progettuali che agevolano l'evoluzione di grossi progetti. Più che un semplice strumento per la creazione di applicazioni ETL Talend Open Studio ha tutte le caratteristiche di un vero e proprio ambiente integrato. Nel pannello “Repository” in alto a sinistra infatti sono disponibili una serie di schede per i documenti che vengono via via realizzati, suddivisi in:

- **Job Designs** Raccoglie le procedure ETL definite dall'utente
- **Metadata** Gestisce le connessioni alle fonti dati e i rispettivi schemi, che si tratti di RDBMS oppure di file fisici
- **Business Models** Come documentazione aggiuntiva l'utente può creare dei diagrammi direttamente da Open Studio, che vengono poi raccolti in questa scheda

- **Contexts** Le procedure di ETL possono essere personalizzate facendo uso di parametri. La repository context prevede la definizione di questi parametri all'interno di scenari di utilizzo, come ad esempio il debugging o l'installazione su ambienti differenti
- **Code** L'utente può creare delle routine personalizzate, in codice Java o Perl a seconda di come viene configurato il progetto. La repository Code raccoglie queste funzioni in librerie che possono venire richiamate dagli strumenti di Open Studio
- **Documentation** File esterni possono essere integrati nel progetto per documentare le scelte adottate
- **Recycle bin** I documenti cancellati nelle repository finiscono in un cestino, in attesa che l'utente ne svuoti il contenuto di sua spontanea volontà

Come sorgenti dati possono essere utilizzati database relazionali oppure file in vari formati (tra cui testo, XML, XLS, ZIP, ecc.) o anche protocolli web come POP3, FTP, HTTP e così via. I database relazionali supportati sono praticamente tutti quelli noti, merito di componenti personalizzati oltre ai classici componenti per connessioni JDBC e ODBC.

Completano le opzioni di esecuzione le ricche funzioni di scheduling. Verrà in tal modo generato un file batch che imposta lo scheduler UNIX cron, di cui è disponibile anche il porting su Windows.

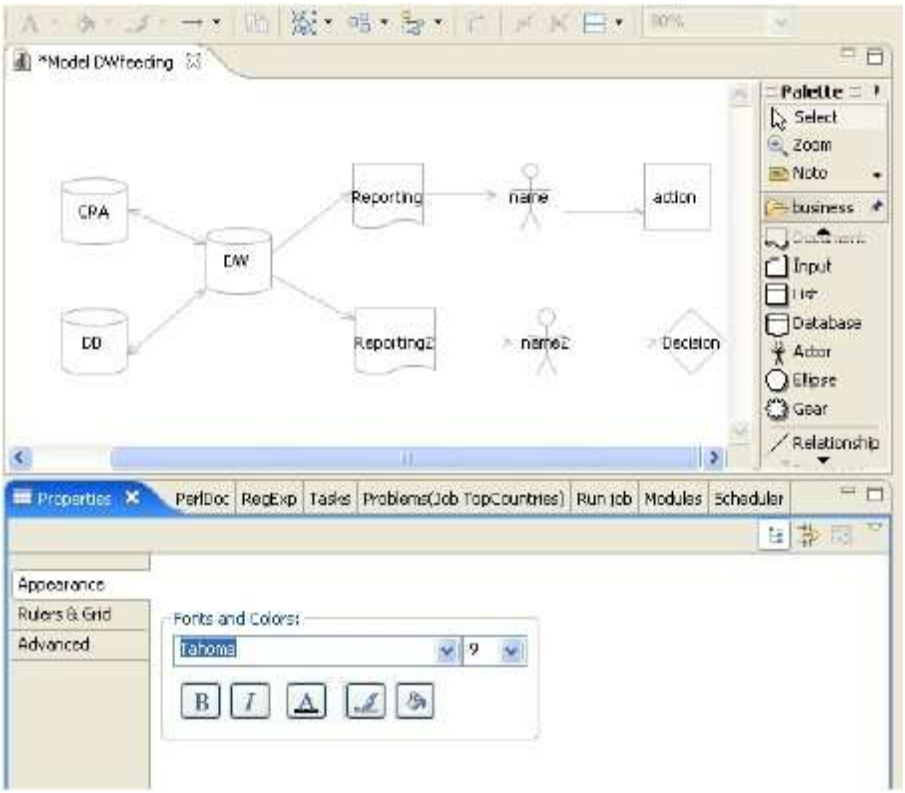


Illustrazione 28: Business Model in Talend Open Studio

3.2.6 Funzionalità aggiuntive

Completano la soluzione Spago BI altri interessanti strumenti di Business Intelligence quali:

- GIS per la visualizzazione di dati in cartine geografiche
- Esportazione dati in presentazioni Power Point
- Data Mining
- Discussione di gruppo dei documenti analitici
- Query by Example
- ToDo list

Per agevolare il monitoraggio dei documenti analitici in SpagoBI dalla versione 1.9.3 sono state aggiunte funzionalità di auditing, non discusse in questa sede che tratta la versione 1.9.2.

Illustrazione 29: Funzionalità GIS in SpagoBI

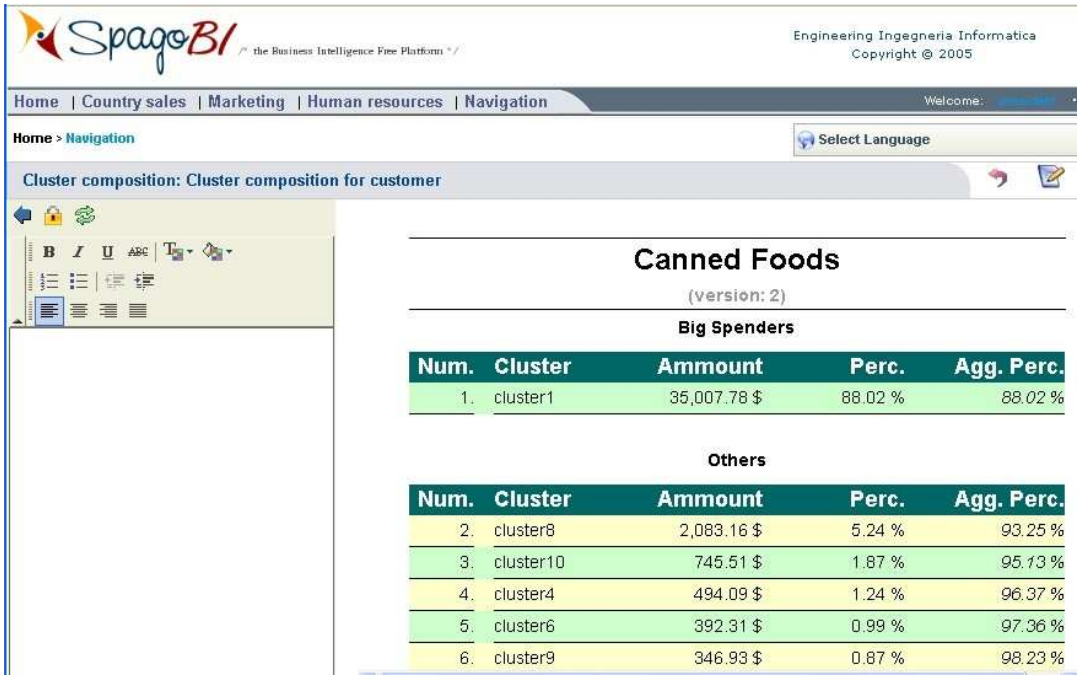


Illustrazione 30: Data Mining e funzione annotazioni in SpagoBI

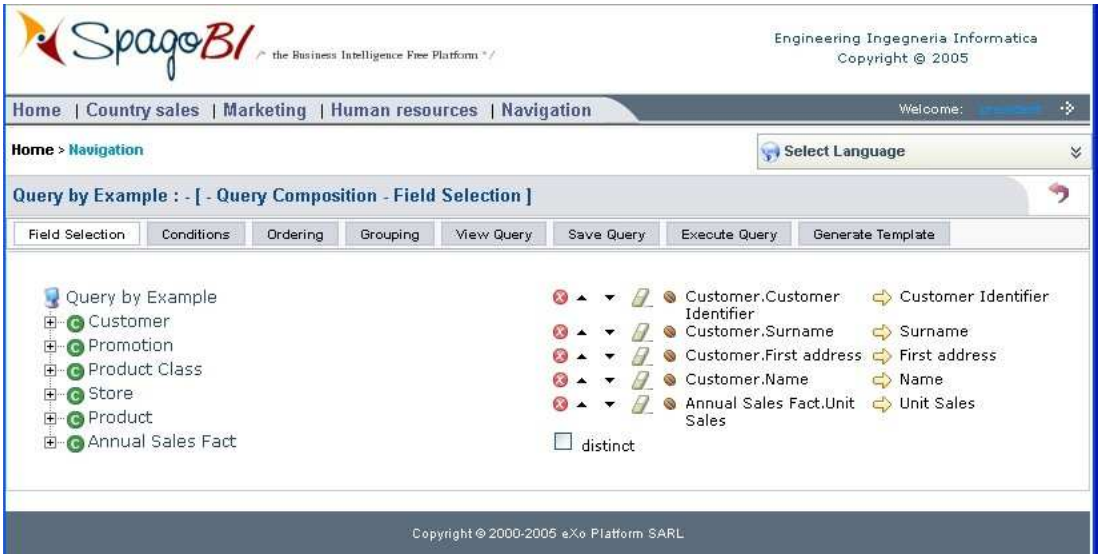


Illustrazione 31: Strumento Query by Example

4 Il progetto

Se le grandi imprese possono rivolgersi per il proprio ambiente di Business Intelligence ai prodotti commerciali più rinomati, le piccole e medie imprese hanno bisogno di soluzioni alternative, più adatte alla disponibilità di capitali e risorse limitate. Il basso TCO (Total Cost of Ownership) e la filosofia di sviluppo aperta rendono i software Open Source particolarmente interessanti in questo senso.

Il progetto nasce per provare due strumenti Open Source di Business Intelligence in un tipico scenario reale, annotandone i pregi e i difetti. È stata scelta come ambiente di sperimentazione la Carrozzeria Fellin Luca, una ditta trentina con caratteristiche organizzative e tecnologiche particolarmente rappresentative della PMI italiana. L'azienda si è dimostrata particolarmente adatta anche per la presenza di due software operazionali, WinCar di SystemData e MyCar, realizzato dal sottoscritto in collaborazione con la Carrozzeria Fellin. Questo consente di implementare un progetto di Business Intelligence considerando che:

- Si devono estrarre ed elaborare dati provenienti da fonti eterogenee, l'una composta da tabelle di Microsoft Access (WinCar) e l'altra da un database Firebird (MyCar)
- I risultati delle analisi devono essere principalmente grafici e facili da comprendere. Inoltre non essendo presente del personale tecnico informatico coloro che accedono alle informazioni saranno direttamente i responsabili e i manager

4.1 Carrozzeria Fellin Luca

La Carrozzeria Fellin (http://www.fellincar.it) opera nel mercato trentino dal 1983 ed è diventata con i suoi 15 dipendenti azienda leader tra le carrozzerie auto della provincia di Trento. I servizi offerti comprendono la completa assistenza per quanto riguarda la riparazione auto fornendo ricambi originali, auto sostitutiva e intermediazione con le assicurazioni.

Negli ultimi anni la crescita dell'azienda è stata notevole e il titolare Luca Fellin ha investito su un rinnovamento organizzativo, ponendo particolare interesse alle soluzioni informatiche per migliorare il flusso delle informazioni sia all'interno dell'azienda che nel contatto con i clienti.

In particolare l'ottimizzazione dei tempi di lavorazione e la gestione dei ricambi auto si dimostra una strategia molto importante, incoraggiando l'adozione di strumenti informatici per la rilevazione e l'analisi di dati.

4.2 Obiettivi

L'obbiettivo aziendale della Carrozzeria Fellin è elaborare i dati già

raccolti dai gestionali e ricavarne informazioni di sintesi, soprattutto per quanto riguarda l'analisi dei tempi di lavorazione e costi.

Non c'è a tal proposito un algoritmo o una elaborazione specifica, si preferisce costruire per il cliente un cubo con i dati delle riparazioni e demandare in futuro la creazione di applicativi specifici, quando le esigenze della carrozzeria saranno più nettamente delineate. Questo rientra nell'ottica di una progettazione iterativa ed evolutiva, dove a cicli periodici si aggiungono nuove funzionalità estendendo quelle precedentemente realizzate.

Il cubo con i dati prenderà il nome di **cubo riparazioni** e particolarmente interessante per il titolare sono i fatti:

- differenza di prezzo fra preventivo e fattura emessa
- differenza di tempo impiegato fra preventivo ed effettivo
- differenza di giorni da preventivo a fine lavorazioni

Le esigenze strategiche dell'azienda traggono inoltre beneficio dall'elenco dei clienti che producono maggior fatturato, avendo la carrozzeria rapporti frequenti con clienti aziendali e non solo privati. A tal fine particolarmente adatta è la realizzazione di un report **fatturato per cliente**, con l'elenco dei clienti diviso tra aziendali e privati e ordinato per fatturato.

In report come questi risulta utile che l'utente possa applicare un filtro per data, cosa che risulta difficoltosa sia in PentahoBI che in SpagoBI per la mancanza di uno strumento semplice di inserimento della data, come ad esempio un calendario. Piuttosto che l'utente finale debba riportare la data a mano alcuni programmatori preferiscono aggiungere un calendario modificando i sorgenti della piattaforma. In assenza di tale soluzione l'utente è chiamato a scrivere date nel formato timestamp “aaaa-mm-gg 00:00:00” (a sta

per cifra dell'anno, m mese e g giorno), cosa che ha dei notevoli limiti e per tale motivo non viene considerato nel progetto.

Sarebbe poi interessante fare un'analisi cluster, dividendo i gruppi dei clienti profittevoli da quelli non profittevoli. L'attuale mancata integrazione di un software di Data Mining in Pentaho BI non permetterebbe però un confronto alla pari con SpagoBI.

L'attività di aggiornamento è un altro parametro importante. Un aggiornamento frequente può incidere sulle prestazioni del server soprattutto quando i dati sono voluminosi, mentre se si imposta un periodo di attesa elevato la base dati può risultare poco rappresentativa. Nelle applicazioni pratiche si utilizzano frequenze che vanno dall'ordine di ore per le applicazioni real time a giorni nei rimanenti casi. Per il caso della Carrozzeria Fellin è stato scelto un tempo di aggiornamento di 12 ore, in modo da avere la base dati alimentata due volte al giorno.

Il cliente ha poi ulteriori esigenze come indicatori di carico di lavoro, analisi delle forniture, analisi delle lavorazioni, eccetera. In un progetto reale queste alternative andrebbero analizzate tutte, almeno superficialmente, per evitare di trovarsi in seguito con una base inadatta e dover eseguire delle ristrutturazioni. Un'analisi completa degli obiettivi richiederebbe un livello di dettaglio certamente maggiore rispetto a quello riportato in questo paragrafo, ma per il fine del progetto, ossia di studiare la qualità di un software Open Source di Business Intelligence, l'analisi racchiude già sufficienti informazioni per iniziare. Se il cliente noterà interesse nella soluzione finale sarà possibile continuare lo sviluppo, con un'ulteriore analisi degli obiettivi nell'ottica della progettazione iterativa ed evolutiva.

Non c'è un procedimento unico per la progettazione di DataWarehouse, ma la definizione degli obiettivi andrebbe

accompagnata con gli schemi delle sorgenti dati esistenti.

4.3 Schema database operazionali

In azienda sono installati e funzionanti due database operazionali. Uno è per il software WinCar finalizzato alla creazione di preventivi e la fatturazione degli stessi. L'altro è MyCar, un gestionale che estende WinCar con funzionalità che coprono tutta la fase di lavorazione delle auto. La base dati di WinCar è su file Microsoft Access, mentre quella di MyCar è sul RDBMS Firebird SQL. Per entrambi è disponibile un driver JDBC che ne consente l'uso in Kettle durante le fasi di alimentazione, mentre per Talend Open Studio si può ricorrere a componenti specifici.

Lo schema della base dati di WinCar opportunamente ridotto con i campi di interesse è il seguente:

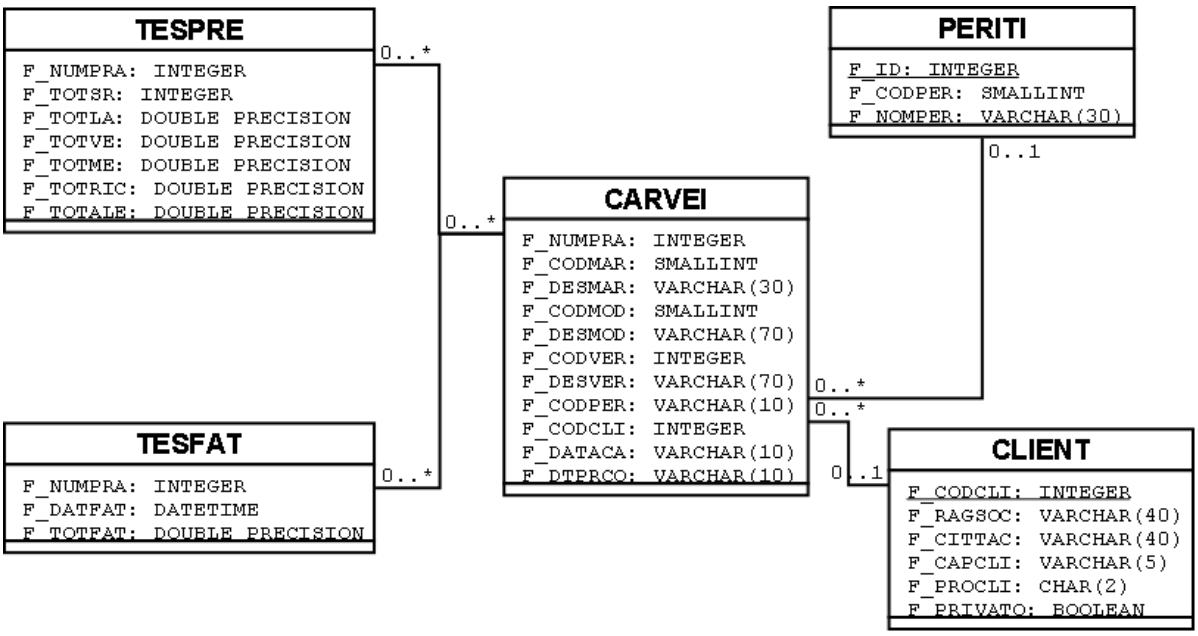


Illustrazione 32: Schema della base dati WinCar

Piuttosto che la notazione E/R si è preferito lo stile UML che risulta più chiaro e compatto. In questa notazione le chiavi primarie vengono sottolineate, mentre la molteplicità è rappresentata dai simboli 0..1 se sono concesse 0 o 1 occorrenze, 0..* se sono concesse 0 o illimitate occorrenze.

Si noti come le tabelle TESFAT, CARVEI e TESPRE siano sprovviste di chiave primaria, inoltre in CARVEI il campo F_CODPER è di tipo testo e viene usato per giungere la tabella dei periti che invece ha un campo di tipo numerico. Lo schema risente certamente delle modifiche e riadattamenti fatti nel tempo per consentire al programma WinCar di implementare nuove funzionalità.

L'anagrafica dei veicoli, pur interessante ai fini analitici, non è direttamente accessibile in WinCar per motivi strategici della software house produttrice, trattandosi di dati che hanno un notevole valore commerciale. L'unica soluzione percorribile è utilizzare le informazioni denormalizzate della tabella CARVEI per costruire un'anagrafica veicoli. In CARVEI infatti sono riportati sia i campi chiave che le descrizioni di marca, modello e versione della vettura.

Nel caso di MyCar si utilizza solo la tabelle PRATICHE_TESTATE_2 che ha il seguente schema opportunamente ridotto:

Illustrazione 33: Schema della base dati MyCar

4.4 Schema database analitico

La base dati analitica si riduce alla creazione del cubo multidimensionale Riparazioni e le dimensioni ad esso associate. Utilizzando l'RDBMS Firebird SQL verrà creato il database “FellinBI” adottando uno schema a stella per motivi di semplicità e prestazioni. Lo schema viene qui riportato usando la notazione UML e con la convenzione di sottolineare il nome dei campi con chiave primaria.

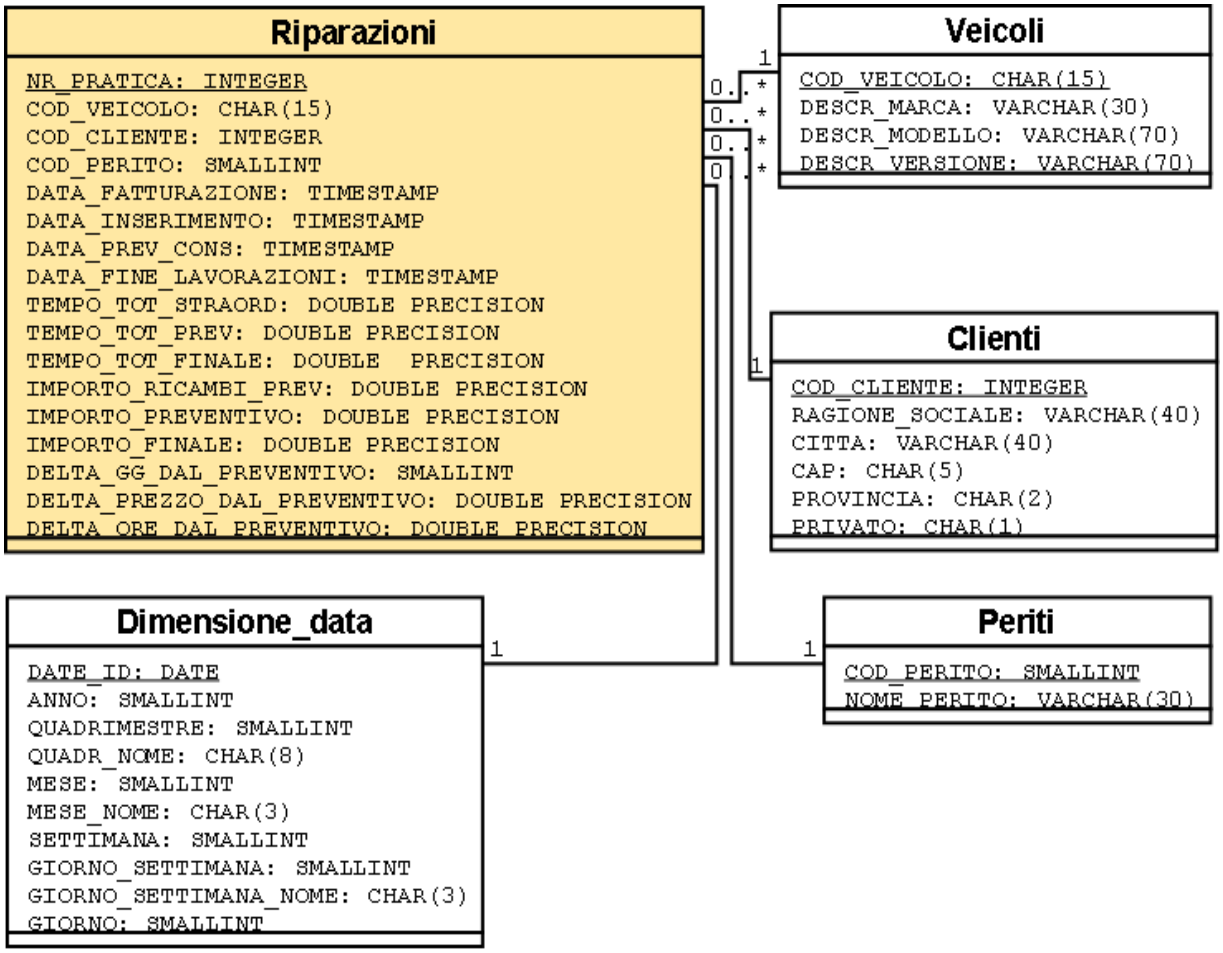


Illustrazione 34: Schema del database analitico

I valori dei campi sono prelevati dai database MyCar e WinCar, eccetto per i campi calcolati DELTA_GG_DAL_PREVENTIVO, DELTRA_PREZZO_DAL_PREVENTIVO e DELTA_ORE_DAL_PREVENTIVO della tabella RIPARAZIONI e per la tabella dimensione_data che è

generata manualmente con le informazioni delle gerarchie per ogni data.

Per le limitazioni del motore OLAP Mondrian le dimensioni non possono avere chiave primaria composita, cioè formata da gruppi di colonne. Questo interferisce nella dimensione Veicoli, dove è stata realizzata una unica colonna con chiave primaria di tipo stringa nata dalla fusione di F_CODMAR, F_CODMOD, F_CODVER della tabella CARVEI.

Come formato dati per i numeri in virgola mobile è stata usata la doppia precisione (Double Precision) per mantenere un elevato grado di accuratezza anche su calcoli complessi, dato che le dimensioni relativamente esigue del database tollerano abbondantemente il peso di memorizzazione della doppia precisione rispetto a colonne con precisione singola.

Particolare è stata la scelta del tipo DATE per il campo DATE_ID di Dimensione_Data, che può essere giunto a campi quali DATA_FATTURAZIONE di tipo TIMESTAMP. Firebird SQL infatti incoraggia l'uso del formato TIMESTAMP per le date, pur consentendo per compatibilità il formato DATA e il formato TIME ritenuti obsoleti. Nel caso della tabella Dimensione_data è importante garantire che sia riportata solo la data, senza informazioni accessorie come l'ora che potrebbero causare la mancata giunzione di record con stessa data ma diversa ora.

4.5 Analisi dei requisiti

Nelle moderne pratiche di progettazione software c'è un momento iniziale in cui gli sviluppatori definiscono i requisiti dell'applicazione finale. I criteri individuati e quindi effettivamente realizzati possono

decretare il successo oppure il fallimento del prodotto stesso.

Ci si limita in questa sede a stilare un elenco di utilizzatori / requisiti per una generica piattaforma di Business Intelligence, come linea giuda per valutare la qualità del software impiegato. Per i limiti imposti dalle risorse e gli scopi di questa tesi si evita di definire parametri molto oggettivi, che oltre a essere problematici da stilare sono anche troppo onerosi da valutare.

Tali punti saranno inoltre presi in considerazione durante la realizzazione degli applicativi, per avere un riferimento di cosa si aspettano dalla piattaforma gli utilizzatori finali.

Utente finale

- ambiente user-friendly
- possibilità di variare le analisi di base, applicando filtri o modificando la prospettiva
- frequenza di aggiornamento adeguata e trasparente
- disponibilità di strumenti di Business Intellingence potenti e flessibili (reportistica, Data Mining, analisi OLAP, ecc.)
- tempi ragionevoli nell'elaborazione dei dati
- visione chiara e completa in un numero ridotto di schermate
- possibilità di stampare i risultati e esportarli in formati comuni, come fogli Excel o file pdf
- possibilità di salvare l'analisi OLAP svolta e riprenderla in seguito
- restrizione d'accesso a persone autorizzate
- integrazione dell'ambiente con altri software, come pacchetti di office automation

- utilizzo di sistemi hardware/software già presenti o facilmente reperibili
- percepire rapidamente situazioni anomali, con l'utilizzo ad esempio di sistemi di alert o cruscotti interattivi

Amministratore di sistema

- software robusto e affidabile
- presenza di tools di configurazione che evitino la modifica di file testuali
- utilizzo di sistemi hardware/software già presenti o facilmente reperibili
- documentazione chiara e completa
- supporto esterno rapido in caso di malfunzionamenti (forum, supporto tecnico, ecc.)
- tempi di installazione ragionevoli
- ridotta manutenzione dell'ambiente una volta installato
- facilità nell'effettuare backup e ripristini

Sviluppatore dei documenti analitici

- tools user-friendly per creare applicativi
- facilità nell'installare e testare gli applicativi nella piattaforma
- utilizzo di linguaggi di programmazione diffusi o comunque facilmente comprensibili
- documentazione chiara e completa
- supporto rapido in caso di malfunzionamenti (forum, supporto tecnico, ecc.)

- facilità nell'apportare estensioni e modifiche agli applicativi già creati
- possibilità di debugging degli applicativi sia in caso di malfunzionamenti sia durante la realizzazione prima di pubblicarli
- disponibilità di strumenti di Business Intellingence potenti e flessibili (reportistica, Data Mining, analisi OLAP, ecc.)

4.6 Ambiente Hardware e Software

Verrà usato per l'ambiente di produzione finale il server Server01 della Carrozzeria Fellin, le cui caratteristiche tecniche sono elencate nella seguente tabella:

Nome di rete	Server01
Processore	Dual Intel Xeon 1.87 Ghz
RAM	1,5 Gb con controllo di errore ECC
Memoria di massa	3 dischi in Raid 5 per un totale di 150 Gbyte
Sistema operativo	Windows Server 2003 R2 con Service Pack 1
Java Virtual Machine	Sun Java JDK 1.6 Update 1

Sul Server01 è già installato il database Open Source Firebird versione 2.0.1. Questo database originariamente non è stato progettato per costruire un vero e proprio DataWarehouse come definito nel capitolo 2 soprattutto per la gestione del tempo nei record che andrebbe corretta aggiungendo flag di stato. Verrà comunque utilizzato come base dati per il progetto grazie alle caratteristiche tecniche che lo contraddistinguono quali:

- robustezza sotto pesanti carichi di lavoro

- ottima resa su database con dimensione dell'ordine di centinaia di megabyte come pure di decine di giga byte
- semplicità di amministrazione e bassissimo TCO (Total Cost of Ownership)
- supporto dalla comunità OpenSource e diffusione in realtà commerciali, nonché adesione a SQL standard ANSI '92
- ottime prestazioni in lettura, caratteristica essenziale in un ambiente DataWarehouse
- disponibilità di funzioni avanzate quali triggers, stored procedures, generatori, UDF (User Defined Functions), domains, integrità referenziale

I client della piattaforma sono macchine eterogenee, comunque equipaggiate dal sistema operativo Windows XP e il browser web Internet Explorer o Firefox.

La macchina di sviluppo e di test è un Notebook con le seguenti caratteristiche:

Nome di rete	Melphis
Modello	Acer Travelmate 291LCi
Processore	Intel Pentium M 1.400, tecnologia Intel Centrino
RAM	512 Mb
Memoria di massa	HDD EIDE 30 Gb
Sistema operativo	Windows XP Home Edition Service Pack 2
Java Virtual Machine	Sun Java JDK 1.6 Update 1

Anche sulla macchina di sviluppo è installato l'RDBMS Firebird 2.0.1 e il browser web Firefox 2.0.0.5.

La fase di ETL è un processo indipendente dalla piattaforma di

Business Intelligence implementata e può essere scelto di usare Pentaho Data Integration (Kettle) o Talend Open Studio oppure qualsiasi altro strumento con supporto ai database sorgenti e destinazione. L'unica annotazione è che PentahoBI consente di creare documenti analitici con applicativi Kettle, mentre SpagoBI permette di eseguire direttamente applicativi Talend Open Studio. Entrambi i software verranno esaminati dal punto di vista operativo durante la realizzazione del progetto.

La struttura informatica è già di per se sufficiente per ospitare una piattaforma come PentahoBI o SpagoBI.

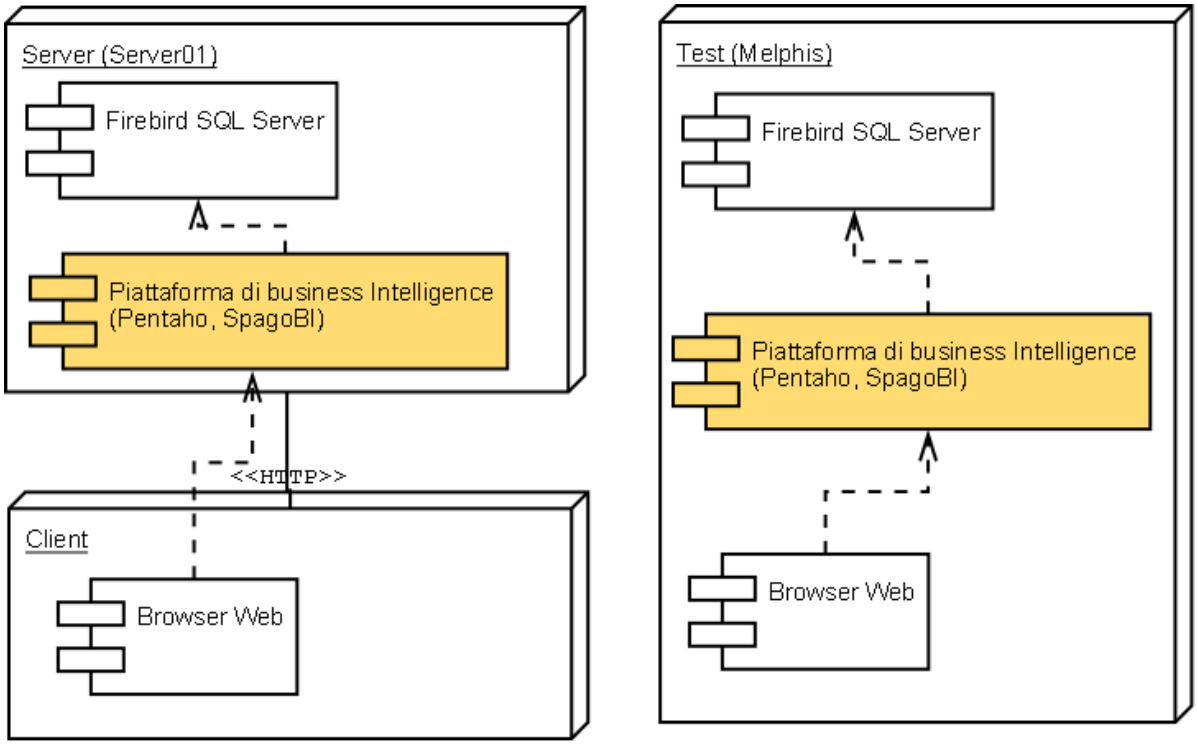


Illustrazione 35: Diagramma di implementazione della piattaforma

5 Realizzazione del progetto

Questo capitolo tratta della realizzazione pratica del progetto usando SpagoBI e PentahoBI. Seppur molto diverse strutturalmente, le due piattaforme condividono alcuni componenti come ad esempio

i report JasperReport e l'analisi OLAP con Mondrian/JPivot. Essendo la fase di design di tali documenti analitici un'operazione indipendente dalla piattaforma, la descrizione del procedimento è riportata nei paragrafi 5.2, 5.3 e 5.4, per poi descrivere con i successivi paragrafi i passaggi per l'integrazione nelle rispettive piattaforme.

5.1 Versioni dei software usati

Trattandosi di software in continua evoluzione, viene riportata a titolo di completezza l'elenco dei software usati con la rispettiva versione.

Software	Versione
Pentaho BI	1.2 GA
Spago BI	1.9.2
Pentaho Data Integration (Kettle)	2.5.0
Talend Open Studio	2.2.0 GA
iReport	2.0.0
Mondrian Schema Workbench	2.3.2
Firebird SQL	2.0.1
Jaybird Firebird JDBC driver	2.1.2

5.2 Alimentazione DataWarehouse

5.2.1 ETL con Kettle

Kettle, conosciuto anche con il nome Pentaho Data Integration, consente di realizzare trasformazioni ETL in un ambiente visuale fortemente orientato al drag & drop di componenti. La procedura

ETL si riduce alla creazione di uno o più progetti trasformazione, che vengono poi salvati su file. Un progetto trasformazione solitamente inizia con il reperimento dei dati da una o più tabelle, poi ci sono operazioni di trattamento dei dati e quindi si termina con l'aggiornamento della base dati di destinazione. Anche se si possono eseguire una ad una tutte le trasformazioni, è più comodo accorparle in un file Job che le richiami nel giusto ordine, cosa che verrà descritta nei prossimi paragrafi.

5.2.1.1 Tabella dei fatti Riparazioni

La parte più delicata e importante dell'intera base dati è l'aggiornamento della tabella Riparazioni, il cui schema è riportato nell'Illustrazione 34.

In Kettle si inizia definendo il collegamento ai database di WinCar, MyCar e il database analitico FellinBI, connessioni queste che vengono aggiunte nell'elenco “Database connections”.

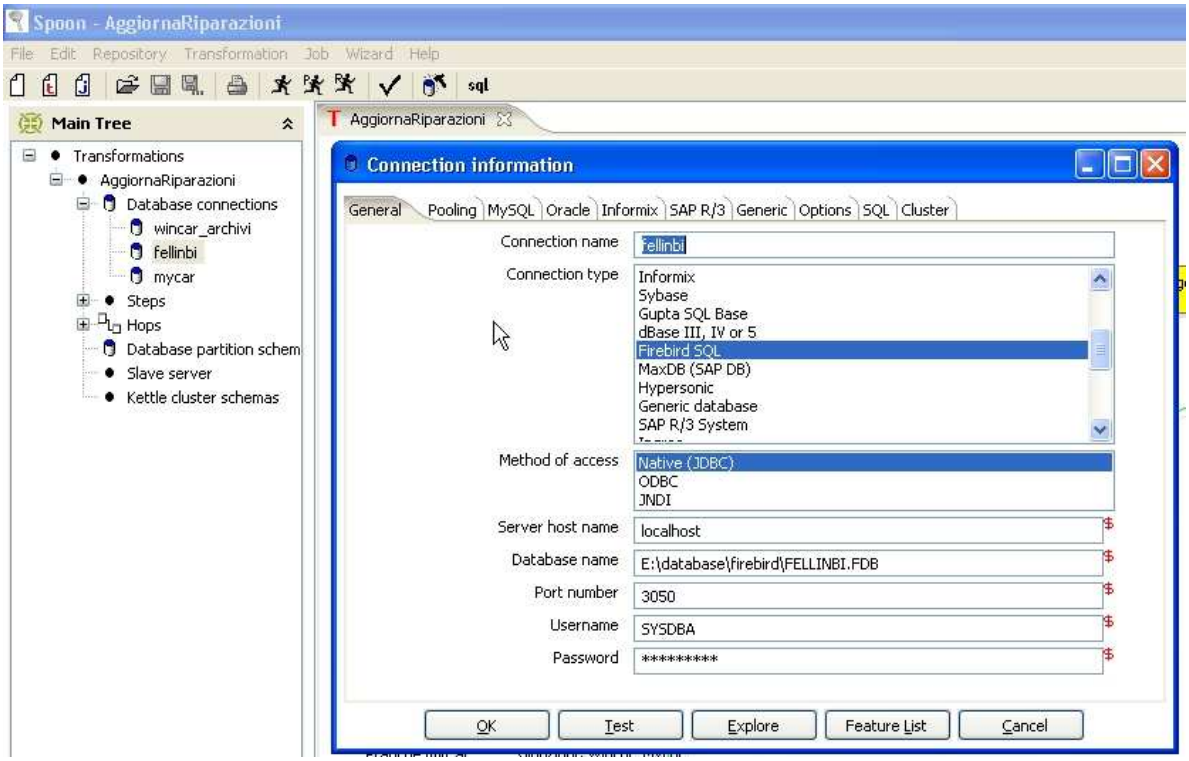


Illustrazione 36: Definizione di una connessione al database in Kettle

Si prosegue trascinando i componenti di interesse, personalizzandoli e collegandoli nell'ordine di Illustrazione 37. Per configurare un componente è sufficiente farvi un doppio click sopra, mentre per collegarlo a un altro si usa il pulsante centrale del mouse.

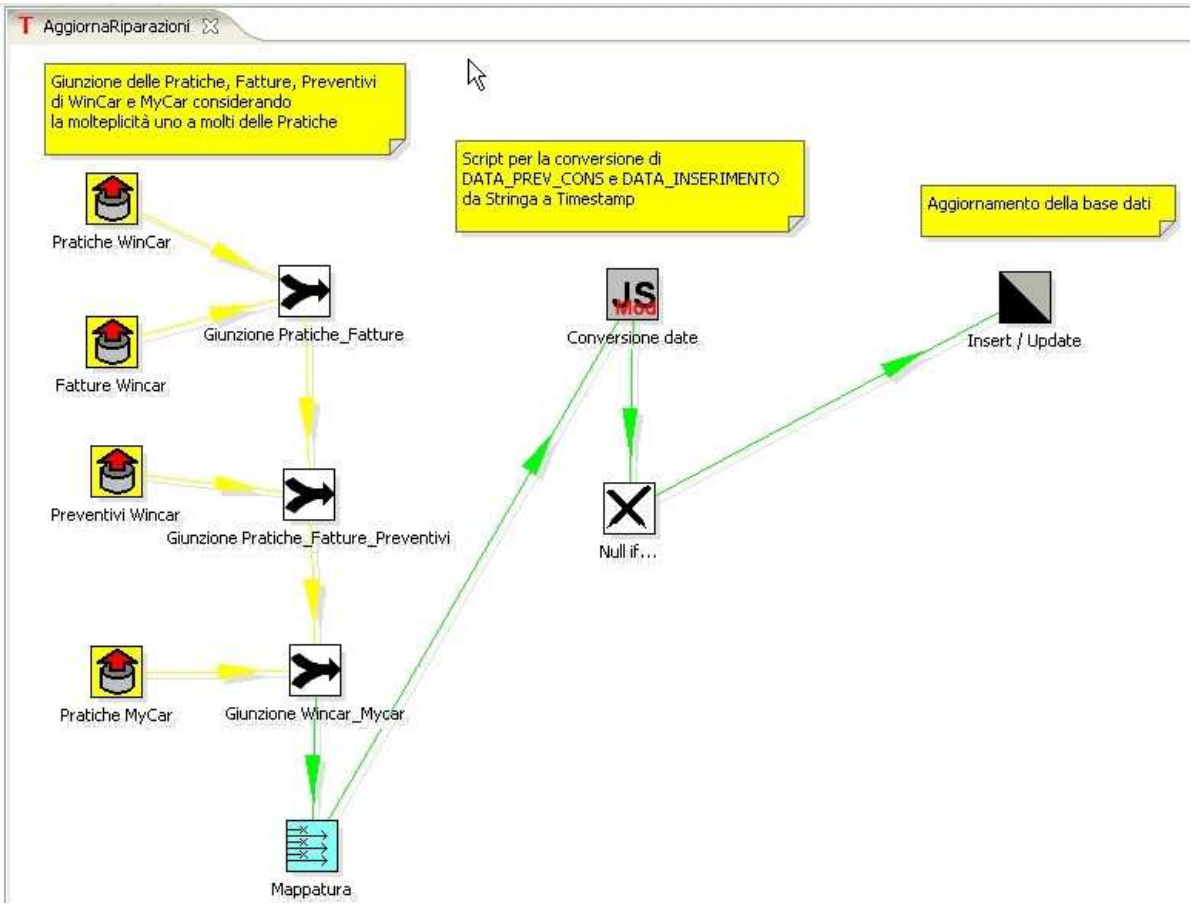


Illustrazione 37: La procedura di aggiornamento del cubo Riparazioni

Il dettaglio dei singoli componenti è il seguente:

- **Pratiche WinCar (Table Input)**

Estrae i dati dall'archivio pratiche di WinCar (tabella CARVEI) usando la seguente query:

```
SELECT C.F_NUMPRA, (C.F_CODMAR & C.F_CODMOD & C.F_CODVER) AS COD_VEICOLO,
C.F_CODCLI, C.F_CODPER, C.F_DATACA, C.F_DTPRCO
FROM CARVEI C
ORDER BY C.F_NUMPRA
```

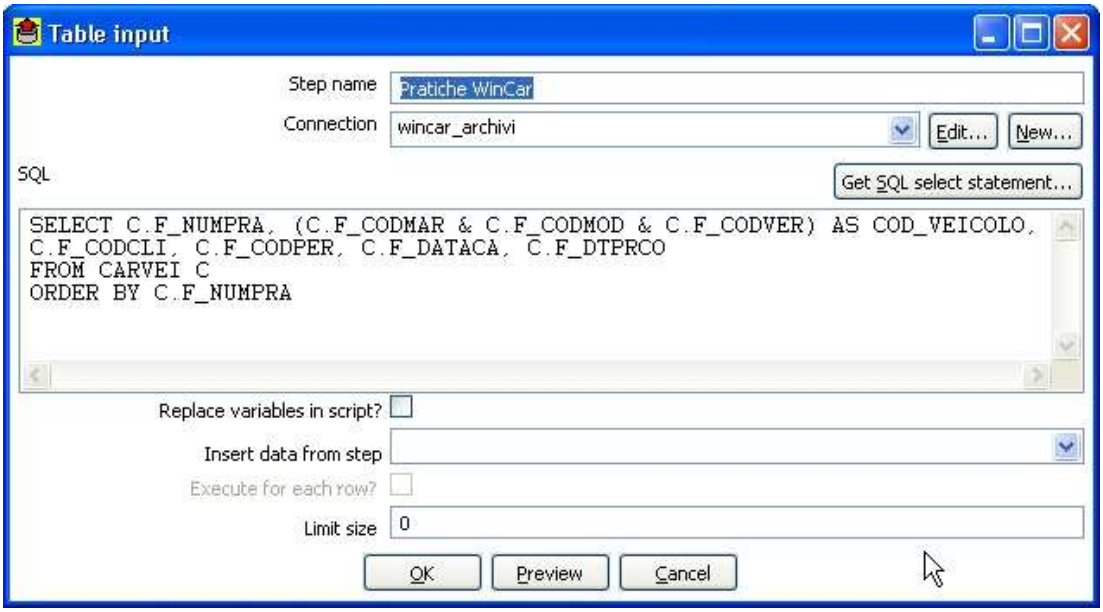


Illustrazione 38: Il componente Table Input "Pratiche WinCar"

Anche se i dati sono già ordinati per indice F_NUMPRA viene lasciata la clausola ORDER BY per garantire l'ordinamento delle chiavi richiesto dal successivo componente "Merge Join". Nella query avviene la giunzione dei campi F_CODMAR, F_CODMOD e F_CODVER convertiti a stringa e quindi uniti con l'operatore & a formare il campo COD_VEICOLO per i motivi tecnici citati nel paragrafo 4.4.

- **Fatture MyCar (Table Input)**

Vengono estratti i dati dalla tabella fatture di MyCar (tabella TESFAT). Il componente, come nel caso di "Pratiche WinCar", è sempre un Table Input con la seguente query:

```
SELECT F.F_NUMPRA AS F_NUMPRA2, SUM(F.F_TOTFAT) AS IMPORTO_FINALE,
MAX(F.F_DATFAT) AS DATA_FATTURAZIONE
FROM TESFAT F
WHERE F.F_NUMPRA > 0
GROUP BY F.F_NUMPRA
ORDER BY F.F_NUMPRA
```

La clausola GROUP BY evita la presenza di righe duplicate con lo stesso valore in F_NUMPRA. Nel caso ci siano più fatture emesse per singola pratica viene scelta la data dell'ultima fatturata (MAX(F.F_DATAFAT)) e la somma di tutti gli imponibili (SUM(F.TOTFAT)) .

- **Giunzione Pratiche_Fatture (Merge Join)**

Consente la giunzione di due sorgenti dati, in questo caso Pratiche MyCar e Fatture MyCar.

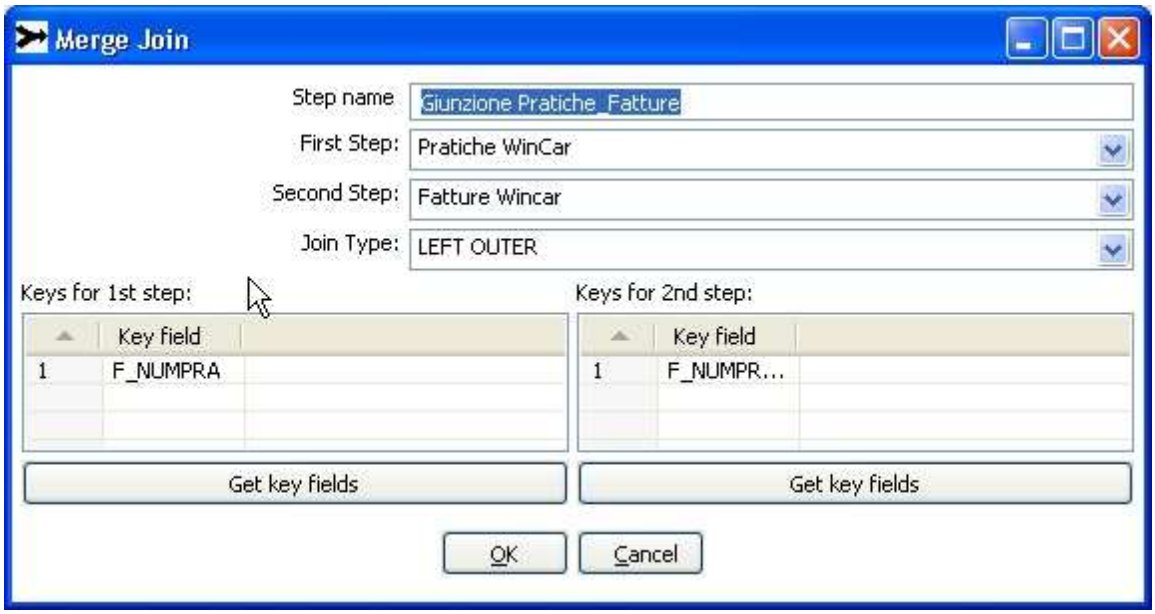


Illustrazione 39: Componente Merge Join per la giunzione di due sorgenti dati

La scelta di un LEFT OUTER join consente di estrarre i record delle pratiche anche se non esiste la corrispondente fattura nella tabella fatture.

- **Preventivi MyCar (Table Input)**

Si tratta ancora di un Table Input con la seguente query:

```
SELECT P.ID_CODPRE, LAST(P.F_TOTSR + P.F_TOTLA +P.F_TOTVE + P.F_TOTME) AS
```

```
TEMPO_TOT_PREV, LAST(P.F_TOTALE) AS IMPORTO_PREVENTIVO
FROM TESPRE P
GROUP BY P.ID_CODPRE
ORDER BY P.ID_CODPRE
```

La clausola GROUP BY garantisce ancora una volta l'unicità del campo ID_COPRE (ossia il numero di pratica) e in caso esistano più preventivi associati alla pratica viene scelto l'ultimo immesso (operatore LAST(P.F_TOTALE)).

- **Giunzione Pratiche_Fatture_Preventivi (Merge Join)**

Viene collegato l'output di Giunzione Pratiche_Fatture con l'output di Preventivi MyCar, usando il RIGHT OUTER JOIN per garantire la presenza dei preventivi anche se non sono fatturati.

- **Pratiche WinCar (Table Input)**

```
SELECT PRATICA_ID, ORE_LAVORATE, ORE_STRAORDINARIE, DATA_FINE_LAVORAZIONI,
0 as DELTA_GG_DAL_PREVENTIVO, 0 as DELTA_PREZZO_DAL_PREVENTIVO, 0 as
DELTA_ORE_DAL_PREVENTIVO
FROM PRATICHE_TESTATE_2
ORDER BY PRATICA_ID
```

A causa dei limiti del linguaggio SQL standard alcuni campi come come DELTA_GG_PREVENTIVO vengono impostati con il valore iniziale di 0 per elaborarli successivamente.

- **Giunzione WinCar_MyCar(Merge Join)**

Si tratta dell'ultimo join, che unisce i record delle tabelle fin qui elaborate. Il vantaggio di usare join a catena sta nell'ottenere sul singolo record tutti i campi di interesse, agevolando le elaborazioni

che richiedono colonne provenienti da database differenti.

- **Mappatura (Select Values)**

Molti nomi dei campi, soprattutto quelli provenienti dalle tabelle di MyCar, sono differenti rispetto ai nomi usati nella tabella Riparazioni. Si tratta di uno scenario molto comune, dove i nomi delle colonne della tabella di origine differiscono dalla tabella di destinazione. In Kettle il componente che permette la corretta mappatura dei campi è il “Select Values”, come si vede nella seguente figura:

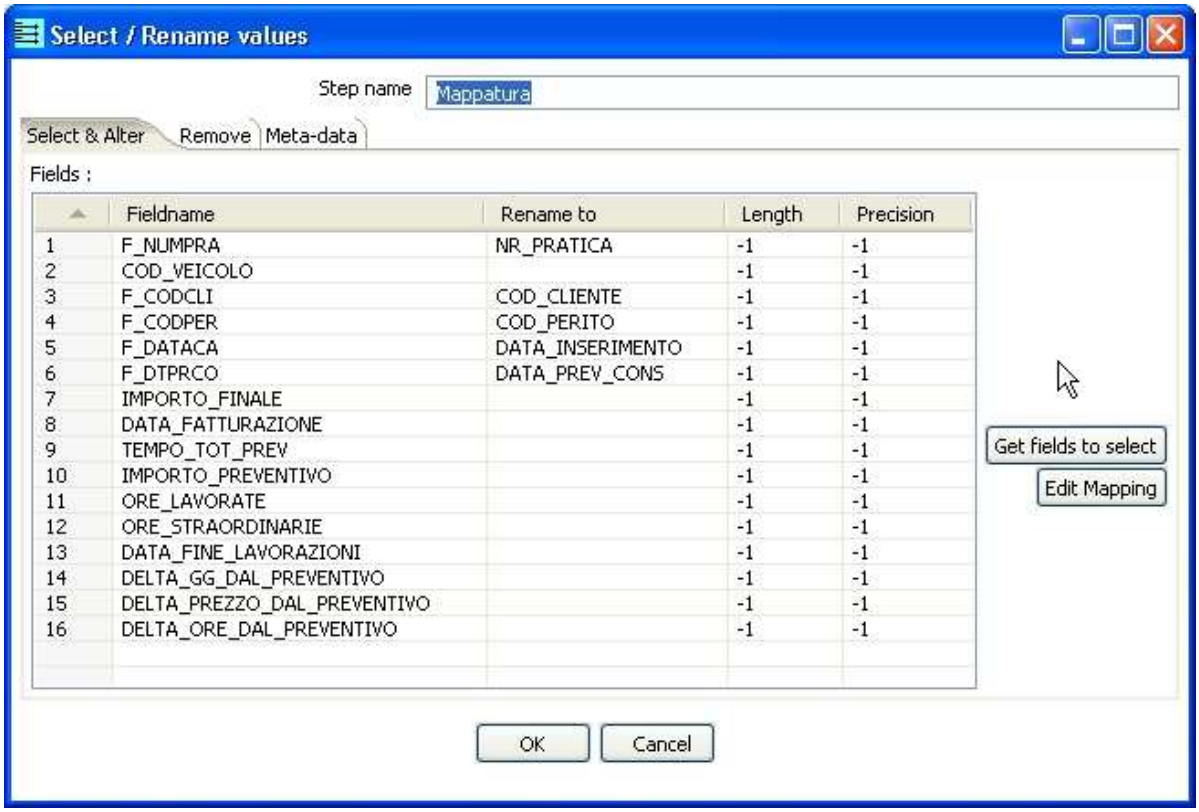


Illustrazione 40: Componente Select Values per la mappatura dei campi

- **Conversione date (Modified Java Script Value)**

DATA_PREV_CONS e DATA_INSERIMENTO sono nel formato stringa “gg/mm/aaaa” (giorno/mese/anno) incompatibile con Firebird e

molti altri RDBMS. L'uso del componente Modified Java Script Value effettua la conversione dei campi dal formato “gg/mm/aaaa” al formato “aaaa-mm-gg 00:00:00”[10] usando uno script Java Script.

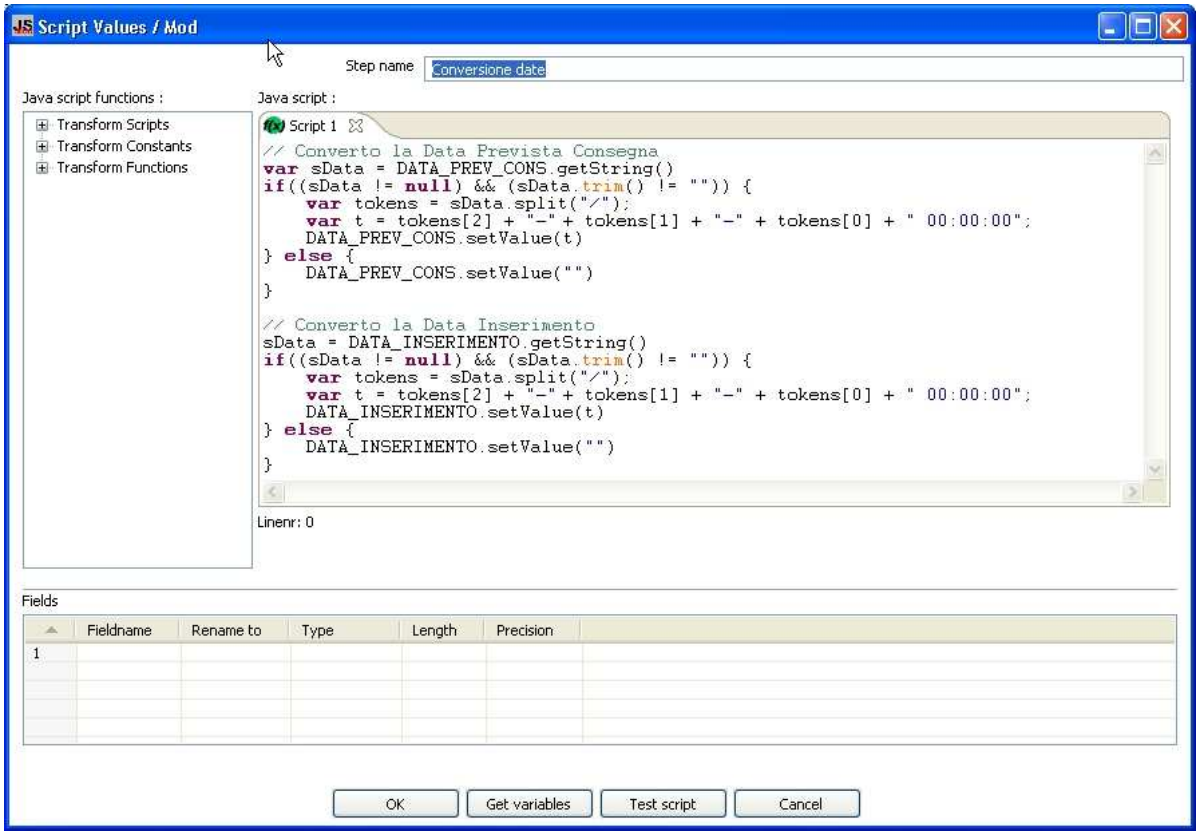


Illustrazione 41: Script per la conversione delle date

- **Null if.. (Null if..)**

Le date DATA_PREV_CONS e DATA_INSERIMENTO potrebbero assumere valori nulli, che in uscita dal passaggio Conversione Date vengono tradotti nella stringa vuota. Il componente “Null if..” riconosce la stringa vuota e la converte in valore null secondo la sintassi SQL.

- **Insert / Update**

10 Firebird consiglia di utilizzare come formato data il Timestamp, cioè il numero di secondi trascorsi dal 01 gennaio 1970. Questo formato viene specificato nella notazione “aaaa-mm-gg 00:00:00”.

Completa la procedura il componente Insert/Update per aggiornare la base dati.

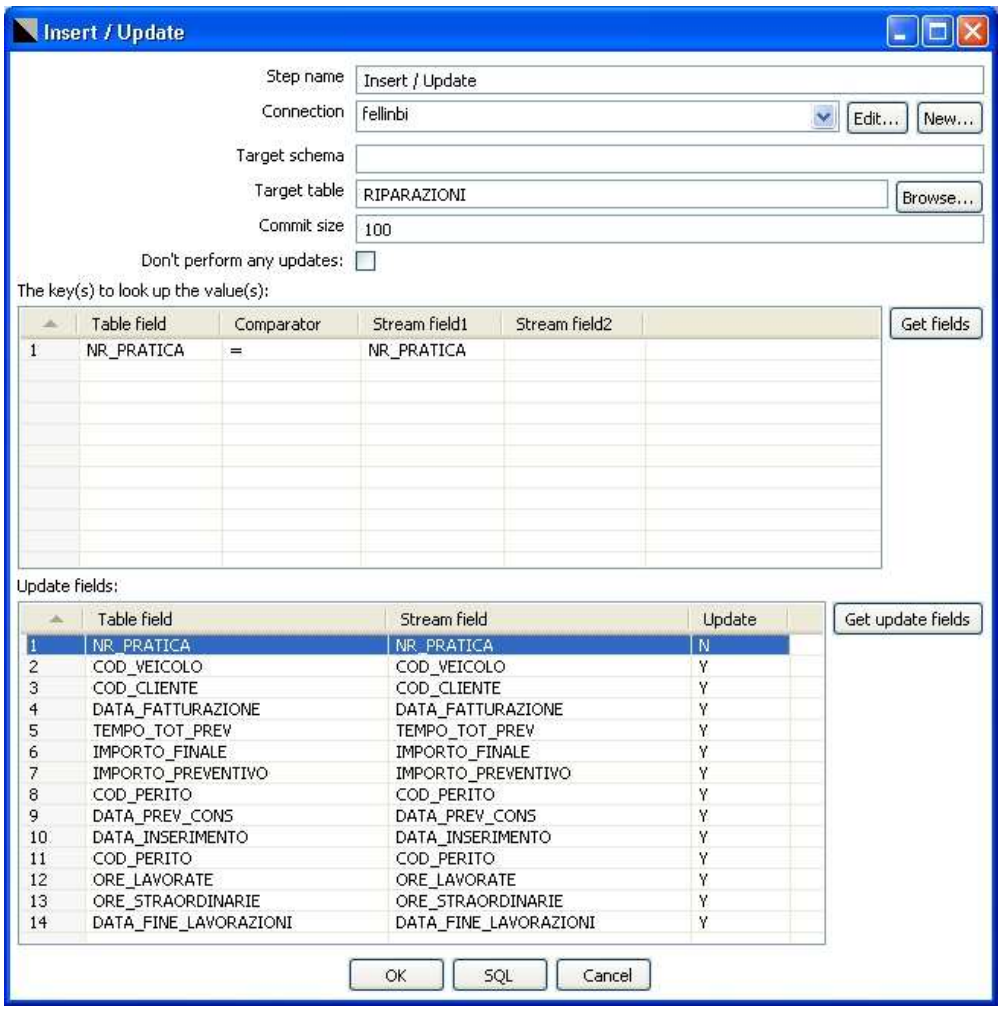


Illustrazione 42: Il componente Insert/Update

Manca in questo processo l'aggiornamento dei campi DELTA_GG_DAL_PREVENTIVO, DELTA_ORE_DAL_PREVENTIVO e DELTA_PREZZO_DAL_PREVENTIVO. La scarsa documentazione del componente Modified JavaScript Value rende l'operazione abbastanza complicata all'interno di Kettle e si è preferito creare un trigger in Firebird che ad ogni aggiornamento o inserimento nella tabella RIPARAZIONI calcoli i campi mancanti. Il codice del trigger è il seguente:

```
SET TERM ^ ;
```

```
CREATE TRIGGER RIPARAZIONI_BIU0 FOR RIPARAZIONI
ACTIVE BEFORE INSERT OR UPDATE POSITION 0
AS
begin
  /* delta gg dal preventivo */
  new.delta_gg_dal_preventivo = cast(new.data_fine_lavorazioni as date) -
cast(new.data_prev_cons as date);
  /* delta ore dal preventivo */
  new.delta_ore_dal_preventivo = (new.tempo_tot_finale -
new.tempo_tot_prev);
  /* delta prezzo dal preventivo */
  new.delta_prezzo_dal_preventivo = (new.importo_finale -
new.importo_preventivo);
end
^
```

5.2.1.2 Dimensione Data Fatturazione

Mondrian non permette di estrarre direttamente i livelli come anno, mese e quadrimestre da un campo data. Il rimedio sta nel creare un'apposita tabella contenente tali informazioni e giungerla usando un campo chiave di tipo data. La procedura per generare la tabella DIMENSIONE_DATA (il cui scema è riportato nell'Illustrazione 34) fa uso delle elaborazioni automatizzate di Kettle, come si può vedere in figura seguente:

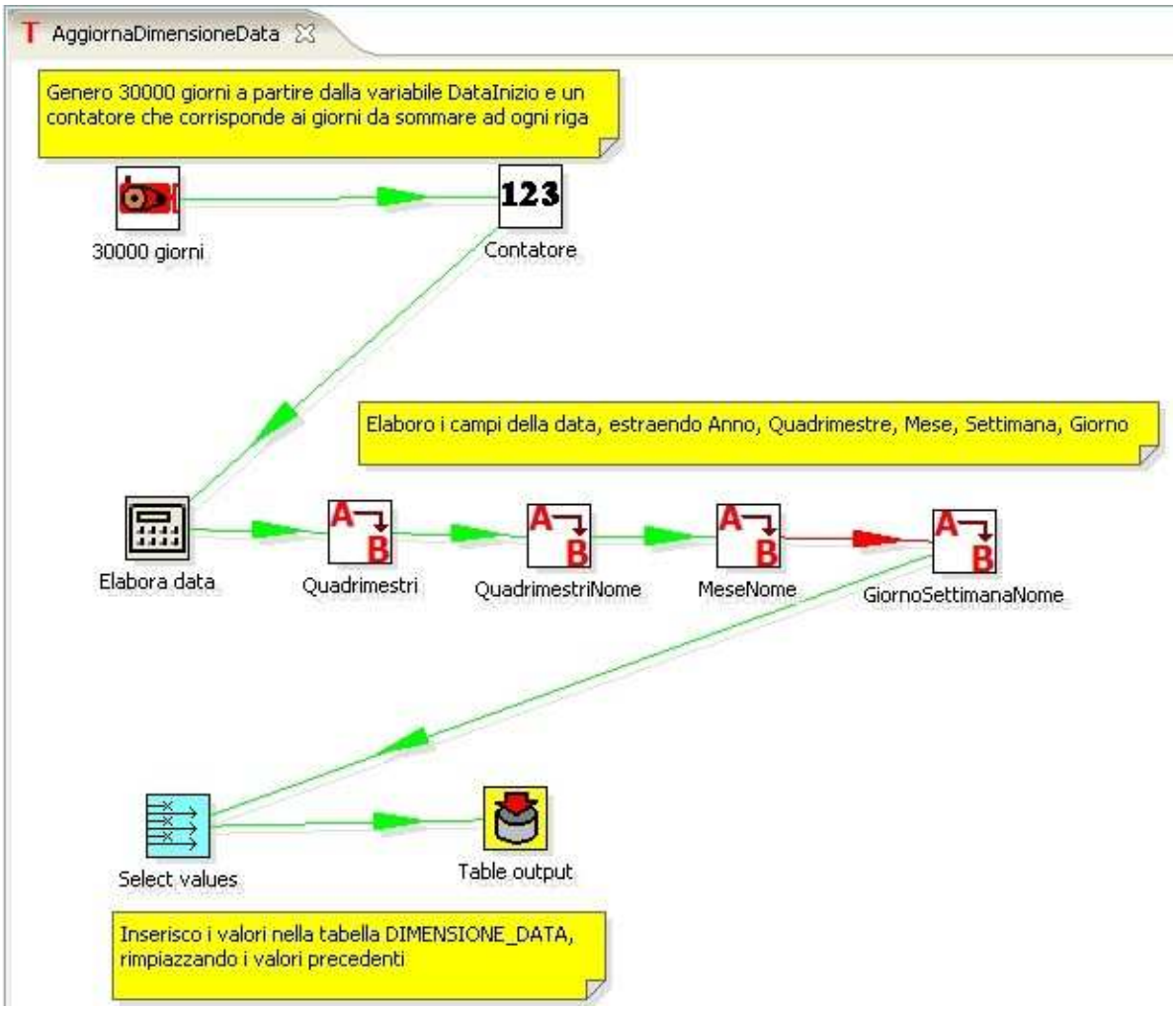


Illustrazione 43: Procedura per l'aggiornamento della dimensione data

Sinteticamente l'algoritmo per la generazione delle date è il seguente:

- Si generano 30000 record contenenti la data 01 Gennaio 1970
- Per ogni record viene incrementata la data 01 Gennaio 1970 di n giorni, dove n è il contatore dei record prodotti fino a quel momento
- Dalla data ottenuta si estraggono le variabili di interesse come anno, mese, giorno, eccetera

- **30000 giorni (Generate Rows)**

Genera 30000 record (in giorni circa 80 anni) contenenti il valore 19700101.

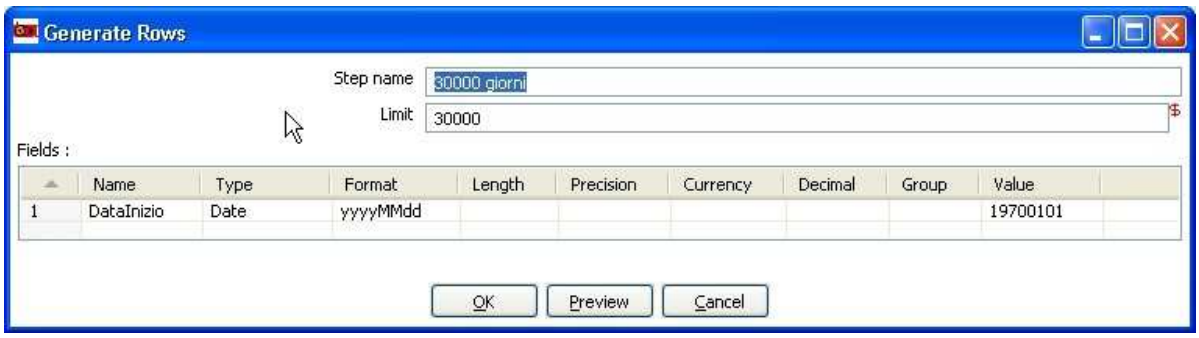


Illustrazione 44: Generazione di n record contenenti il valore 19700101

- **Contatore (Add Sequece)**

Conta il numero di record generati fino a quel momento e riporta il risultato nella colonna Contatore.

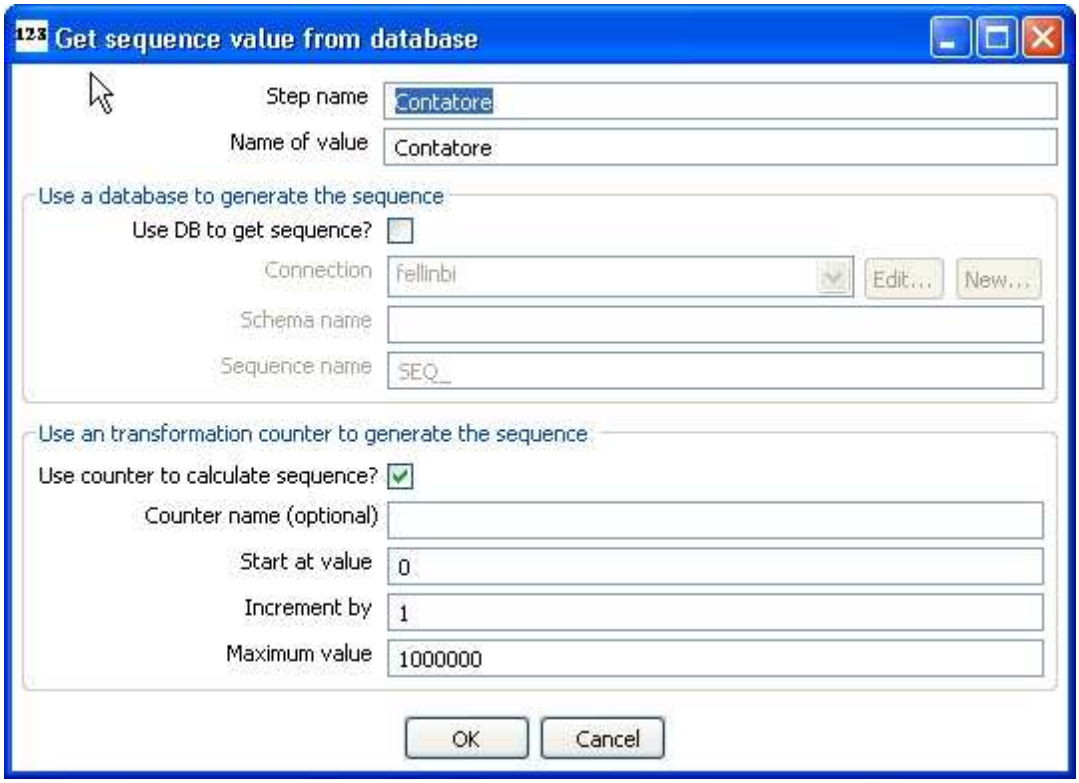


Illustrazione 45: Contatore dei record generati

- **Elabora data (Calculator)**

Il componente Calculator semplifica notevolmente l'elaborazione della data, sommando i giorni del contatore e estraendo i campi Anno, Mese, Settimana, Giorno e Giorno della settimana.

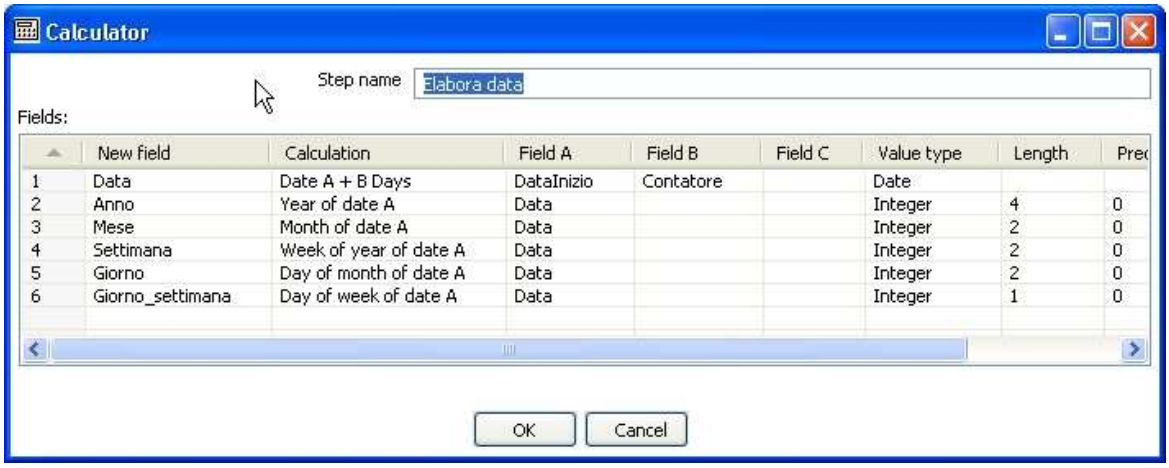


Illustrazione 46: Elaborazione della data con il componente Calculator

- **Quadrimestri, QuadrimestriNome (Value Mapper)**

Permette di popolare la colonna Quadrimestre con il numero del quadrimestre e la colonna QuadrimestreNome con una descrizione testuale del quadrimestre.

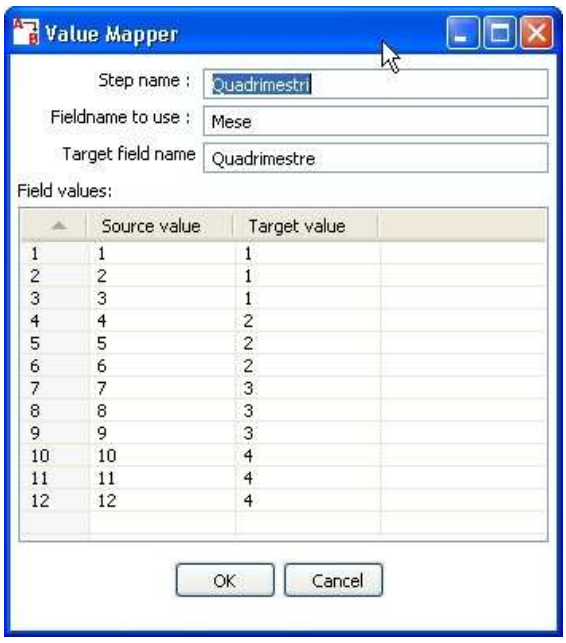


Illustrazione 47: Estrazione quadrimestri

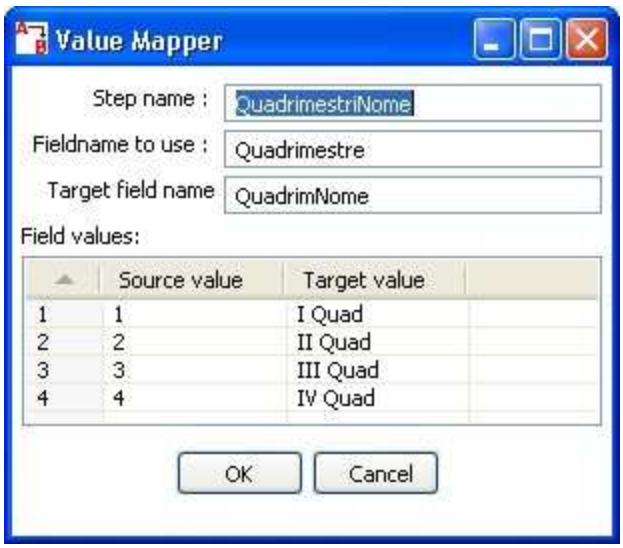


Illustrazione 48: Nomi dei quadrimestri

- **MeseNome, GiornoSettimanaNome (Value Mapper)**

Sono componenti Value Mapper che consentono l'estrazione rispettivamente del nome del mese e del giorno della settimana.

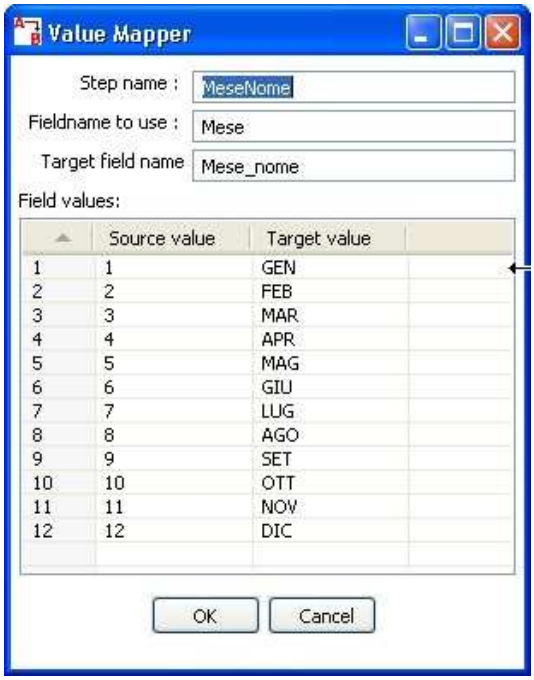


Illustrazione 49: Nome del mese

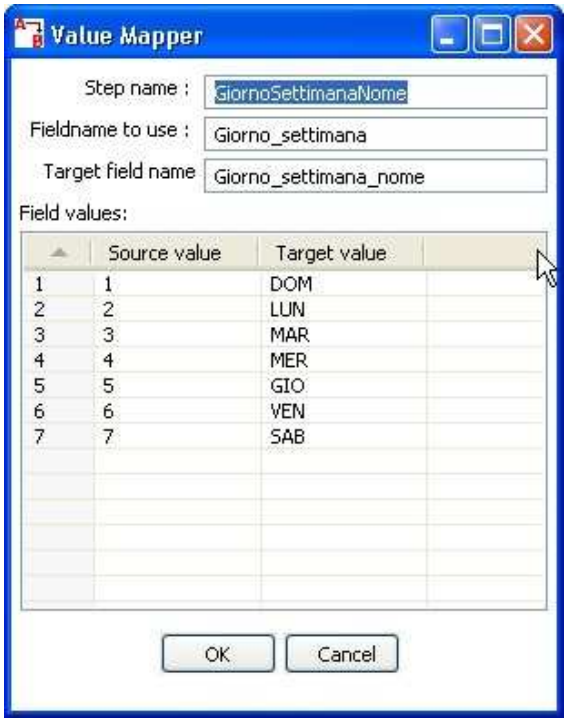


Illustrazione 50: Nome giorno settimana

- **Select Values (Select Values)**

Si occupa della mappatura dei nomi di colonna. Per il significato di questo componente si rimanda alle descrizioni precedenti.

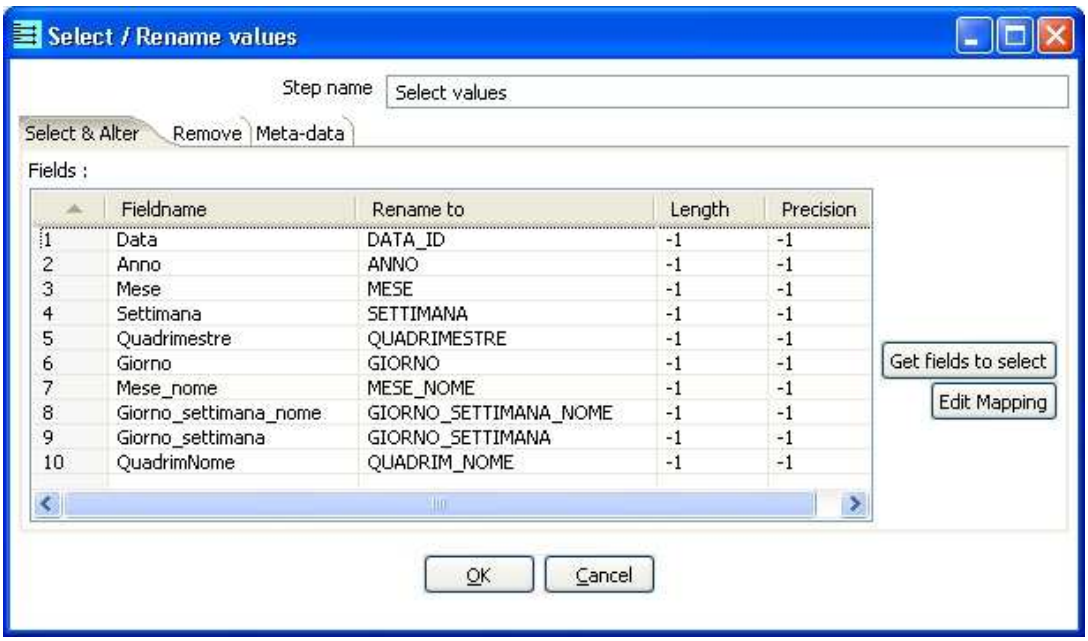


Illustrazione 51: Componente Select Values

- **Table Output (Table output)**

Il passo finale è quello di aggiornare la base dati. In questo caso si è preferito utilizzare al posto del classico strumento Insert / Update un componente Table Output con l'opzione truncate, il che significa che la tabella viene prima cancellata e poi ripopolata. Il fatto di svuotare la tabella ha senso se si pensa che DIMENSIONE_DATA viene popolata una tantum. Invece per le altre tabelle aggiornate di frequente è più efficiente usare un componente Insert / Update, che consente tra l'altro un caricamento trasparente all'utente finale.

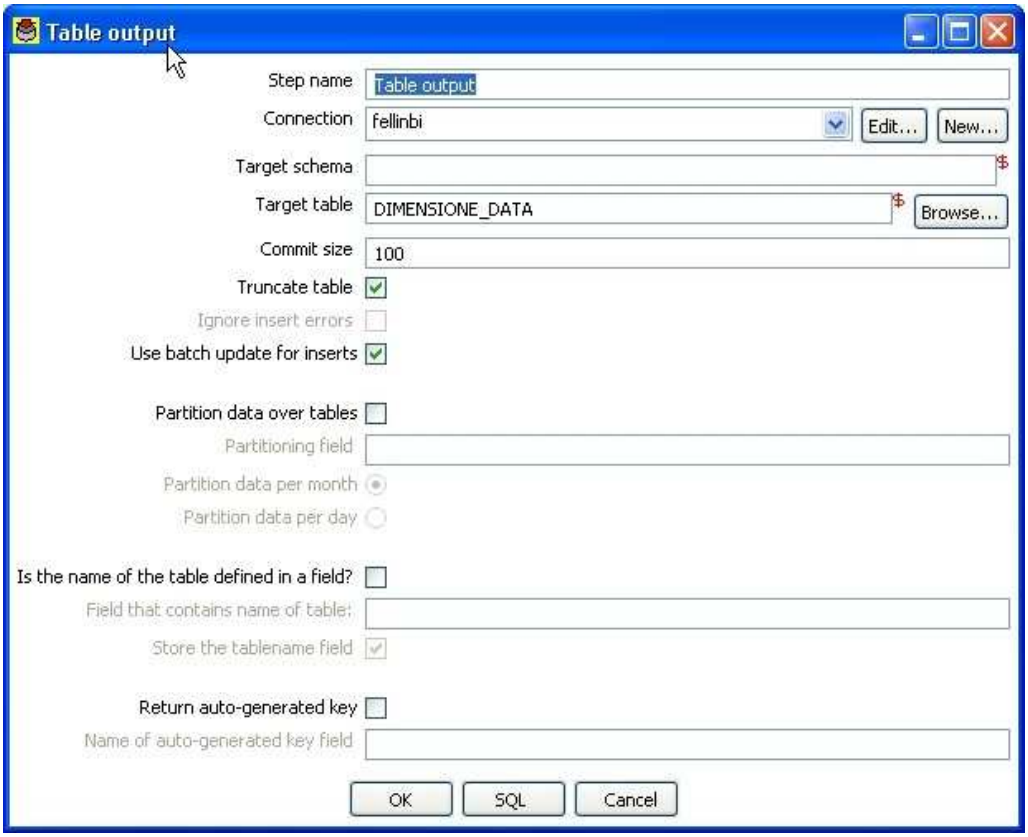


Illustrazione 52: Componente Table Output

5.2.1.3 Dimensione Clienti

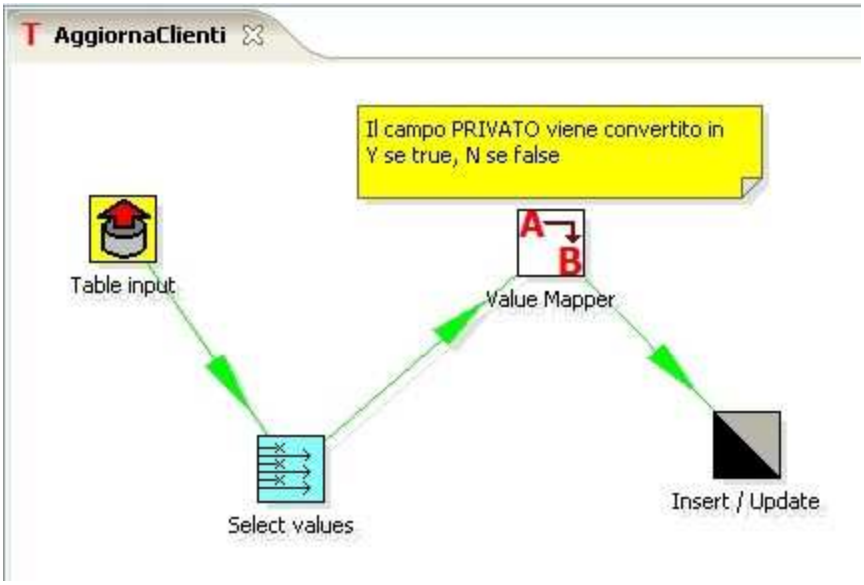


Illustrazione 53: Aggiornamento anagrafica clienti

Aggiornare le anagrafiche clienti è una operazione molto più semplice di quelle viste finora e si riduce a prelevare i dati di anagrafica clienti da WinCar con la seguente query:

```
SELECT F_CODCLI , F_RAGSOC , F_CITTAC , F_CAPCLI , F_PROCLI , F_PRIVATO
FROM CLIENT
```

Poi si prosegue con la mappatura del nome dei campi e l'aggiornamento tramite un componente Insert / Update dell'anagrafica CLIENTI.

Un passaggio intermedio all'aggiornamento sta nel convertire con un componente Value Mapper i campi di tipo Booleano True/False non supportati da Firebird in valori Y/N.

5.2.1.4 Dimensione Veicoli

L'anagrafica dei veicoli non è direttamente accessibile in WinCar per motivi strategici della software house produttrice, trattandosi di dati che hanno un notevole valore commerciale.

Al momento l'unica soluzione percorribile è di creare l'anagrafica usando la tabella pratiche di WinCar (tabella CARVEI) che non è completamente normalizzata in quanto la descrizione dei veicoli è accompagnata dal rispettivo codice.

La trasformazione usata è quella dell'Illustrazione 54, dove sul componente Table Input è impostata la seguente query:

```
SELECT (F_CODMAR & F_CODMOD & F_CODVER) as COD_VEICOLO,
UCASE(LAST(F_DESMAR)) AS DESCR_MARCA, UCASE(LAST(F_DESMOD)) AS
DESCR_MODELLO, UCASE(LAST(F_DESVER)) AS DESCR_VERSIONE
FROM CARVEI
WHERE F_DESMAR <> '' and F_DESMOD <> ''
GROUP BY (F_CODMAR & F_CODMOD & F_CODVER)
```

Con GROUP BY si garantisce l'unicità del campo chiave e nel caso di

record ripetuti vengono scelti marca, modello e versione dell'ultimo record immesso (operatore LAST seguito dall'operatore UCASE per convertire la stringa in maiuscolo). Inoltre la colonna COD_VEICOLO è risultata dalla concatenazione di F_CODMAR, F_CODMOD e F_CODVER per i motivi descritti nel paragrafo 4.4.

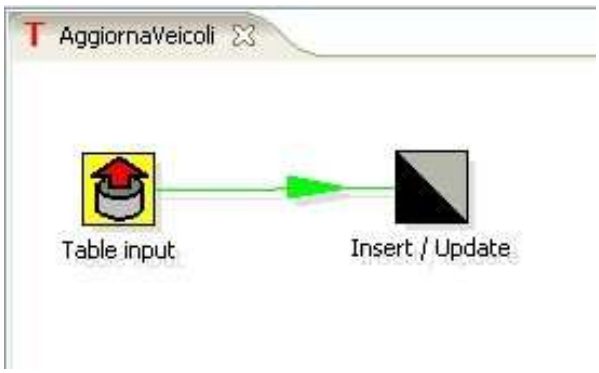


Illustrazione 54: Aggiornamento anagrafica veicoli

Non è richiesto in questo caso la mappatura dei nomi delle colonne perché la tabella destinazione ha gli stessi nomi del risultato della query.

5.2.1.5 Dimensione Perizie

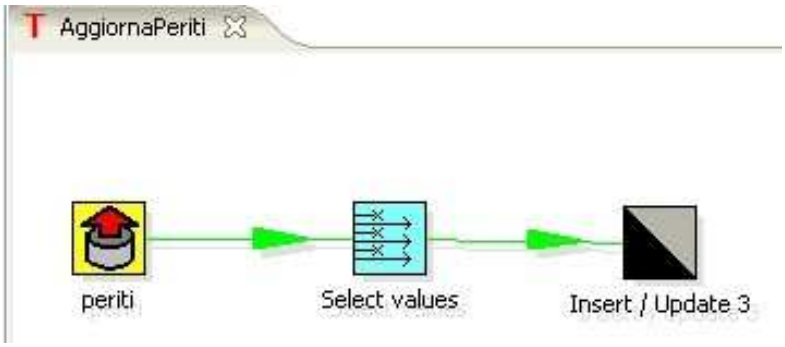


Illustrazione 55: Aggiornamento anagrafica periti

L'aggiornamento dell'anagrafica periti consiste nell'estrarre i dati dall'anagrafica di MyCar e aggiornare la base dati con un componente Insert / Update. La query è la seguente:

```
select F_CODPER, F_NOMPER from PERITI
```

5.2.1.6 Integriamo il tutto

Abbiamo finora creato delle procedure di aggiornamento salvate su singolo file che possono essere eseguiti con il comando kitchen.bat da MS-DOS o kitchen.sh da Shell UNIX:

```
kitchen.bat /file:<nome del file trasformazione>
```

Risulta comodo raggruppare le trasformazioni in un singolo file Job, come nell'Illustrazione 56. Il progetto Job segue i principi di un normale progetto trasformazione, con la differenza che vengono collegati sequenzialmente componenti utili all'esecuzione di singole trasformazioni come si vede nell'Illustrazione 57.

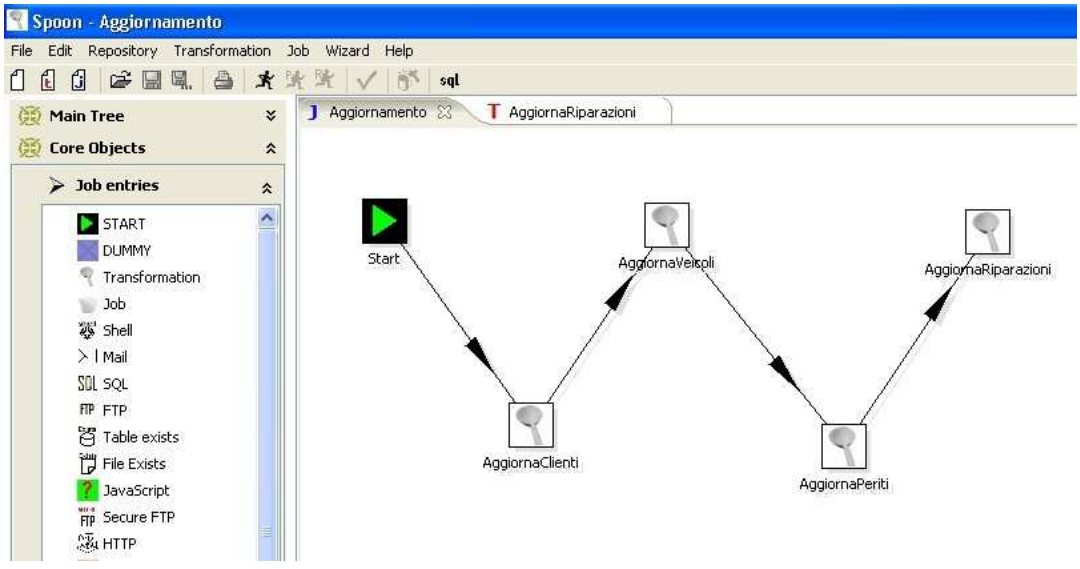


Illustrazione 56: Un progetto Job integra le singole trasformazioni

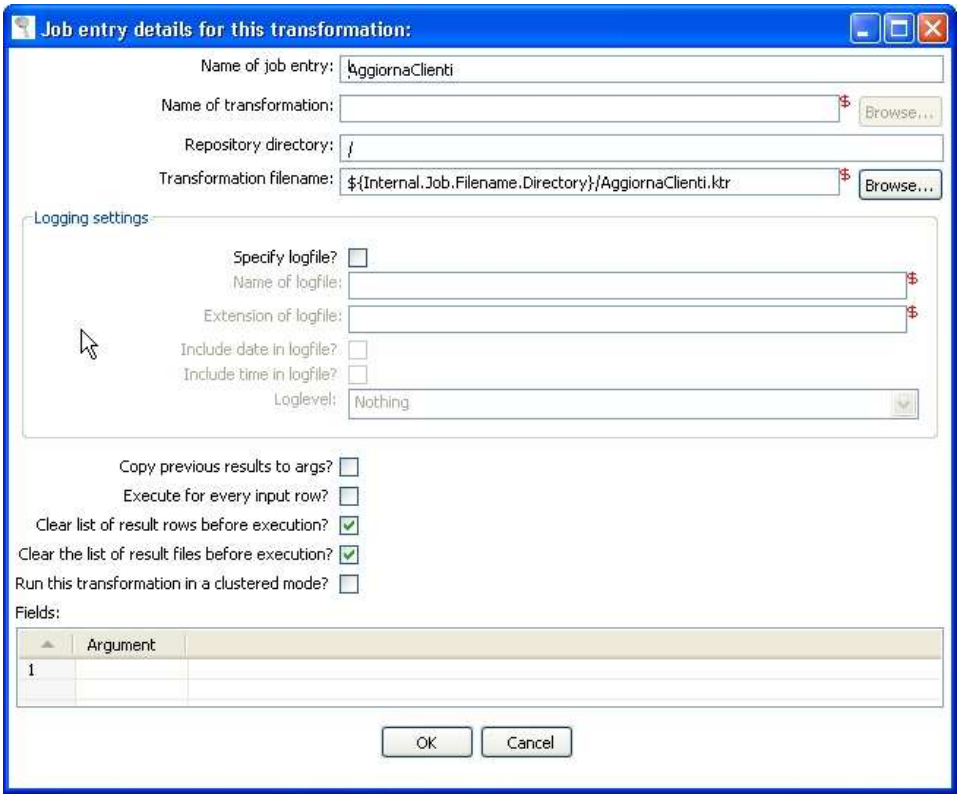


Illustrazione 57: Dettaglio di un componente Transformation

Il progetto Job una volta salvato su file può essere lanciato con il comando kitchen.bat (kitchen.sh su UNIX):

```
kitchen.bat /file:<nome del file job> /norep
```

Il flag /norep indica che il file va cercato nel disco e invece che in una repository di Pentaho BI.

Per l'aggiornamento del DataWarehouse a intervalli di tempo prefissati con Kettle esistono due alternative:

- si utilizza un programma esterno per eseguire il comando Kitchen a intervalli di tempo prefissati
- si utilizzano le funzioni di schedulazione di Kettle creando un progetto Job con il componente Start personalizzato

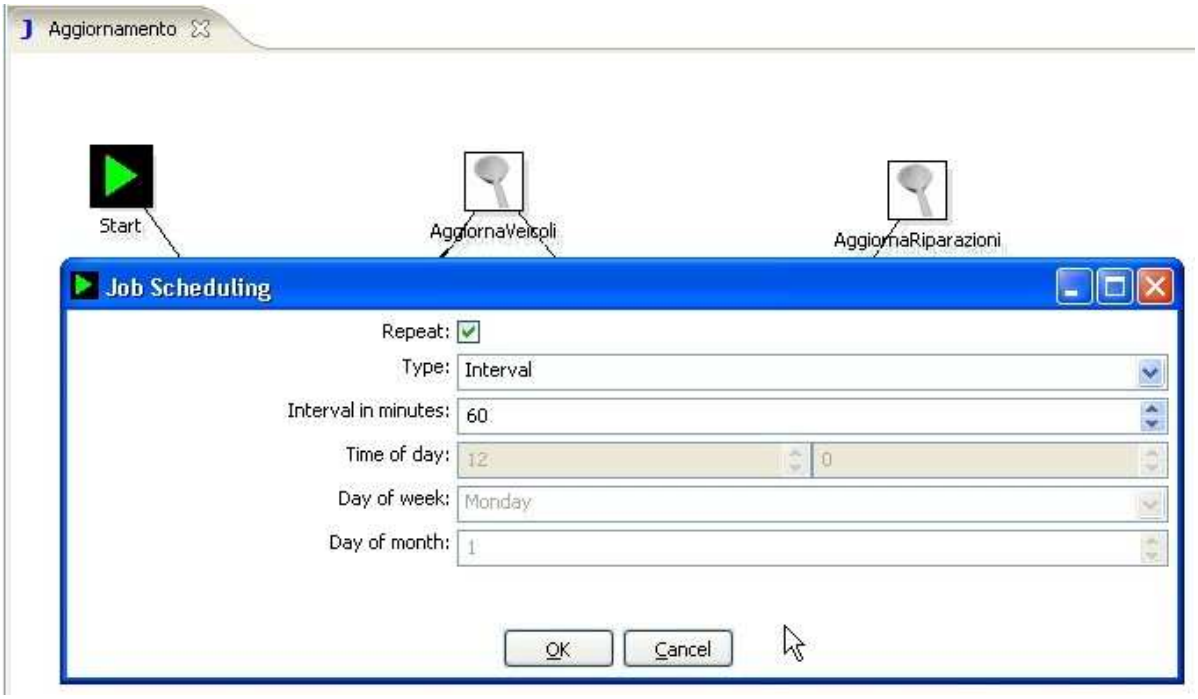


Illustrazione 58: Progetto Job con esecuzione a intervalli regolari di 60 minuti

Nella mancanza di uno schedulatore completo e funzionale per Windows si è preferito in questo caso optare per la seconda alternativa, impostando un tempo di aggiornamento di 12 ore come definito nel paragrafo 4.2. Rimane fuori dal progetto job la trasformazione per l'aggiornamento di Dimensione_data, eseguita una tantum dall'amministratore di sistema.

5.2.2 ETL con Open Studio

Il processo di ELT per popolare lo schema di Illustrazione 34 verrà riproposto nella variante Talend Open Studio. Come prima operazione è necessario creare un nuovo progetto che farà da raccoglitore non solo per le le procedure ETL ma anche per le librerie di funzioni personalizzate nonché gli schemi e tutta la documentazione a corredo. Open Studio grazie a questa e altre caratteristiche si può ritenere un IDE (Integrated Development

Enviroment) che non si limita alla sola programmazione delle procedure ETL ma fornisce una serie di strumenti utili all'intero processo, il tutto integrato in un unico ambiente. Durante la fase di creazione del progetto è richiesta la scelta del linguaggio in cui verrà generato il codice, che può essere Perl oppure Java.

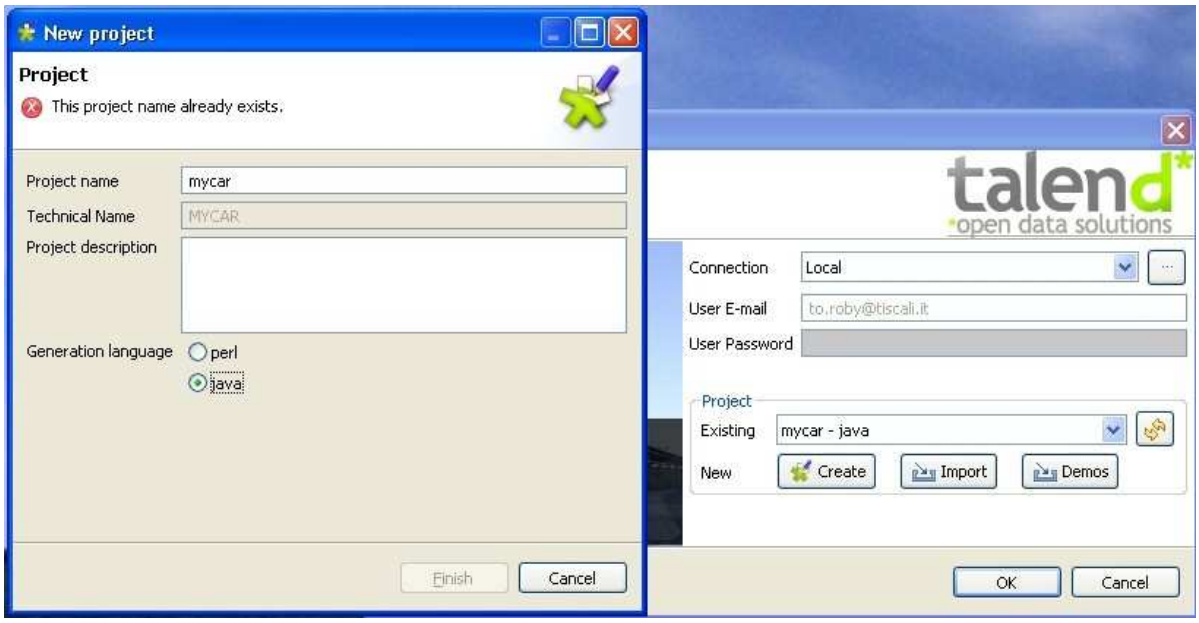


Illustrazione 59: Definizione di un progetto in Open Studio

L'adozione di Java o Perl dipende soprattutto dalle competenze degli sviluppatori, nel nostro caso si utilizzerà Java per la maggior diffusione in ambienti commerciali, lo stile di programmazione orientato agli oggetti e le performance leggermente superiori al concorrente. Perl rimane comunque un'ottima scelta, soprattutto per la notevole espressività del linguaggio e la manipolazione delle stringhe, che ne hanno reso l'unico linguaggio disponibile in Talend Open Studio fino alla versione 2.0.

Una volta scelto il progetto su cui lavorare si presenta all'utente l'ambiente con tutte le sue funzionalità. Prima di realizzare una procedura ELT è necessario registrare il collegamento alla fonte dati cliccando con il pulsante destro del mouse su “Db connections”

(sottovoce di “Metadata” nel pannello “Repository”). I collegamenti alle sorgenti dati realizzati in questo progetto sono gli stessi del caso di Kettle e precisamente:

- **wincar_tabelle** Il database wcTabelle di WinCar
- **wincar_archivi** Il database wcArchivi di WinCar
- **mycar** il database del programma MyCar
- **fellinbi** il DataWarehouse di questo progetto

Nell'Illustrazione 60 si nota il ricco set di database supportati in Open Studio, pur garantendo con i driver ODBC e JDBC già un'amplissima gamma di RDBMS.

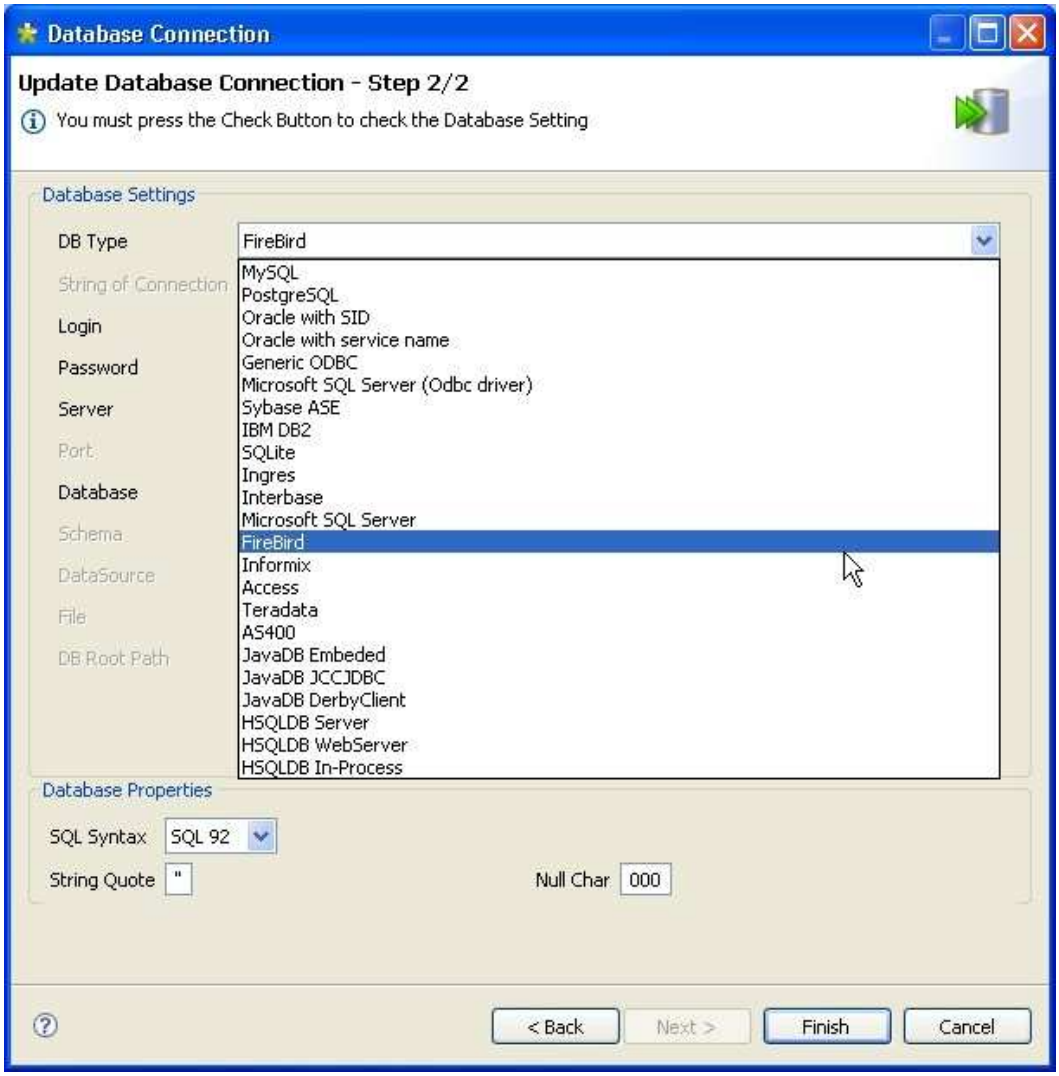


Illustrazione 60: Definizione di un collegamento al database

Come riportato nel paragrafo 3.2.5 Talend Open Studio pone particolare attenzione alla correttezza degli schemi durante tutto il design delle procedure ETL. La maggior parte dei componenti che interagiscono con il flusso dati permettono di definire la struttura dei record, che in uscita deve essere uguale allo schema in entrata del componente successivo. Per evitare di specificare di volta in volta la struttura dati dati è possibile reperire gli schemi direttamente dal RDBMS e salvarli nella repository “Metadata”. Il tutto avviene cliccando con il pulsante destro del mouse sulla collegamento alla base dati e scegliendo “Retrieve Schema”.

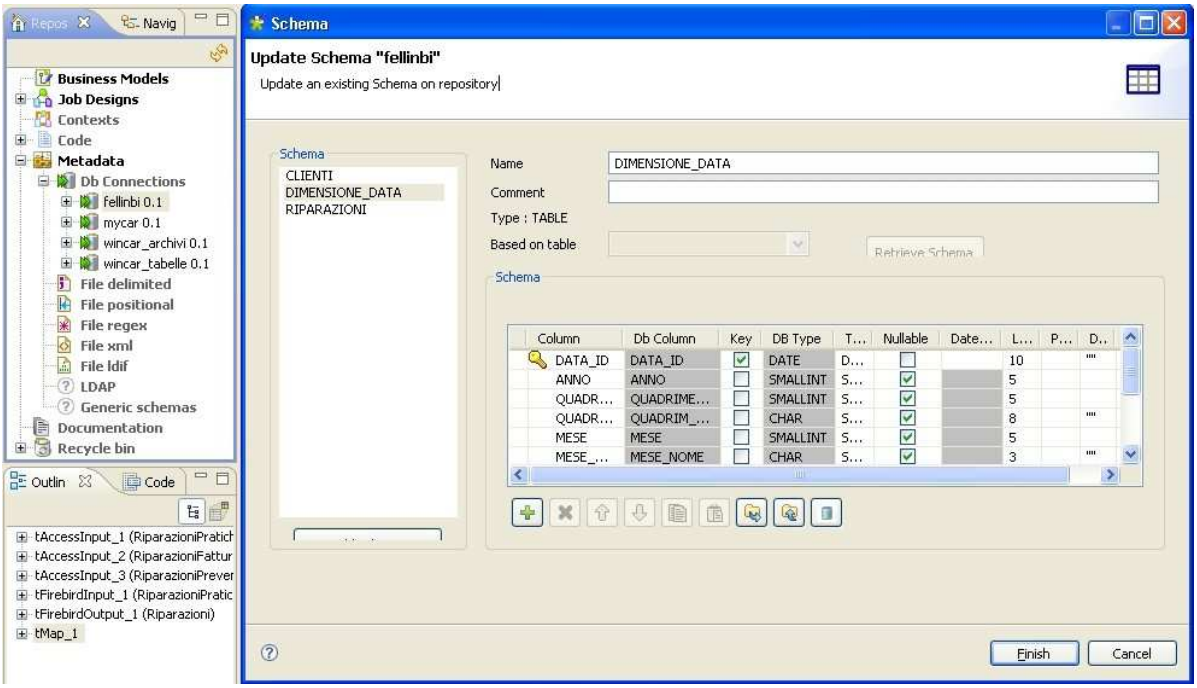


Illustrazione 61: Funzione Retrieve Schema da una connessione database

Una volta definiti gli schemi delle tabelle e i collegamenti ai RDBMS si può proseguire con la definizione delle singole procedure ETL.

5.2.2.1 Tabella dei fatti Riparazioni

Per aggiornare la tabella dei fatti riparazioni viene creato un apposito job cliccando con il pulsante destro del mouse su “Job Designs” e quindi su “Create job”. Interessante è la comparsa di mascherine guidate quando vengono aggiunti contenuti nelle varie repository. In tal modo è possibile attribuire informazioni aggiuntive come autore, versione del file, brevi descrizioni eccetera. Nel caso dei nuovi jobs la mascherina che appare è la seguente:

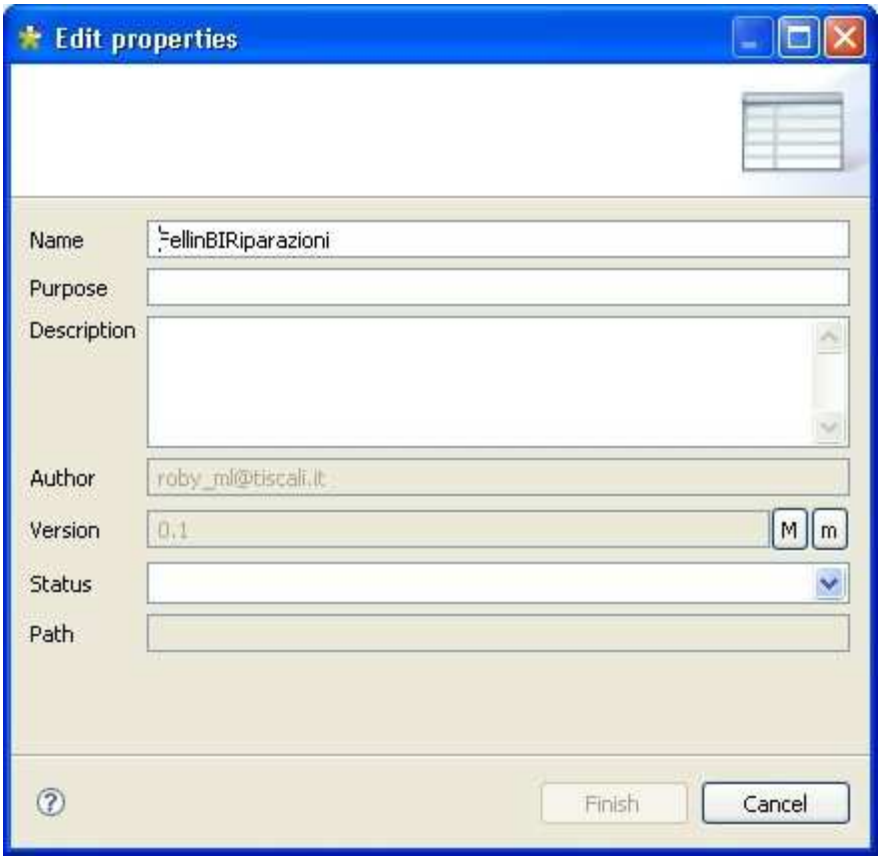


Illustrazione 62: Creazione di un nuovo Job

A tal punto verrà visualizzato un foglio di lavoro vuoto, che l'utente provvederà a completare trascinando gli elementi del riquadro “Palette” e collegandoli in senso sequenziale di esecuzione. Il collegamento avviene cliccando con il pulsante destro del mouse sopra il componente e scegliendo il tipo di collegamento da effettuare. A differenza di Kettle che consente solo un tipo di collegamento per trasmettere una riga generata alla volta, Open Studio fornisce ben sei varianti (collegamento di tipo Main, Iterate, Then run, Run if, Run if Ok, Run if Error). È così possibile mandare il flusso di esecuzione al componente successivo riga per riga (collegamento Main o Iterate), oppure quando il lavoro del componente è terminato (collegamento Then run), o anche al verificarsi di determinate condizioni (collegamenti Run if..). Nel nostro progetto verranno utilizzati, se non diversamente specificato,

solo collegamenti di tipo Main.

Il job per l'aggiornamento della tabella Riparazioni nel nostro caso appare come nella seguente illustrazione:

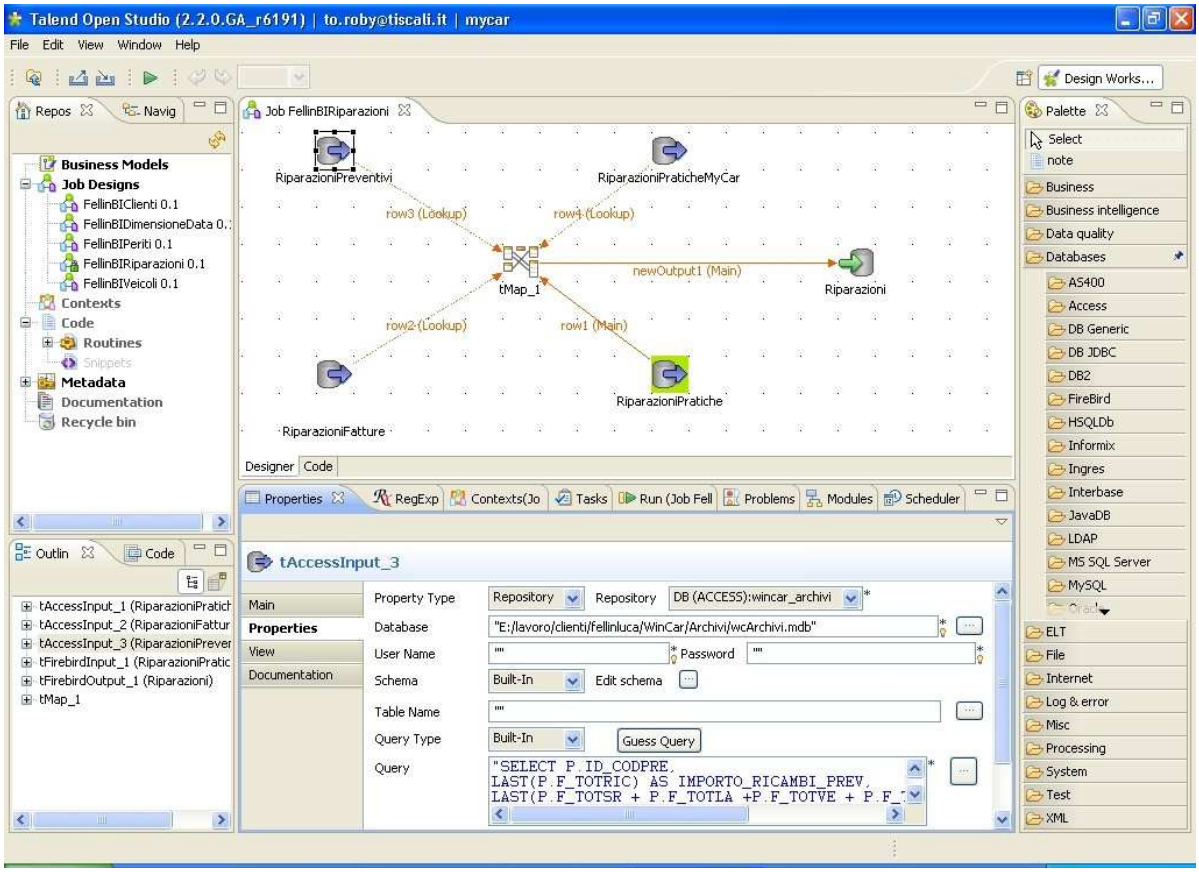


Illustrazione 63: Procedura di aggiornamento tabella Riparazioni

Singolarmente i componenti vengono configurati nel seguente modo:

- **RiparazioniPreventivi (tAccessInput)**

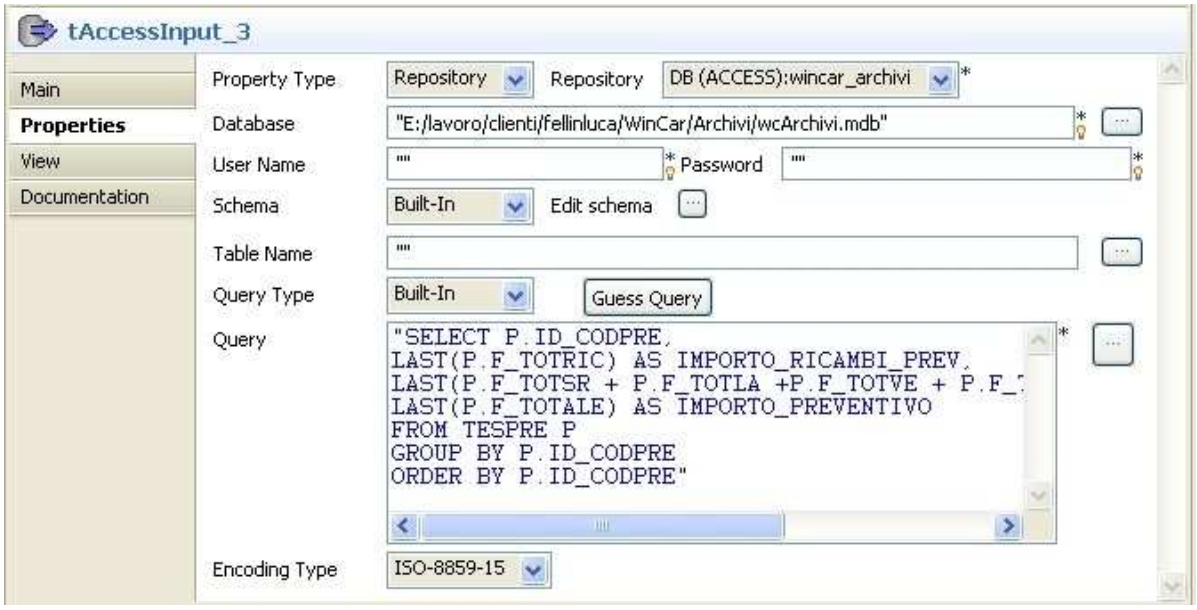


Illustrazione 64: Proprietà di RiparazioniPreventivi

Questo componente consente di estrarre i record dalla tabella preventivi di WinCar. La query è la stessa vista nel caso di Kettle, con l'uso delle clausole GROUP BY ID_CODPRE e LAST per garantire l'unicità della chiave ID_CODPRE. A differenza di Kettle, che reperisce gli schemi automaticamente, nel caso di Open Studio l'utente è chiamato ad attribuire uno schema valido ai dati in uscita. L'operazione, che va ripetuta per ogni componente che estragga o aggiorni basi dati, viene effettuata compilando la proprietà “Schema” con le voci salvate in repository. Cliccando sul pulsante “Edit schema” a fianco della proprietà è pur sempre possibile personalizzare uno schema preesistente, che nel nostro caso diventa come nella Illustrazione 65.

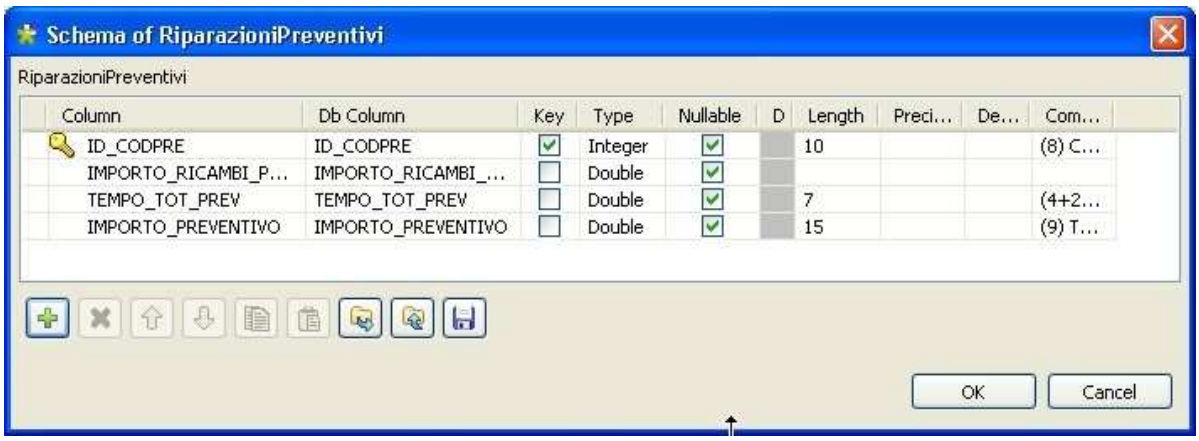


Illustrazione 65: Definizione dello schema RiparazioniPreventivi

- **RiparazioniFatture (tAccessInput)**

Questo componente serve al prelievo dei dati dalla tabella fatture di Wincar e viene impostato come nel caso precedente, con la proprietà query alla seguente stringa:

```
"SELECT F.F_NUMPRA, MAX(F.F_DATFAT) AS DATA_FATTURAZIONE, SUM(F.F_TOTFAT)
AS IMPORTO_FINALE
FROM TESFAT F
WHERE F.F_NUMPRA > 0
GROUP BY F.F_NUMPRA ORDER BY F.F_NUMPRA"
```

- **RiparazioniPraticheMyCar (tFirebirdInput)**

I componenti per l'input da database anche se destinati a RDBMS diversi hanno proprietà simili. In questo caso tFirebirdInput per il prelievo dei dati dalla tabella PRATICHE_TESTATE_2 è simile al tAccessInput visto per il database Microsoft Access. Il componente personalizzato come nella Illustrazione 66 permette di reperire dati dalla tabella delle pratiche di MyCar.

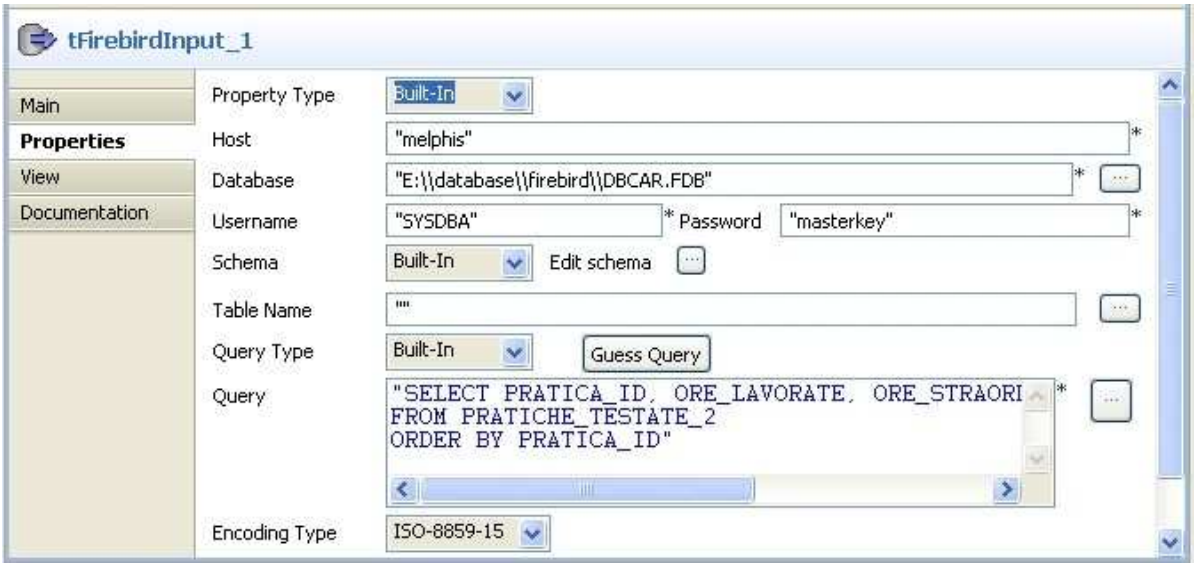


Illustrazione 66: Proprietà di RiparazioniPraticheMyCar

- **RiparazioniPratiche (tAccessInput)**

Per i dati delle pratiche si usa un componente tAccessInput con la seguente query:

```
"SELECT C.F_NUMPRA, (C.F_CODMAR & C.F_CODMOD & C.F_CODVER) AS COD_VEICOLO,
C.F_CODCLI, C.F_CODPER, C.F_DATACA, C.F_DTPRCO
FROM CARVEI C
ORDER BY C.F_NUMPRA"
```

- **tMap_1 (tMap)**

Il componente tMap è uno strumento molto importante all'interno delle applicazioni Open Studio. Con tMap si possono giungere sorgenti dati differenti, eseguire filtraggi, elaborare i campi utilizzando funzioni del linguaggio di programmazione o le librerie definite dall'utente. È importante nel nostro caso che il primo componente collegato a tMap_1 sia RiparazioniPratiche, per poi collegare tutti gli altri componenti visti finora. La situazione finale è quella della seguente figura:

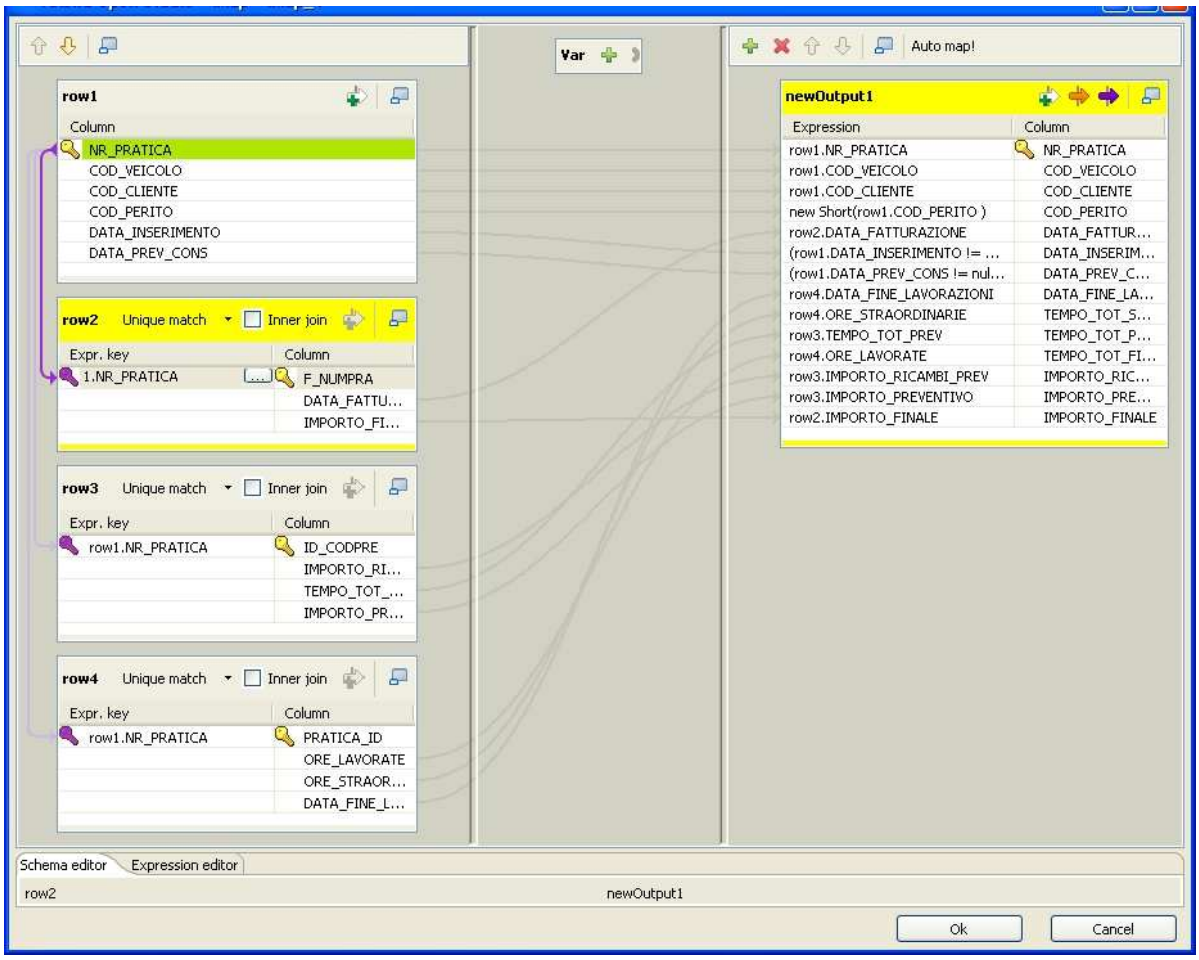


Illustrazione 67: Utilizzo del componente tMap

Nella figura si notano i join tra tabelle (vedi collegamenti nel riquadro di sinistra) e i rispettivi campi in uscita (collegamenti verso il riquadro di destra). Il campo COD_PERITO va convertito da stringa a short (intero a 16 bit) scrivendo nel campo Expression il codice per istanziare una classe Short di Java:

```
new Short(row1.COD_PERITO )
```

Possono essere usate espressioni anche complesse del linguaggio di programmazione, come nei campi DATA_INSERIMENTO e DATA_PREV_CONSEGNA calcolati rispettivamente con le stringhe:

```
(row1.DATA_INSERIMENTO != null && row1.DATA_INSERIMENTO.length() == 10) ?
new SimpleDateFormat("dd/mm/yyyy").parse(row1.DATA_INSERIMENTO) : null
```

e

```
(row1.DATA_PREV_CONS != null && row1.DATA_PREV_CONS.length() == 10) ? new
SimpleDateFormat("dd/mm/yyyy").parse(row1.DATA_PREV_CONS) : null
```

Con questi codici se la data in formato stringa non è nulla ed è lunga 10 caratteri allora verrà restituito un oggetto Date usando la classe SimpleDateFormat e il rispettivo metodo parse(), altrimenti verrà restituito il valore null. Le espressioni ovviamente dipendono dal linguaggio che si è scelto per il progetto, tra cui Perl o Java e le relative librerie.

- **Riparazioni (tFirebirdOutput)**

L'ultimo passaggio sta nell'aggiornare la tabella Riparazioni di Firebird con un componente tFirebirdOutput. Nella seguente figura si nota come il campo “Action on data” sia impostato su “Update or Insert”, consentendo di aggiornare i valori qualora siano già presenti, oppure di inserirli in caso contrario.

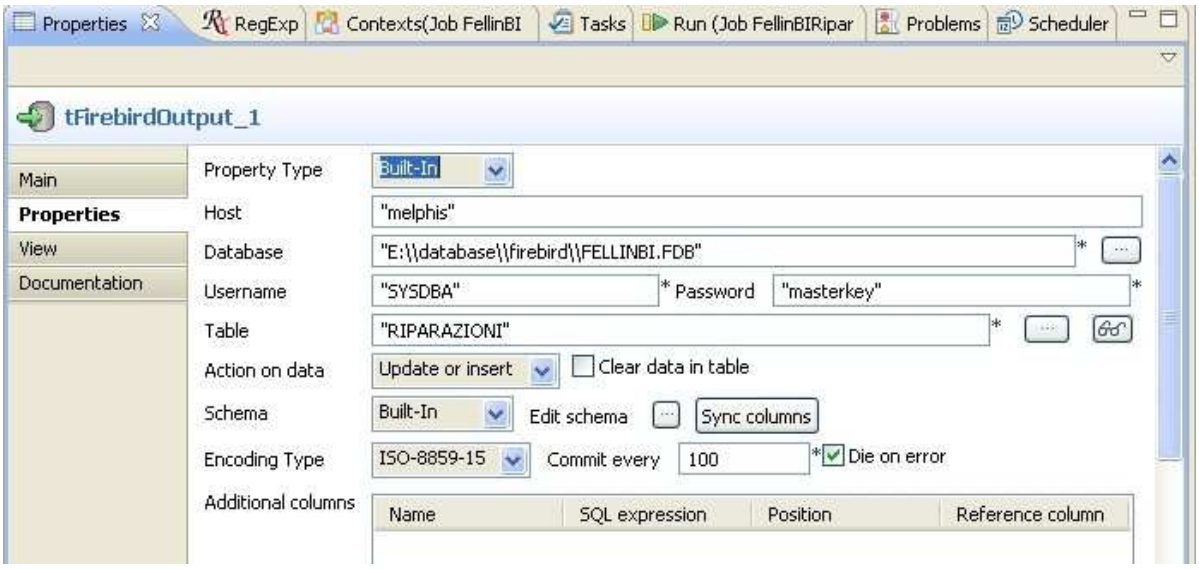


Illustrazione 68: Proprietà del componente Riparazioni

In ogni momento è possibile mandare in esecuzione la procedura e valutare i risultati con il debugger integrato.

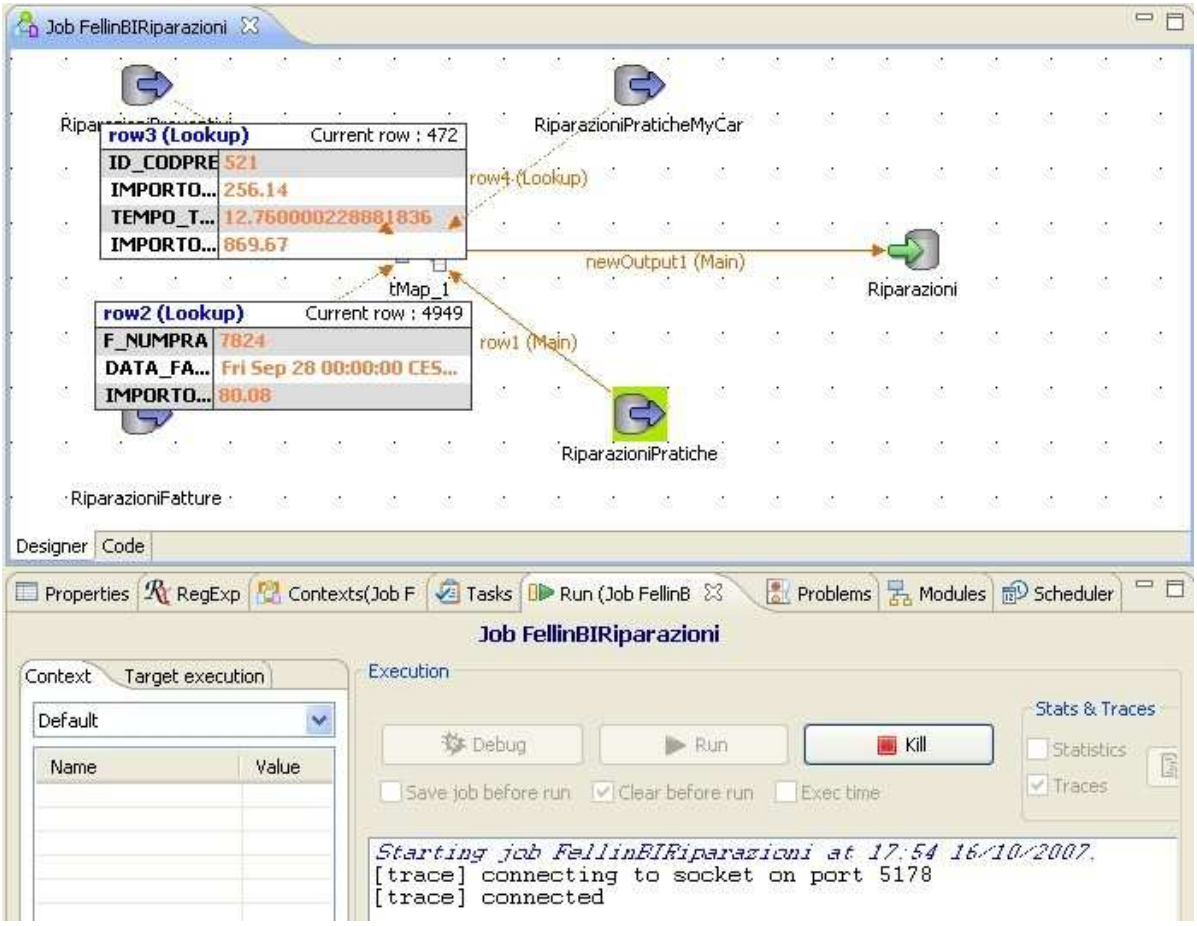


Illustrazione 69: Debugging della procedura di ETL

5.2.2.2 Dimensione Data Fatturazione

La procedura è concettualmente simile a quelle adottata con Kettle:

Si generano 30000 record (circa 80 anni) e per ogni ennesimo record si sommano n giorni a partire dalla data 01 Gennaio 1970. Dalla data così ottenuta vengono estratte le informazioni richieste come giorno, mese, anno, eccetera. A differenza di Kettle, dove venivano usati dei componenti specifici per l'elaborazione, in Open Studio si è preferito sfruttare le capacità di programmazione dell'ambiente. A un componente tRowGenerator per la generazione dei record viene collegato un componente tJavaRow che ne elabora l'output e manda il risultato al componente tFirebirdOutput per l'aggiornamento della base dati. Il tutto è visibile nella seguente

figura:

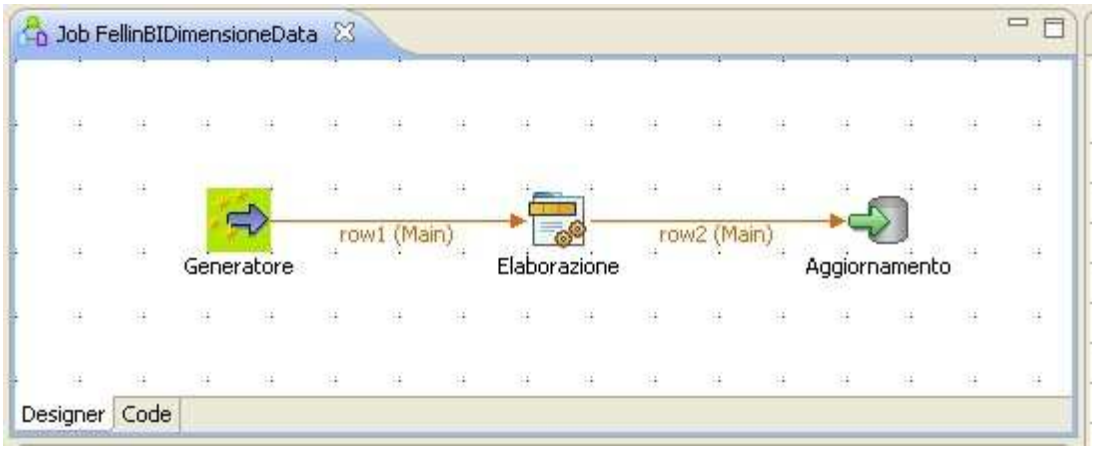


Illustrazione 70: Procedura di aggiornamento della tabella Dimensione_data

- Generatore (tRowGenerator)

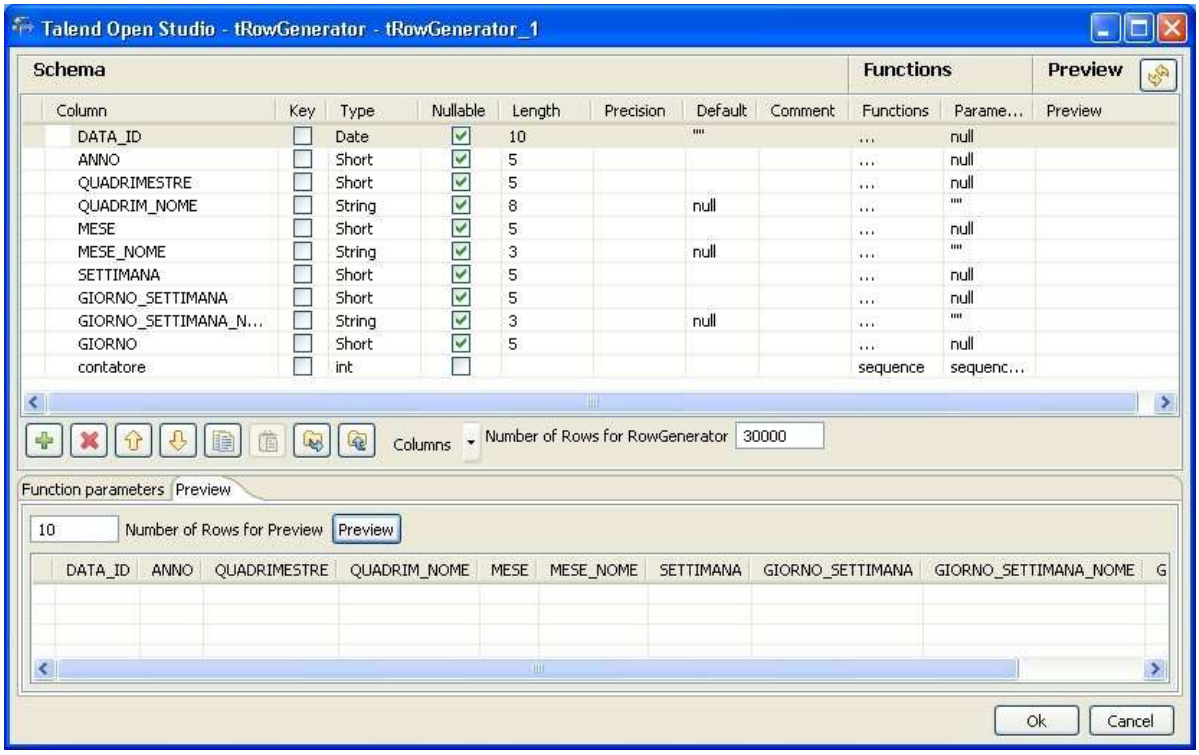


Illustrazione 71: Configurazione del generatore dei record

Con un doppio click sul componente tRowGenerator si vede lo schema delle righe generate, tutte con valori iniziali nulli (vedi colonna “Parameters” della Illustrazione 71).

- **Elaborazione (tJavaRow)**

Il componente tJavaRow permette di definire un pezzo di codice (in Java o Perl a seconda del progetto) che viene eseguito riga per riga consentendo ad esempio di modificare i valori del record.

Nel nostro caso il codice aggiunge alla data 01 Gennaio 1970 n giorni dove n è il numero di record generati fino a quel momento. La data viene poi elaborata per completare le informazioni della tabella Dimensione_data usando il linguaggio di programmazione Java. Nella codifica si consideri che in Java dalla versione 1.1 l'utilizzo dei metodi dell'oggetto Date quali getDays(), getMonth(), eccetera sono considerati obsoleti. Date rimane il tipo di dati predefinito per le date ma per agevolare l'internazionalizzazione[11] delle applicazioni è stata introdotta la classe Calendar. Trattandosi di funzioni non di facile utilizzo si è preferito creare una nuova libreria in Open Studio definendo la classe wrapper DateUtils, che potrà tornare utile nelle evoluzioni future. La libreria e il codice in essa contenuto sono riportati nelle seguenti illustrazioni:

11 Per internazionalizzazione si intende la preparazione delle applicazioni a mercati intenrnazionali. Con l'internazionalizzazione l'applicazione supporta numeri e date in formati che dipendono dalla cultura, dai calendari, eccetera.

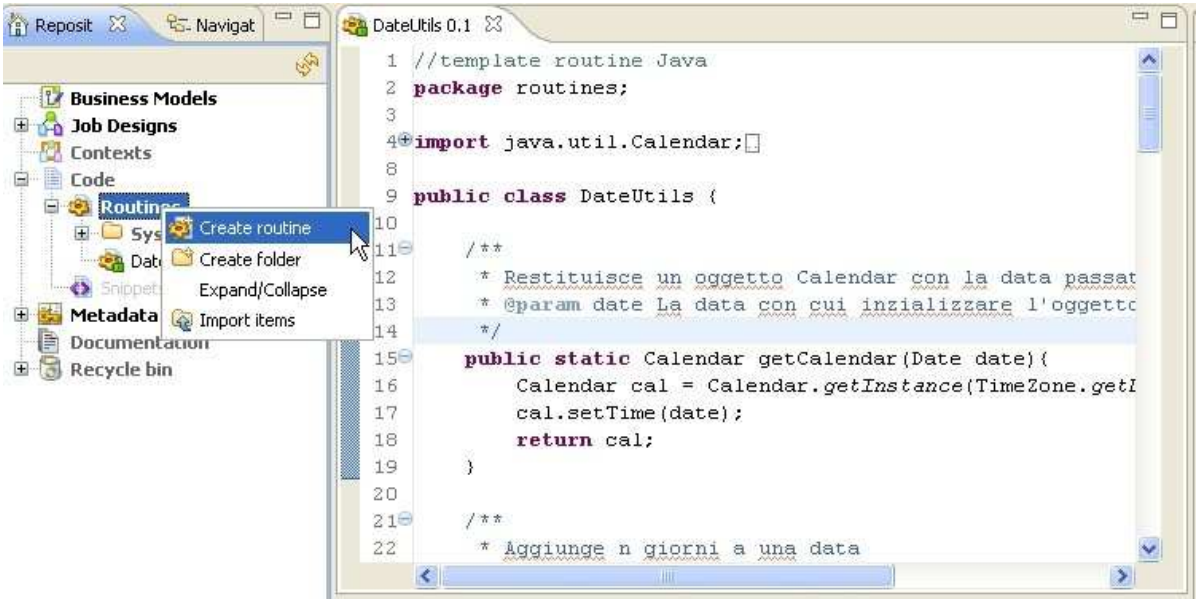


Illustrazione 72: Definizione della libreria DateUtils

```
//template routine Java
package routines;

import java.util.Calendar;
import java.util.Locale;
import java.util.TimeZone;
import java.util.Date;

public class DateUtils {

/**
 * Restituisce un oggetto Calendar con la data passata come parametro
 * @param date La data con cui inizializzare l'oggetto Calendar
 */
public static Calendar getCalendar(Date date){
 Calendar cal = Calendar.getInstance(TimeZone.getDefault(),
Locale.ITALIAN);
 cal.setTime(date);
 return cal;
}

/**
 * Aggiunge n giorni a una data
 * @param cal l'oggetto Calendar con la data da elaborare
```

```
 * @param days il numero di giorni da aggiungere, positivi o negativi
 */
public static void addDays(Calendar cal, int days){
      cal.add(Calendar.DAY_OF_MONTH, days);
}

/**
 * Restituisce l'anno della data fornita
 * @return l'anno con tutte le cifre significative (es. 2000)
 */
public static short getYear(Calendar cal){
      return (short) cal.get(java.util.Calendar.YEAR);
}

/**
 * Restituisce il quadrimestre a cui appartiene il mese
 * @param month il numero del mese (1 gennaio, 12 dicembre)
 * @return il numero del quadrimestre
 */
public static short getQuarter(Calendar cal){
      int val = (cal.get(java.util.Calendar.MONTH)+1);
      // se il mese val non e' divisibile per tre restituisce (val/3)+1,
      // altrimenti restituisce val/3
 return ((val%3) != 0) ? ((val/3)+1) : (val/3);
}

/**
 * Restituisce una rappresentazione in numero romano del quadrimestre
 * @param month il numero del quadrimestre
 * @return il nome del quadrimestre nel formato "<numero romano> Quad"
 */
public static String getQuarterName(Calendar cal){
      String quarterNames[] = {"I Quad", "II Quad",
                               "III Quad", "IV Quad"};
      return quarterNames[getQuarter(cal)-1];
}

/**
 * Restituisce il numero del mese
 * @return il numero del mese, 1 = gennaio e 12 dicembre
 */
```

```
public static short getMonth(Calendar cal){
      return (short) (cal.get(java.util.Calendar.MONTH)+1);
}

/**
 * Restituisce il nome del mese
 * @return il nome del mese, prime 3 lettere maiuscole in italiano
 */
public static String getMonthName(Calendar cal){
      String monthNames[] = {"GEN", "FEB", "MAR",
                             "APR", "MAG", "GIU",
                             "LUG", "AGO", "SET",
                             "OTT", "NOV", "DIC"};
      return monthNames[cal.get(java.util.Calendar.MONTH)];
}

/**
 * Restituisce il numero della settimana nell'anno
 * @return il numero della settimana dell'anno, a partire dal numero 1
 */
public static short getWeek(Calendar cal){
      return (short) cal.get(java.util.Calendar.WEEK_OF_YEAR);
}

/**
 * Restituisce il giorno della settimana in formato numerico
 * @return il giorno della settimana: 1=dom, 2=lun, 7=sab
 */
public static short getDayOfWeek(Calendar cal){
      return (short) (cal.get(java.util.Calendar.DAY_OF_WEEK));
}

/**
 * Restituisce il nome del giorno della settimana
 * @return il nome del giorno della settimana, prime 3 lettere in
maiuscolo in italiano
 */
public static String getDayOfWeekName(Calendar cal){
      String names[] = {"DOM","LUN","MAR","MER",
                        "GIO","VEN","SAB"};
      return names[cal.get(java.util.Calendar.DAY_OF_WEEK)-1];
```

```
}


/**
 * Restituisce numero del giorno nel mese
 * @return il numero del giorno nel mese, da 1 a 31
 */
public static short getDay(Calendar cal){
      return (short) cal.get(java.util.Calendar.DAY_OF_MONTH);
}
}
```

Definita la routine DateUtils è possibile richiamarla all'interno del componente tJavaRow, come mostra la seguente figura:

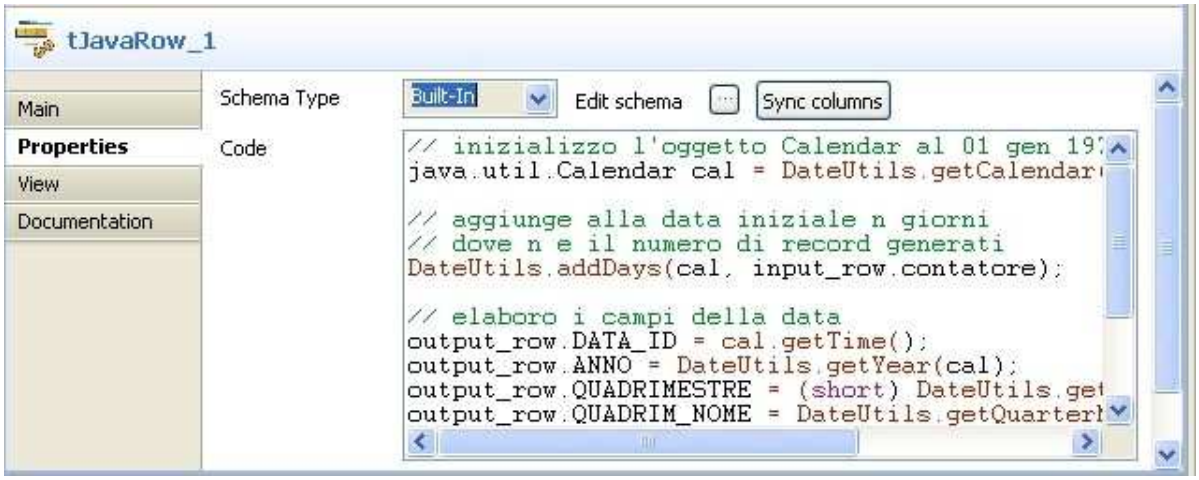


Illustrazione 73: Proprietà del componente Elaborazione

Il codice per motivi di chiarezza viene riportato qui di seguito:

```
// inizializzo l'oggetto Calendar al 01 gen 1970
java.util.Calendar cal = DateUtils.getCalendar(new java.util.Date(0));

// aggiunge alla data iniziale n giorni
// dove n e il numero di record generati
DateUtils.addDays(cal, input_row.contatore);

// elaboro i campi della data
output_row.DATA_ID = cal.getTime();
output_row.ANNO = DateUtils.getYear(cal);
```

```
output_row.QUADRIMESTRE = (short) DateUtils.getQuarter(cal);
output_row.QUADRIM_NOME = DateUtils.getQuarterName(cal);
output_row.MESE = DateUtils.getMonth(cal);
output_row.MESE_NOME = DateUtils.getMonthName(cal);
output_row.SETTIMANA = DateUtils.getWeek(cal);
output_row.GIORNO_SETTIMANA = DateUtils.getDayOfWeek(cal);
output_row.GIORNO_SETTIMANA_NOME = DateUtils.getDayOfWeekName(cal);
output_row.GIORNO = DateUtils.getDay(cal);
```

- **Aggiornamento (tFirebirdOutput)**

L'ultimo passaggio è quello di aggiornare la tabella Dimensione_data di Firebird. Come si vede nella successiva figura si è scelta l'opzione “Insert” come “Action on data” e si è spuntato il flag “Clear data in table”, così da cancellare tutti i record presenti e inserire quelli generati con istruzioni INSERT. La scelta è condizionata dal fatto che l'aggiornamento della dimensione data avviene una tantum a differenza delle altre tabelle del database.

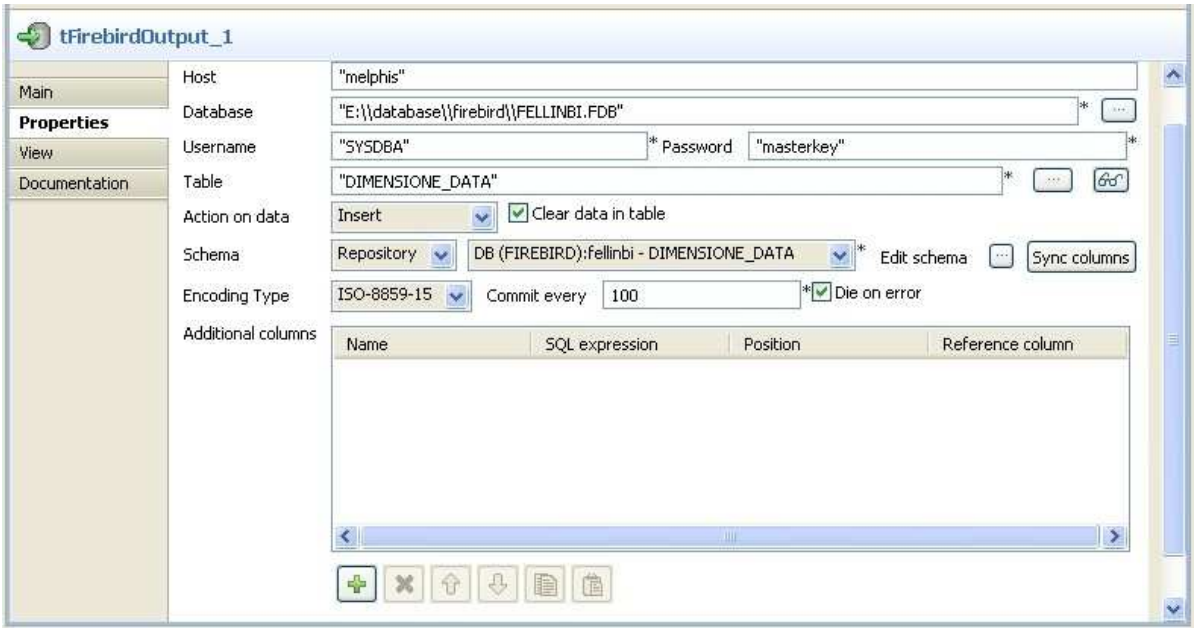


Illustrazione 74: Proprietà del componente Aggiornamento

5.2.2.3 Dimensione Clienti

L'aggiornamento delle rimanenti tabelle è relativamente più semplice. Per l'anagrafica clienti i dati vengono prelevati dal database WinCar con il componente tAccessInput di Illustrazione 75.

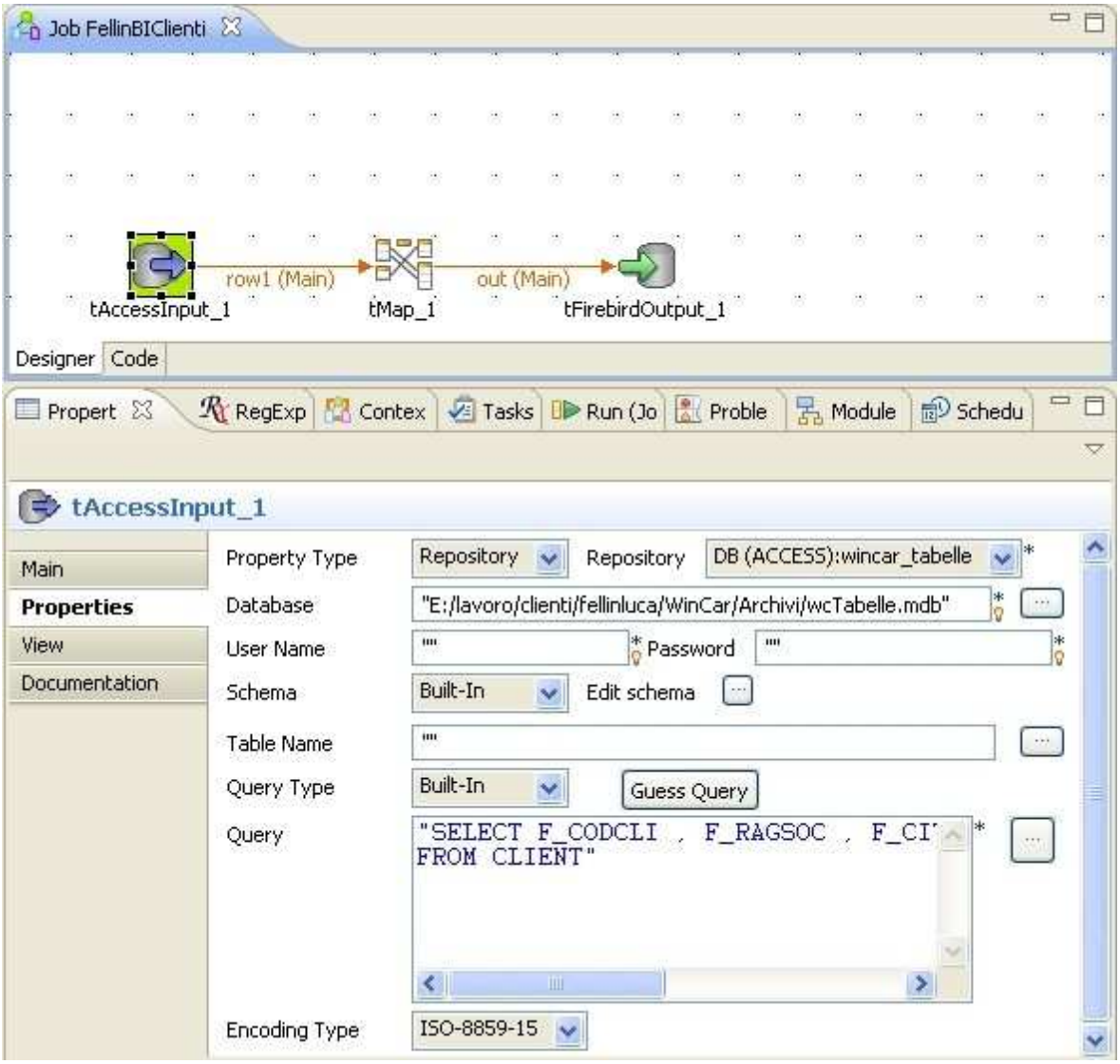


Illustrazione 75: Aggiornamento tabella clienti

La query usata è la seguente:

```
"SELECT F_CODCLI , F_RAGSOC , F_CITTAC , F_CAPCLI , F_PROCLI , F_PRIVATO
FROM CLIENT"
```

È poi necessario convertire il campo PRIVATO di CLIENT dal tipo booleano True/False al tipo stringa “Y”/”N”. Per tale motivo il flusso

dei dati da tAccessInput viene mandato al componente tMap di Illustrazione 76, sfruttando la possibilità di dichiarare variabili intermedie e la seguente stringa di codice:

```
(row1.PRIVATO ) ? "Y" : "N"
```

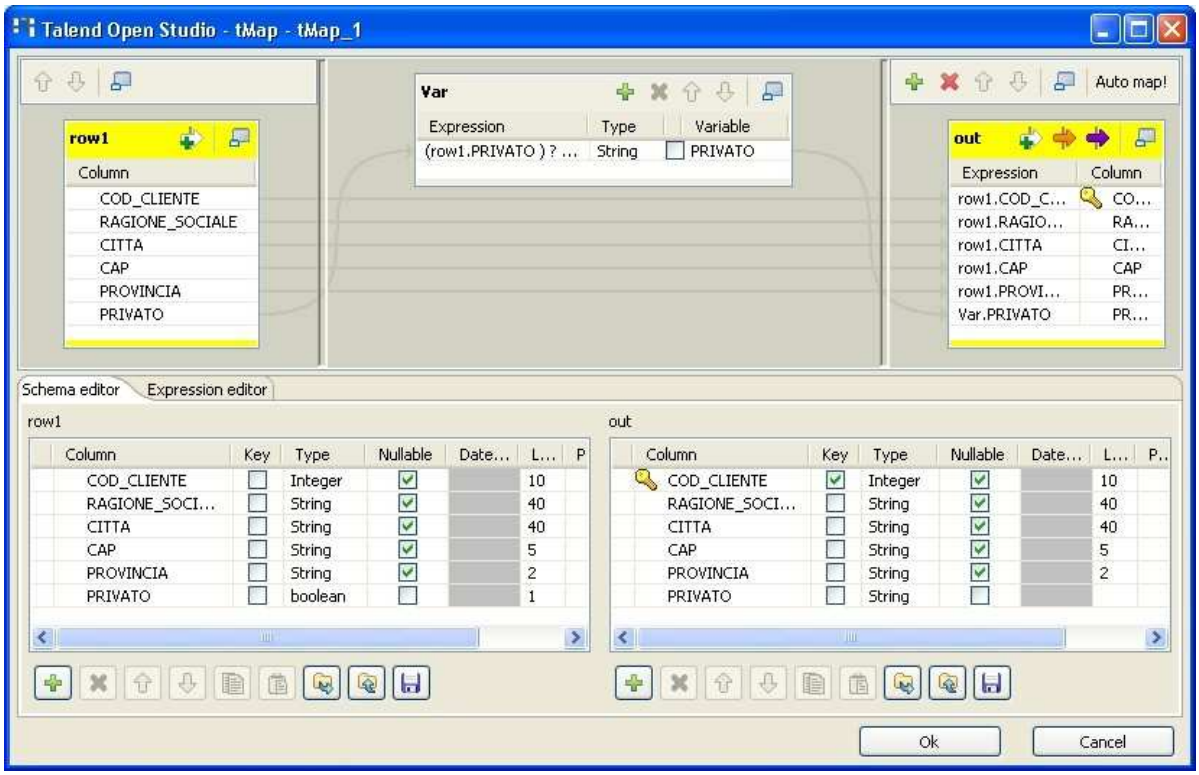


Illustrazione 76: Conversione del campo PRIVATO di CLIENT

L'ultimo passaggio sta nell'aggiornare la base dati, usando un componente tFirebirdInput configurato come segue:

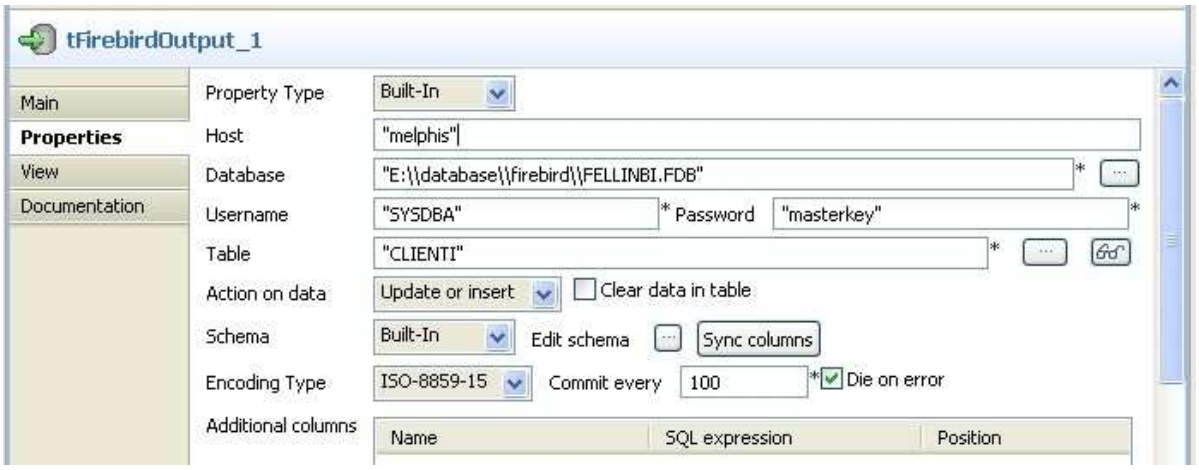


Illustrazione 77: Componente per l'aggiornamento della tabella Clienti

5.2.2.4 Dimensione Veicoli

L'aggiornamento della tabella Veicoli viene realizzato collegando assieme due componenti. Il primo è un tAccessInput che punti al database wcArchivi di WinCar con la seguente query:

```
"SELECT (F_CODMAR & F_CODMOD & F_CODVER) as COD_VEICOLO,
UCASE(LAST(F_DESMAR)) AS DESCR_MARCA, UCASE(LAST(F_DESMOD)) AS
DESCR_MODELLO, UCASE(LAST(F_DESVER)) AS DESCR_VERSIONE
FROM CARVEI
WHERE F_DESMAR <> '' and F_DESMOD <> ''
GROUP BY (F_CODMAR & F_CODMOD & F_CODVER)"
```

Il componente successivo è un tFirebirdOutput, configurato come nella figura:

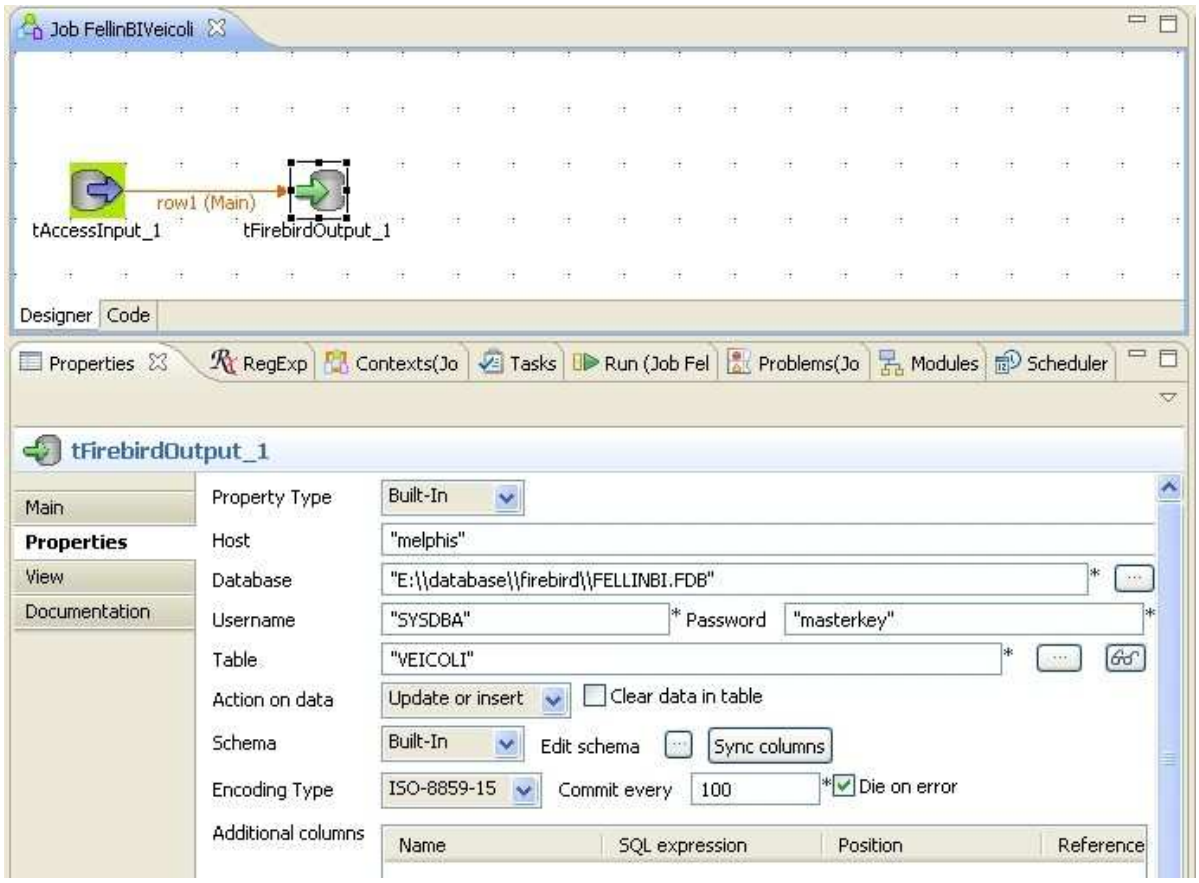


Illustrazione 78: Componente per l'aggiornamento della tabella Veicoli

5.2.2.5 Dimensione Perizie

Per aggiornare la tabella Periti è sufficiente usare un componente tAccessInput e collegarlo a un componente tFirebirdOutput, il tutto come in Illustrazione 79. La query di tAccessInput, che viene impostato per puntare al database wcTabelle di Wincar, è la seguente:

```
"select F_CODPER, F_NOMPER from PERITI"
```

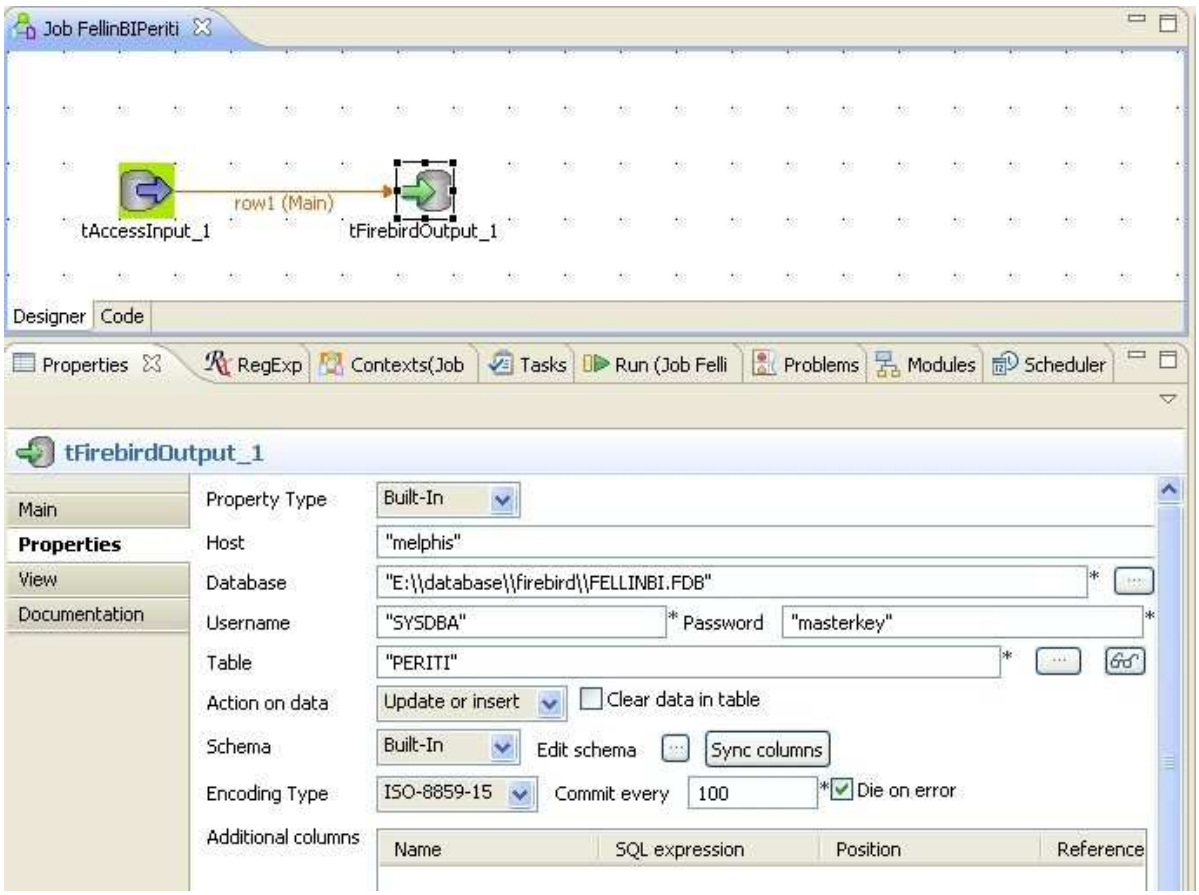


Illustrazione 79: Aggiornamento dimensione Periti

5.2.2.6 Integriamo il tutto

Le procedure ETL fin qui esaminate vengono eseguite singolarmente una dopo l'altra così da aggiornare l'intera base dati. Per agevolare l'operazione è possibile creare un progetto job che le raggruppi e le esegua in senso sequenziale. Nella Illustrazione 80 si

vede come i jobs vengano incorporati usando componenti tRunJob e collegamenti di tipo Then Run. L'aggiornamento di Dimensione data rimane invece escluso dal job ed eseguito singolarmente una tantum.

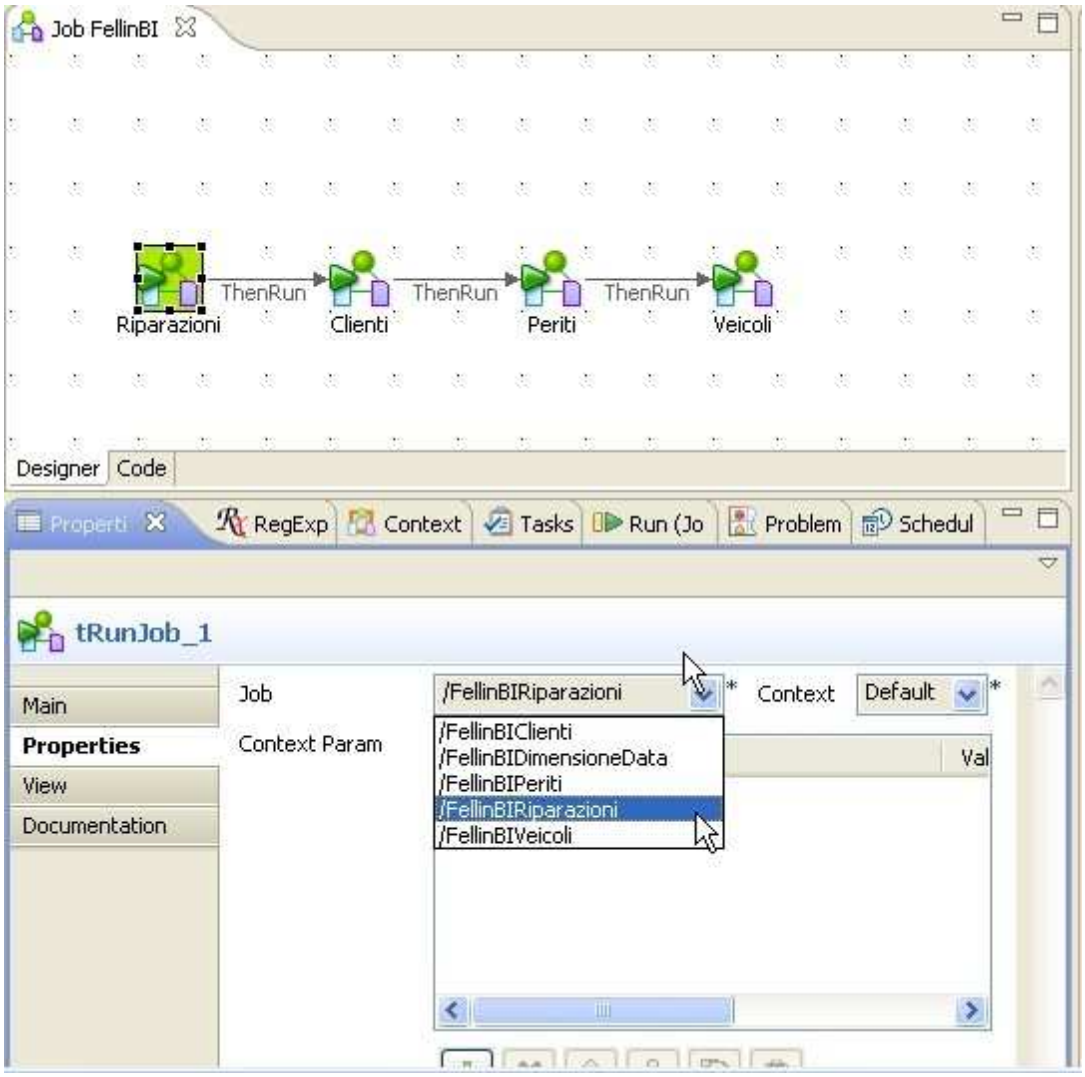


Illustrazione 80: Procedura di aggiornamento FellinBI

Finora i progetti sono stati mandati in esecuzione all'interno dell'ambiente Open Studio. Nelle implementazioni pratiche i singoli job vengono compilati ed esportati in un archivio per poi essere mandati in esecuzione. Come si vede in Illustrazione 81 l'operazione avviene con una mascherina che appare cliccando con il pulsante destro del mouse sul job scelto e quindi sulla voce “Export Job

Scripts”. Nell'archivio viene raccolto tutto il codice Java o Perl necessario per mandare in esecuzione la procedure ETL senza disporre dell'ambiente Open Studio.

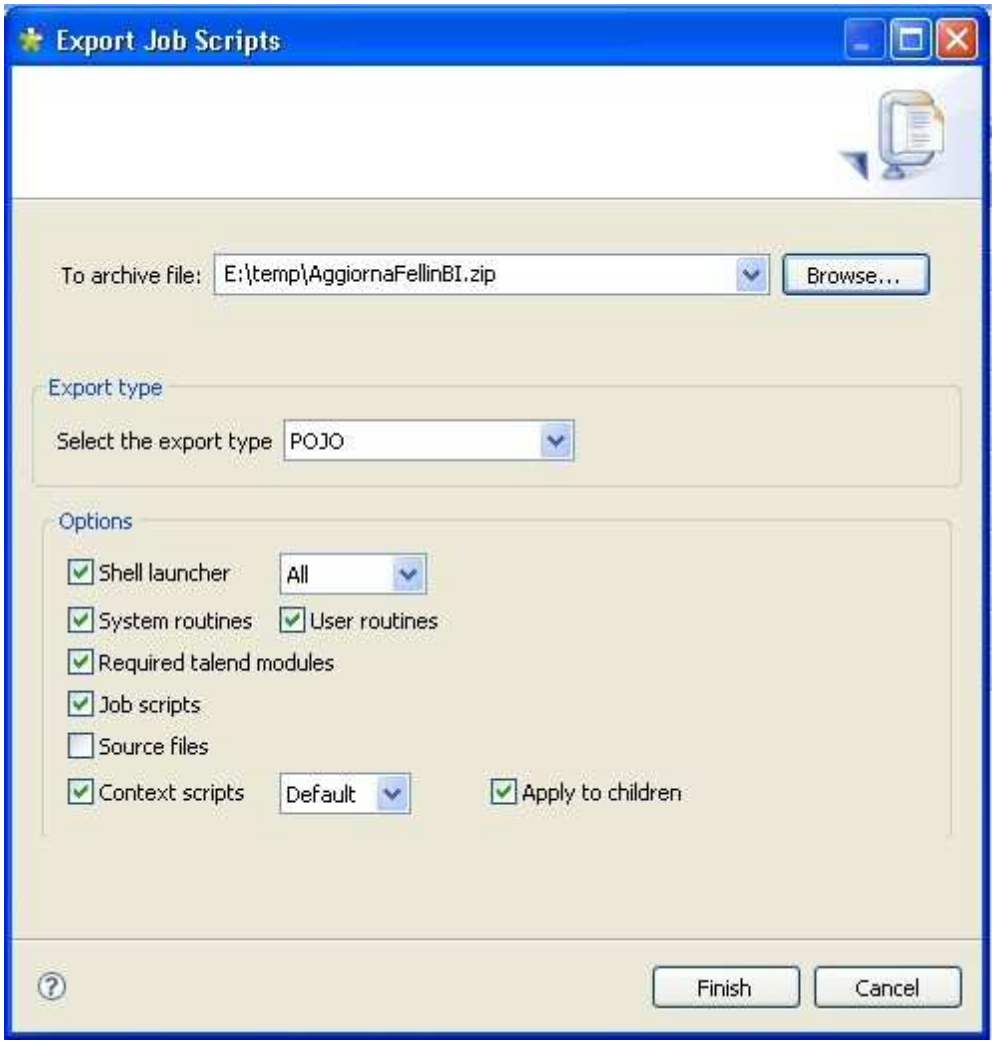


Illustrazione 81: Compilazione del job in un archivio

Rispetto a Kettle questa è una caratteristica molto importante in fase di distribuzione delle applicazioni, perché richiede la sola presenza dell'interprete Java o Perl per l'esecuzione delle procedure ETL.

Una ulteriore funzionalità molto importante è lo scheduling degli aggiornamenti, che Talend Open Studio consente di definire con dettaglio all'interno della scheda Scheduler. In fase di esportazione

del job verrà così generato uno script che configura il funzionamento del servizio cron, disponibile in quasi tutte le varianti UNIX e su Windows installando uno dei tanti port come CRONw (http://cronw.sourceforge.net).

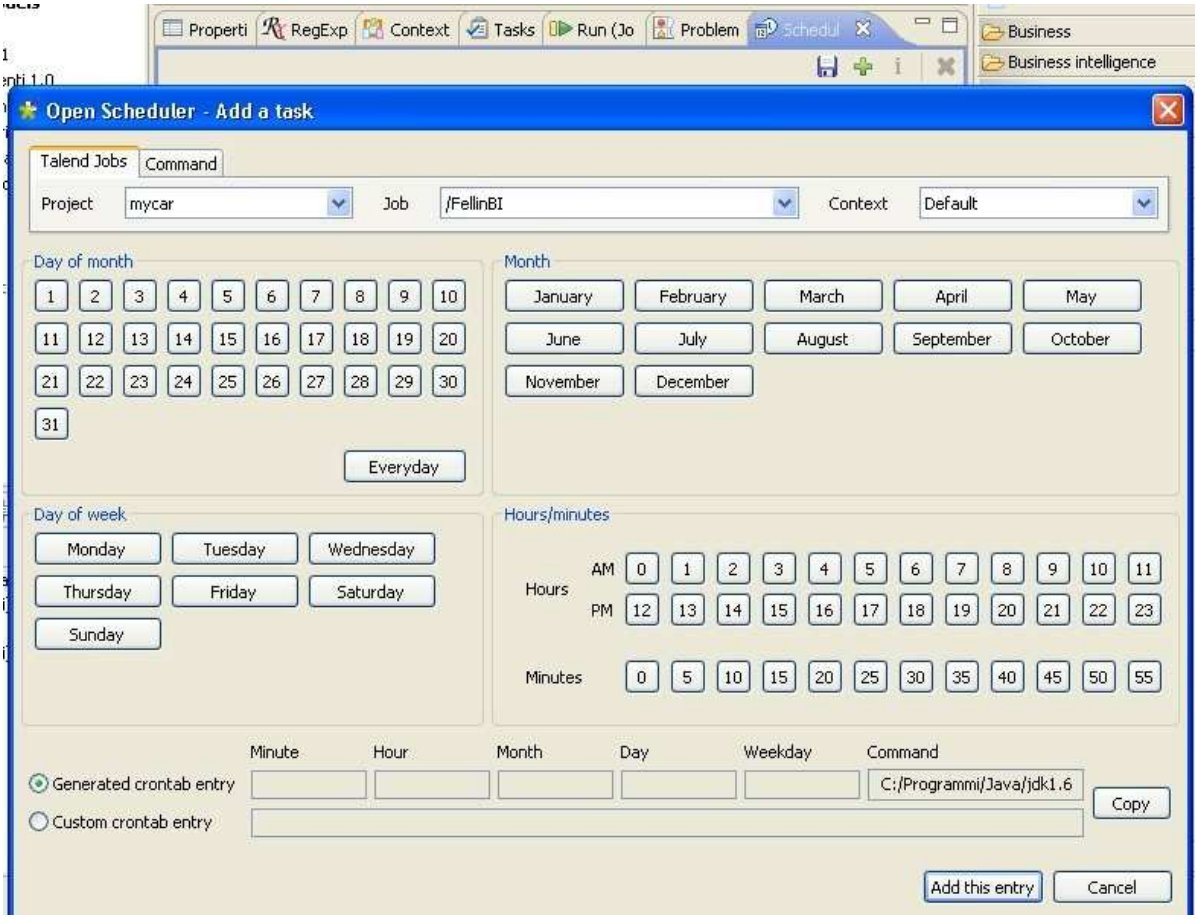


Illustrazione 82: Scheduling dell'esecuzione del job

5.3 Schema OLAP con Mondrian

La tabella RIPARAZIONI del paragrafo 4.4 nasce per ospitare i dati del cubo multidimensionale Riparazioni. I fatti elaborabili con gli strumenti di analisi OLAP sono:

- Totale tempo riparazione preventivato in ore (colonna TEMPO_TOT_PREV)
- Totale tempo effettivo in ore (TEMPO_TOT_FINALE)

- Totale tempo effettivo straordinario in ore (TEMPO_TOT_STRAORD)
- Importo ricambi riparazione preventivato (IMPORTO_RICAMBI_PREV)
- Importo riparazione preventivato (IMPORTO_PREVENTIVO)
- Importo effettivo (IMPORTO_FINALE)
- Differenza data consegna prevista e effettiva in giorni (positiva se veicolo consegnato in ritardo) (DELTA_GG_DAL_PREVENTIVO)
- Differenza prezzo preventivato e effettivo in giorni (positiva se il prezzo risulta maggiore del preventivo) (DELTA_PREZZO_DAL_PREVENTIVO)
- Differenza ore preventivate e effettive (positiva se il tempo impiegato è maggiore del preventivato) (DELTA_ORE_DAL_PREVENTIVO)

Per ogni fatto poi sono poi disponibili le seguenti dimensioni:

dimensione	contenuto	gerarchie

Clienti (tabella CLIENTI)	I clienti dell'auto sottoposta a riparazione	• Ragione Sociale
		  • Città
		  • CAP
		  • Provincia
		• Privato

Data fatturazione (tabella Dimensione_data)	La data in cui la riparazione è stata fatturata	• Anno
		  • Quadrimestre
		  • Nome Quadrimestre
		  • Mese
		  • Nome Mese

		• Settimana • Giorno settimana • Nome giorno settimana • Giorno
Periti (tabella PERITI)	I periti incaricati delle rilevazioni	• Nome perito
Veicoli (tabella VEICOLI)	Descrizione dell'auto riparata	• Marca • Modello • Versione

In un ottica di evoluzione futura durante lo sviluppo del DataWarehouse si è preferito lasciare nelle gerarchie un numero maggiore di informazioni rispetto a quelle effettivamente richieste, come ad esempio il campo CAP di Clienti o il campo Nome giorno della settimana in Dimensione Data. Inoltre i fatti prima elencati per il cubo Riparazione non verranno utilizzati tutti nell'analisi OLAP. Sarà il feedback in azienda a definire con precisione quali sono le informazioni veramente necessarie da riportare nel cubo Riparazioni, cosa che sicuramente si chiarirà nell'utilizzo della piattaforma da parte dei soggetti interessati in un'ottica evolutiva.

Quello che è necessario ora è lo schema Mondrian per la mappatura della base dati con il cubo multidimensionale OLAP. Definito il file schema questo verrà registrato all'interno della piattaforma e verrà creato un documento analitico che esegui l'interrogazione del cubo Riparazioni.

Per realizzare lo schema è stato utilizzato il software Mondrian Schema Workbench, ma anche Pentaho Cube Designer permette di crearne una bozza poi rifinita a mano. Una buona conoscenza della

struttura del file schema in XML è in ogni caso doverosa per utilizzare con efficacia gli strumenti messi a disposizione. Di seguito è riportato il file tracciato per il cubo Riparazioni:

```
<Schema name="FellinBI">
<Cube name="Riparazioni" cache="true" enabled="true">
      <Table name="RIPARAZIONI">
      </Table>
      <Dimension type="TimeDimension" foreignKey="DATA_FATTURAZIONE"
                  name="DATA_FATTURAZIONE">
            <Hierarchy hasAll="true" allMemberName="All Periods"
                        primaryKey="DATA_ID">
                  <Table name="DIMENSIONE_DATA"></Table>
                  <Level name="ANNO" column="ANNO" type="Numeric"
                              uniqueMembers="true" levelType="TimeYears"
                                    hideMemberIf="Never"></Level>
                  <Level name="QUADRIMESTRE" column="QUADRIMESTRE"
                              nameColumn="QUADRIM_NOME"
ordinalColumn="QUADRIMESTRE"                  type="String"
uniqueMembers="false"
      levelType="TimeQuarters" hideMemberIf="Never"></Level>
                  <Level name="MESE" column="MESE" nameColumn="MESE_NOME"
                              ordinalColumn="MESE" type="Numeric"
                                    uniqueMembers="false"
levelType="TimeMonths"
      hideMemberIf="Never"></Level>
                  <Level name="GIORNO" column="GIORNO"
ordinalColumn="GIORNO"                  type="Numeric"
uniqueMembers="false"
      levelType="TimeDays" hideMemberIf="Never"></Level>
            </Hierarchy>
      </Dimension>
      <Dimension foreignKey="COD_CLIENTE" name="CLIENTI">
            <Hierarchy name="CLIENTI" hasAll="true" allMemberName="All
clienti"               primaryKey="COD_CLIENTE">
                  <Table name="CLIENTI"></Table>
                  <Level name="CLIENTI.PROVINCIA" table="CLIENTI"
                              column="PROVINCIA" type="String"
uniqueMembers="false"                  levelType="Regular"
hideMemberIf="Never">  </Level>
```

```
                <Level name="CLIENTI.CITTA" table="CLIENTI"
column="CITTA"                          type="String"
uniqueMembers="false" levelType="Regular"
      hideMemberIf="Never"></Level>
                <Level name="CLIENTI.RAGIONE_SOCIALE" table="CLIENTI"
                           column="RAGIONE_SOCIALE" type="String"
                                 uniqueMembers="false"
levelType="Regular"
      hideMemberIf="Never"></Level>
            </Hierarchy>
            <Hierarchy name="CLIENTI_COMPLETO" hasAll="true"
allMemberName="All                clienti" primaryKey="COD_CLIENTE">
                <Table name="CLIENTI"></Table>
                <Level name="CLIENTI.PRIVATO" table="CLIENTI"
column="PRIVATO"                  type="String" uniqueMembers="false"
levelType="Regular"                     hideMemberIf="Never"></Level>
                <Level name="CLIENTI.PROVINCIA" table="CLIENTI"
                           column="PROVINCIA" type="String"
uniqueMembers="false"                   levelType="Regular"
hideMemberIf="Never">   </Level>
                <Level name="CLIENTI.CITTA" table="CLIENTI"
column="CITTA"                          type="String"
uniqueMembers="false" levelType="Regular"
      hideMemberIf="Never"></Level>
                <Level name="CLIENTI.RAGIONE_SOCIALE" table="CLIENTI"
                           column="RAGIONE_SOCIALE" type="String"
                                 uniqueMembers="false"
levelType="Regular"
      hideMemberIf="Never"></Level>
            </Hierarchy>
      </Dimension>
      <Dimension foreignKey="COD_VEICOLO" name="VEICOLI">
            <Hierarchy name="VEICOLI" hasAll="true" allMemberName="All
veicoli"               primaryKey="COD_VEICOLO">
                <Table name="VEICOLI"></Table>
                <Level name="VEICOLI.DESCR_MARCA" table="VEICOLI"
                           column="DESCR_MARCA" type="String"
uniqueMembers="false"                   levelType="Regular"
hideMemberIf="Never"></Level>
                <Level name="VEICOLI.DESCR_MODELLO" table="VEICOLI"
                           column="DESCR_MODELLO" type="String"
```

```
                                    uniqueMembers="false"
levelType="Regular"
      hideMemberIf="Never"></Level>
                  <Level name="VEICOLI.DESCR_VERSIONE" table="VEICOLI"
                              column="DESCR_VERSIONE" type="String"
                                    uniqueMembers="false"
levelType="Regular"
      hideMemberIf="Never"></Level>
            </Hierarchy>
      </Dimension>
      <Dimension foreignKey="COD_PERITO" name="PERIZIE">
            <Hierarchy name="PERIZIE" hasAll="true" allMemberName="All
                        perizie" primaryKey="COD_PERITO">
                  <Table name="PERITI"></Table>
                  <Level name="PERIZIE.NOME_PERITO" table="PERITI"
                              column="NOME_PERITO" type="String"
uniqueMembers="false"                     levelType="Regular"
hideMemberIf="Never">  </Level>
            </Hierarchy>
      </Dimension>
      <Measure name="Importo finale" column="IMPORTO_FINALE"
datatype="Numeric"           formatString="#,##0.00"
aggregator="sum"></Measure>
      <Measure name="Importo preventivato" column="IMPORTO_PREVENTIVO"
                  datatype="Numeric" formatString="#,##0.00"
                        aggregator="sum"></Measure>
      <Measure name="Importo preventivato ricambi"
column="IMPORTO_RICAMBI_PREV"            datatype="Numeric"
formatString="#,##0.00"
      aggregator="sum"></Measure>
      <Measure name="Nr pratiche lavorate" column="NR_PRATICA"
                  datatype="Integer" formatString="#,##0"
                        aggregator="count"></Measure>
      <Measure name="Tempo totale lavorazioni" column="TEMPO_TOT_FINALE"
                  datatype="Numeric" formatString="#,##0.00"
                        aggregator="sum"></Measure>
      <Measure name="Tempo totale preventivato" column="TEMPO_TOT_PREV"
                  datatype="Numeric" formatString="#,##0.00"
                        aggregator="sum"></Measure>
      <Measure name="Tempo straordinario lavorazioni"
column="TEMPO_TOT_STRAORD"         datatype="Numeric"
```

```
formatString="#,##0.00"
      aggregator="sum"></Measure>
      <Measure name="Differenza giorni dal preventivo"
                  column="DELTA_GG_DAL_PREVENTIVO" datatype="Numeric"
formatString="#"       aggregator="sum" visible="true"></Measure>
      <Measure name="Differenza prezzo dal preventivo"
                  column="DELTA_PREZZO_DAL_PREVENTIVO" datatype="Numeric"
                        formatString="#,##0.00" aggregator="sum"
visible="true"></Measure>    <Measure name="Differenza ore dal
preventivo"
      column="DELTA_ORE_DAL_PREVENTIVO" formatString="#,#0"
            aggregator="sum" caption="" visible="true"></Measure>
</Cube>
</Schema>
```

Tale file verrà registrato nelle rispettive piattaforme come descritto nei paragrafi 5.5.4 e 5.6.4.

5.4 Reports

Fra i software di reportistica a disposizione per il progetto (BIRT, JasperReport e per PentahoBI anche JFreeReport) la scelta è caduta su JasperReport per il livello di funzionalità, stabilità e semplicità d'uso che lo contraddistinguono. Per completezza è stato fatto un collaudo anche di BIRT e JFreeReport come prova dell'effettiva integrazione con le due piattaforme.

5.4.1 JasperReport

JasperReport è una libreria Java che legge e interpreta un file XML con estensione .jrxml e produce in uscita un report in vari formati come PDF, HTML, XML, CVS, XLS. La realizzazione di un report diventa molto intuitiva utilizzando il programma di design iReport.

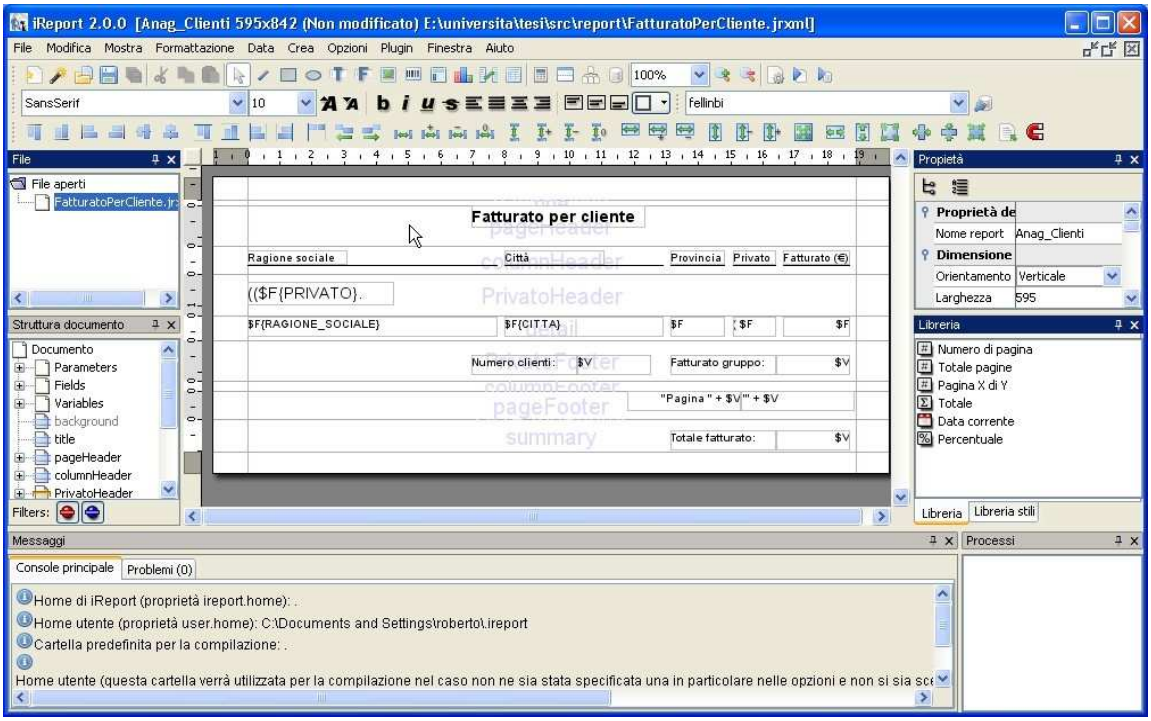


Illustrazione 83: Design del report Fatturato per cliente in iReport

Il primo passaggio consiste nella definizione della connessione al database e della query per l'estrazione dei dati.

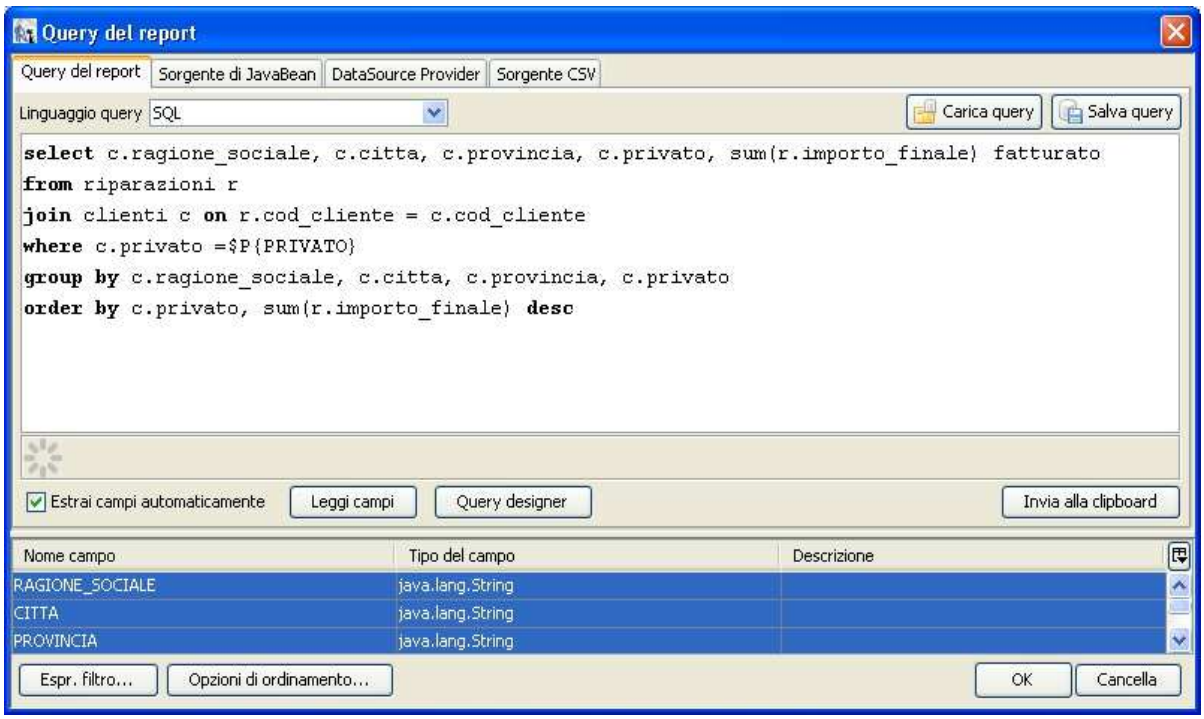


Illustrazione 84: Definizione della query in iReport

E' previsto l'uso di parametri all'interno della stringa SQL nella forma $P{PARAMETRO}. Questi parametri, che compariranno nell'apposito elenco “Parameters” di iReport, vengono assegnati durante l'esecuzione e consentono di filtrare e modificare il comportamento finale. Nel nostro progetto è stato aggiunto il parametro $P{PRIVATO} all'interno della seguente query:

```
select c.ragione_sociale, c.citta, c.provincia, c.privato,
sum(r.importo_finale) fatturato
from riparazioni r
join clienti c on r.cod_cliente = c.cod_cliente
where c.privato =$P{PRIVATO}
group by c.ragione_sociale, c.citta, c.provincia, c.privato
order by c.privato, sum(r.importo_finale) desc
```

Contraddistinguono le successive operazioni di design la logica completamente a bande, dove vengono posizionati e quindi stampati i vari componenti come campi di database, etichette e immagini.

Il raggruppamento dei dati, anche a livelli multipli, è possibile utilizzando lo strumento Mostra->Gruppi. Aggiungendo il campo PRIVATO come separatore dei gruppi verranno in tal modo distinti i record in base al loro valore assunto. Notare che questa funzionalità è superflua perché i record vengono già filtrati in questo progetto in base al valore attribuito al parametro $P{PRIVATO}.

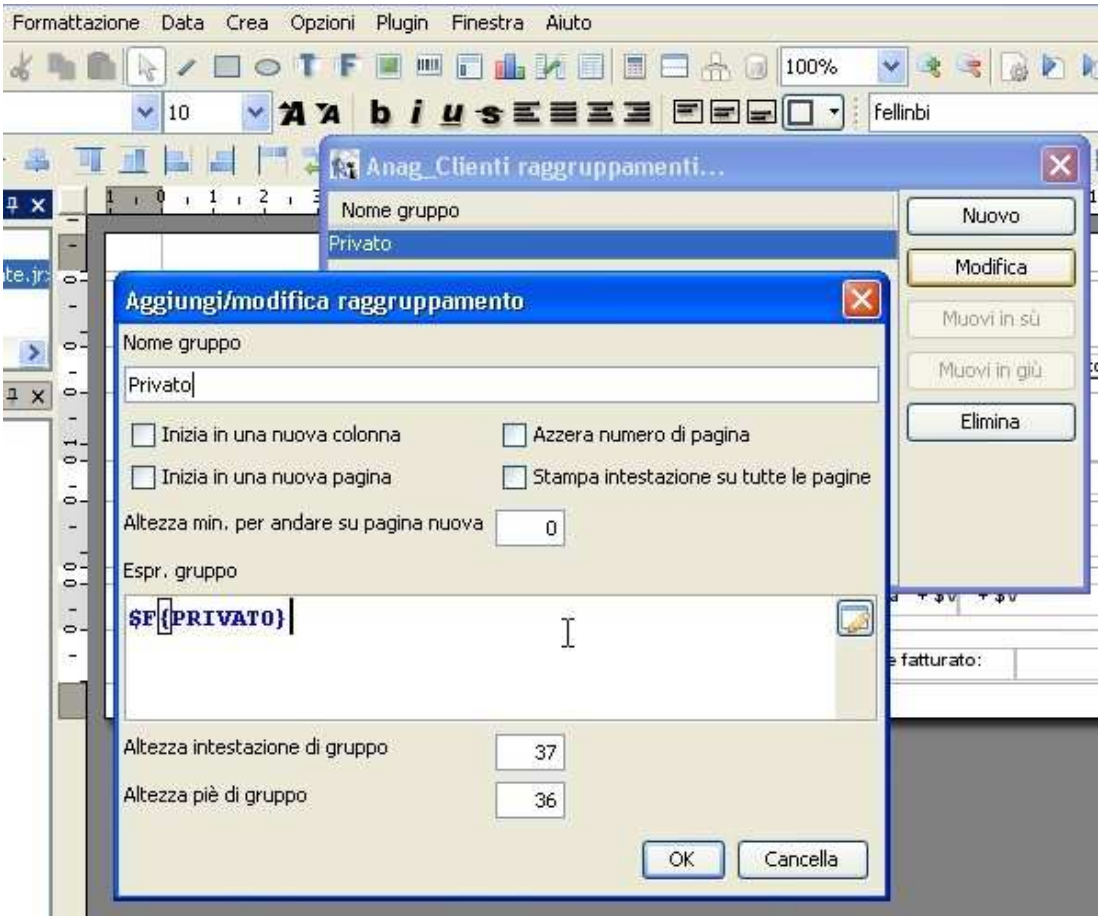


Illustrazione 85: Definizione di un raggruppamento dei dati

Illustrazione 86: Risultato della funzione di raggruppamento

L'ultimo passaggio sta nel convertire i valori “Y” o “N” di PRIVATO in “Si” o “No” utilizzando l'editor delle espressioni che appare cliccando sulla proprietà “Espressione” del campo. Ad ogni elemento di testo si può infatti associare una espressione in

linguaggio Groovy che ne elabori il risultato oppure esegua calcoli come totali, conteggi, stampa di data e ora, eccetera.

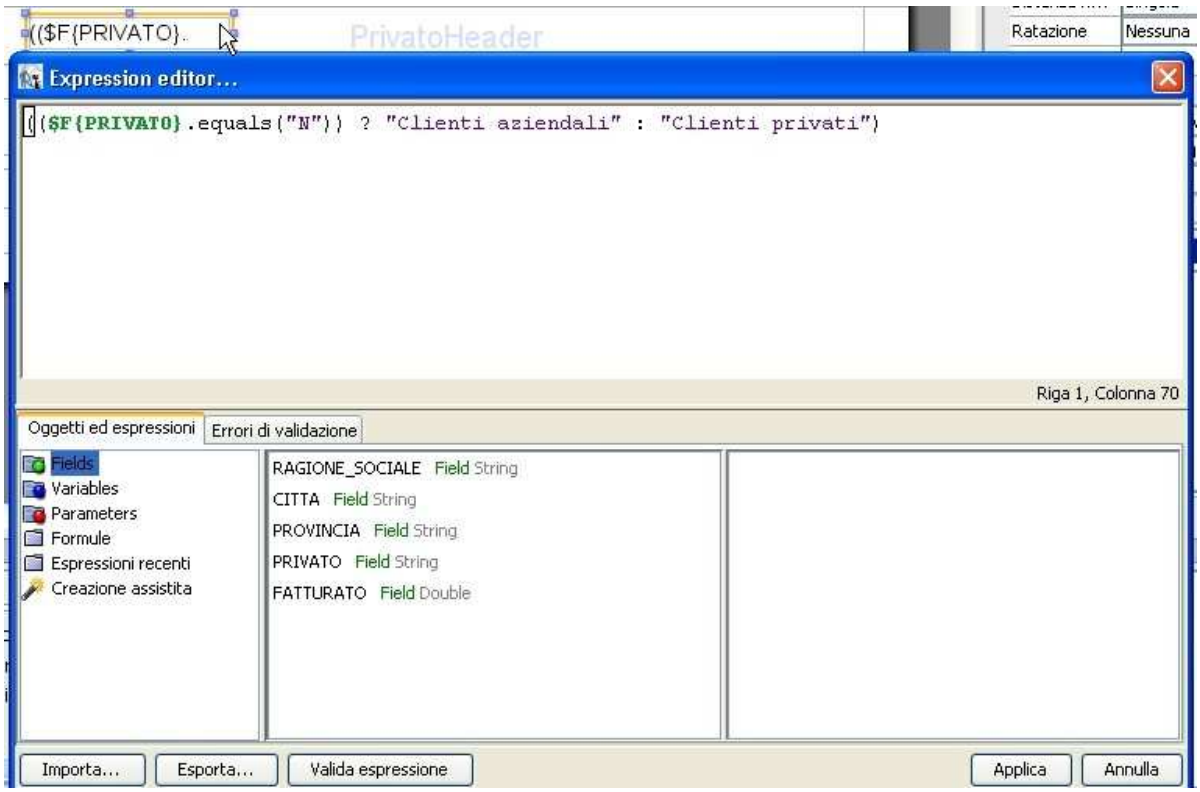


Illustrazione 87: Editor delle espressioni di un campo

Ad ogni momento con iReport è possibile testare il progetto per poi salvarlo su un file con estensione .jrxml. Per eseguire il report si provvederà a caricare il file nella piattaforma di Business Intelligence, ma è anche possibile lanciarlo direttamente da un programma Java utilizzando la libreria JasperReport.

Illustrazione 88: Anteprima del report

5.4.2 BIRT

Birt ha funzionalità paragonabili a JasperReport ma con una logica di posizionamento di componenti su tabelle piuttosto che su bande. Pur con differenze nel metodo di disegno il risultato è simile al report generato nel paragrafo 5.4.1.

Fatturato per Cliente

Ragione sociale	Citta	Prov.	Fatturato (€)
Clienti aziendali			
ARVAL SERVICE LEASE SPA	SCANDICCI	FI	
SAVARENT SPA	TORINO	TO	
LEASYS SPA	FIUMICINO	RM	
EUROCAR SPA	TRENTO	TN	
AUTOSTRADA DEL BRENNERO SPA	TRENTO	TN	
PASTORELLO SRL	TRENTO	TN	
LEASE PLAN ITALIA S.p.A.	ROMA	RM	
S.R.O. SCARL	GARDOLO	TN	
BIMOTOR SPA	BOLZANO	BZ	
AXUS ITALIANA SRL	ROMA	RM	
H3G ITALIA SPA	TREZZANO SUL NAVIGLIO	MI	
ALISEI SCARL	ROVERETO	TN	
EUROPCAR ITALIA SPA	ROMA	RM	
CLICKAR ASSISTANCE SRL	ARESE	MI	
COSTRUZIONI PISETTA LUIGI E C. SRL	ALBIANO DI TRENTO	TN	
FERRARI MAURO	RAVINA	TN	
ACI GLOBAL SPA	ROMA	RM	
AEFFELOGISTICA SRL	MATTARELLO	TN	

Illustrazione 89: Report Fatturato per cliente generato con BIRT

La fase di design prevede la creazione di due tabelle. La prima raccoglie il campo “Fatturato per cliente” di tipo Text, la seconda stampa i dati che provengono dalla query di Data Explorer.

La query usata è la seguente:

```
select c.ragione_sociale, c.citta, c.provincia, c.privato,
sum(r.importo_finale) fatturato
from riparazioni r
join clienti c on r.cod_cliente = c.cod_cliente
group by c.ragione_sociale, c.citta, c.provincia, c.privato
order by sum(r.importo_finale) desc
```

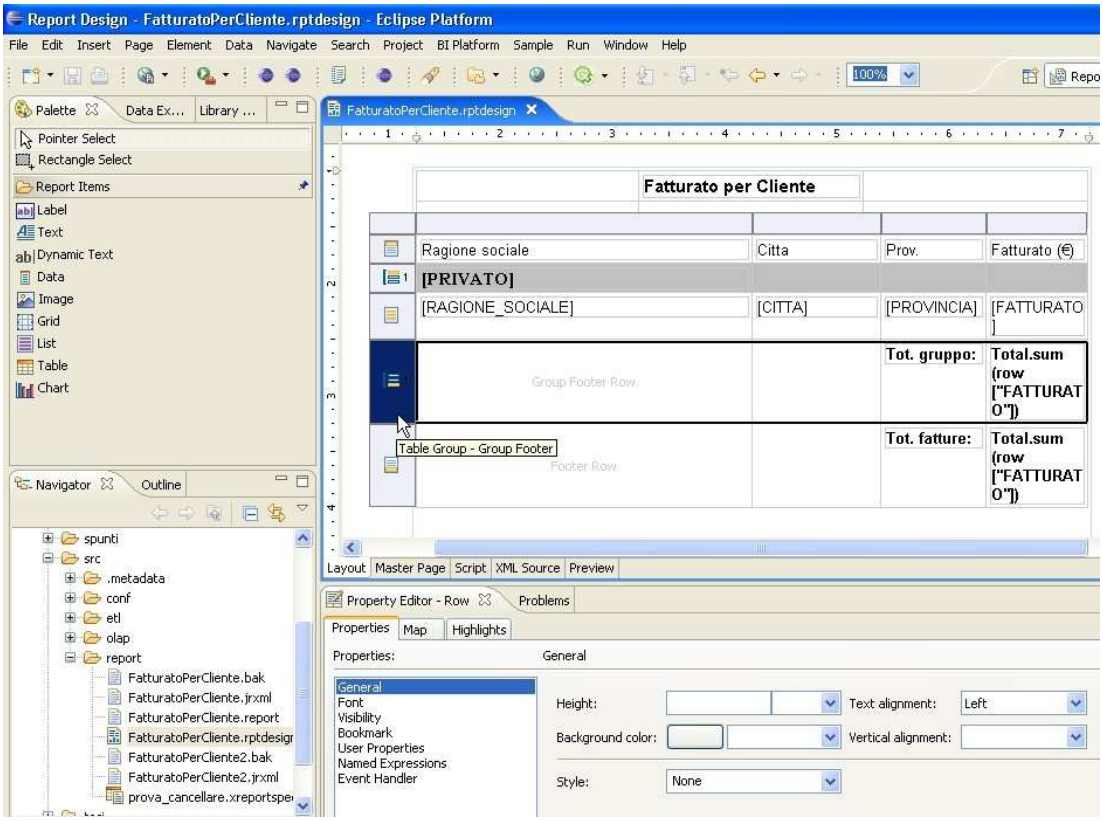


Illustrazione 90: Design del report Fatturato per cliente con BIRT

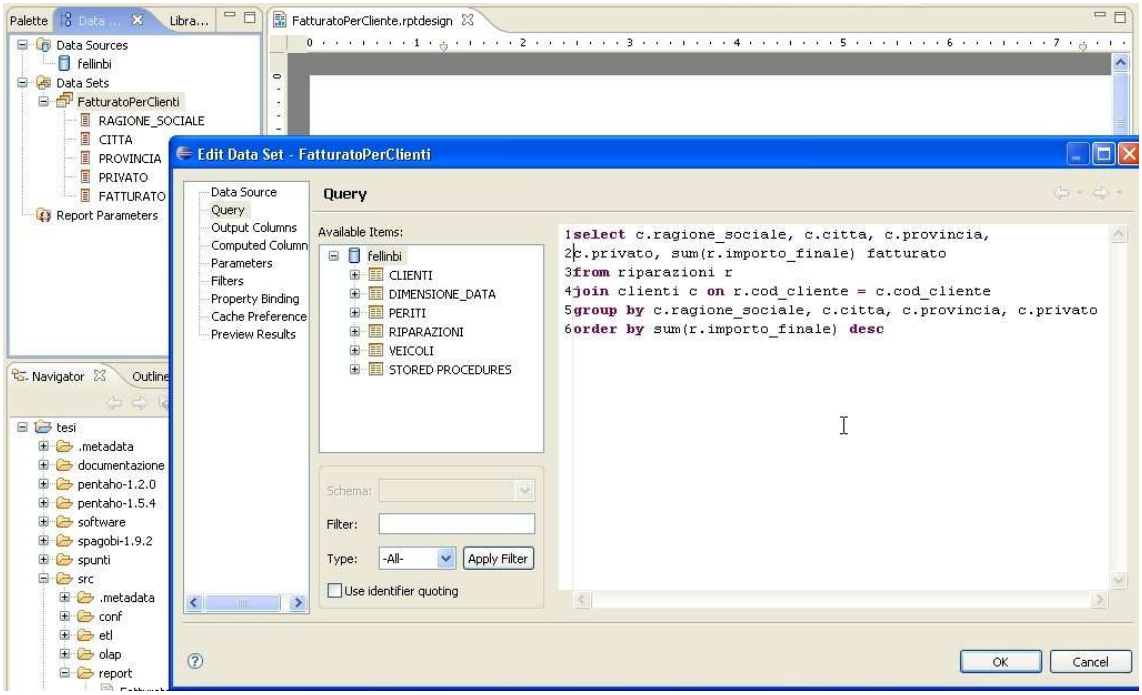


Illustrazione 91: Definizione della query del report

Anche BIRT prevede la possibilità di definire raggruppamenti usando il pannello Outline. In tal modo vengono a crearsi nella tabella le due bande “Group header row” e “Group footer row”.

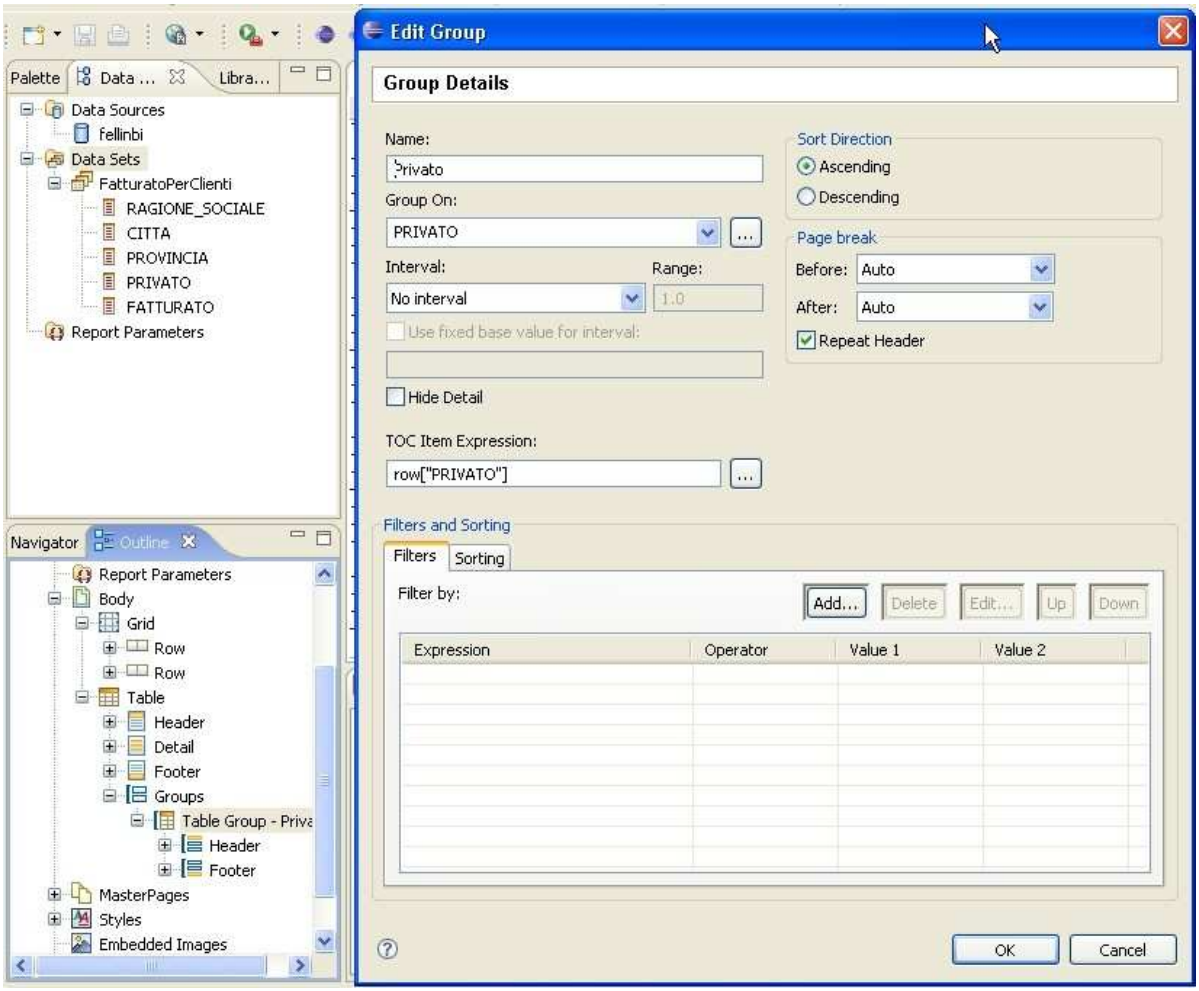


Illustrazione 92: Definizione di un gruppo in BIRT

Sui campi di tipo dinamico è previsto l'uso di espressioni JavaScript per calcolarne il valore. Nel nostro caso il codice di Illustrazione 93 permette di stampare il valore “Clienti aziendali” qualora il campo privato sia “N”, altrimenti stampa “Clienti privati”. Usando le espressioni si è aggiunto anche il campo “Totale gruppo” e “Totale fatture” con la funzione Total.sum(row["FATTURATO"]) rispettivamente nella banda “Group header row” e “ Group footer row”.

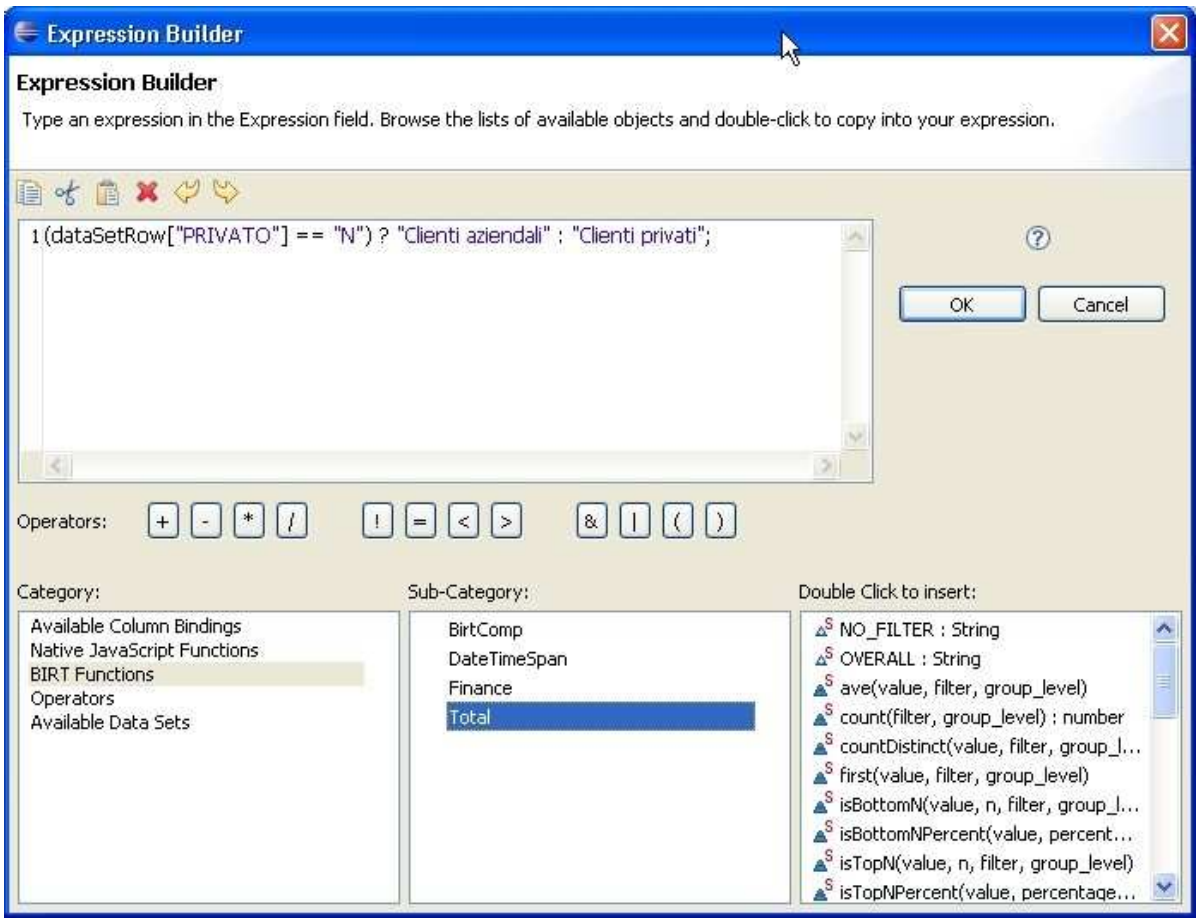


Illustrazione 93: Espressione di formattazione del campo PRIVATO

La formattazione grafica dei campi avviene applicando stili CSS che vengono aggiunti nell'elenco Styles tramite un comodo editor.

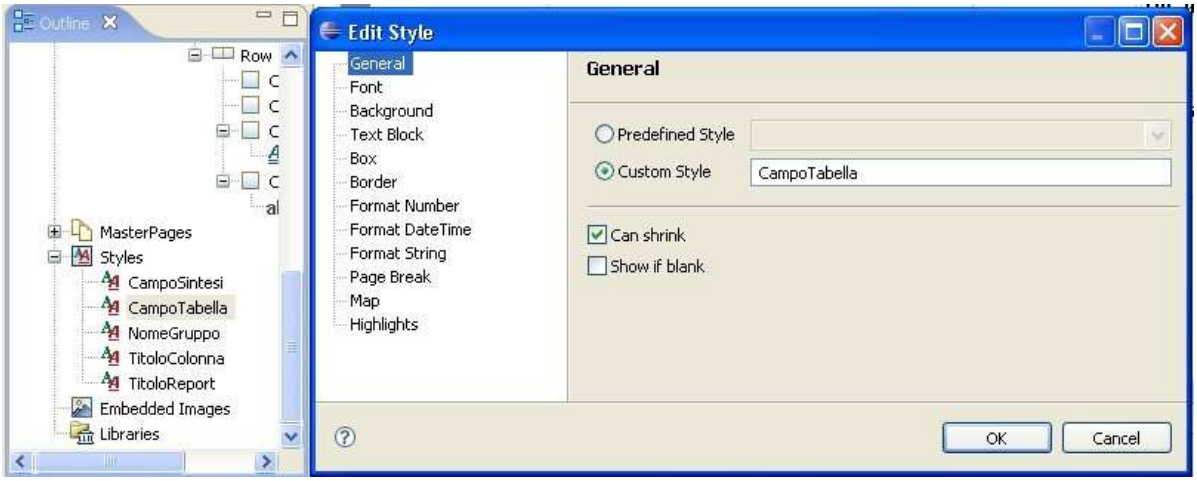


Illustrazione 94: Definizione di uno stile CSS

In ogni momento è possibile testare il report selezionando la scheda Preview. Il progetto viene salvato su un file con estensione .rptdesign e interpretato usando le librerie Java di BIRT o caricato all'interno delle piattaforme di Business Intelligence PentahoBI e SpagoBI.

5.4.3 JFreeReport

Lo stesso report viene realizzato con JFreeReport e precisamente usando lo strumento di design Pentaho ReportDesigner.

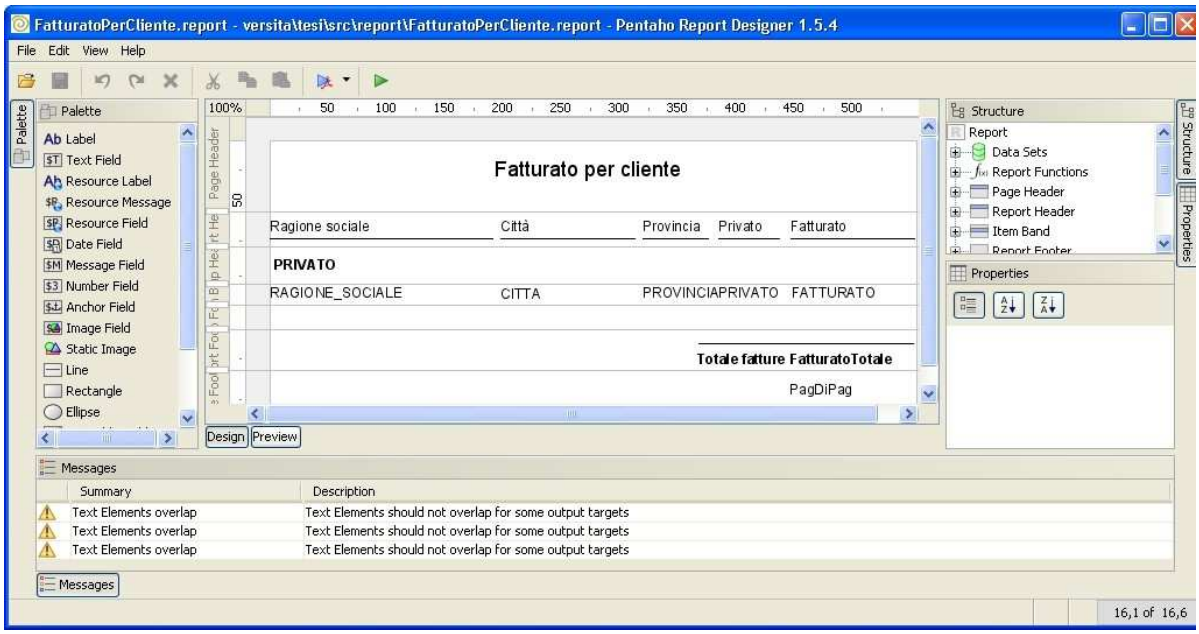


Illustrazione 95: Pentaho Report Designer

Come nel caso di BIRT e JasperReport il primo passo sta nel registrare una nuova connessione al database e definire la query sorgente. Pentaho ReportDesigner provvederà a registrare i campi della query, i cui valori possono essere attribuiti ai componenti di testo che vengono trascinati nel report.

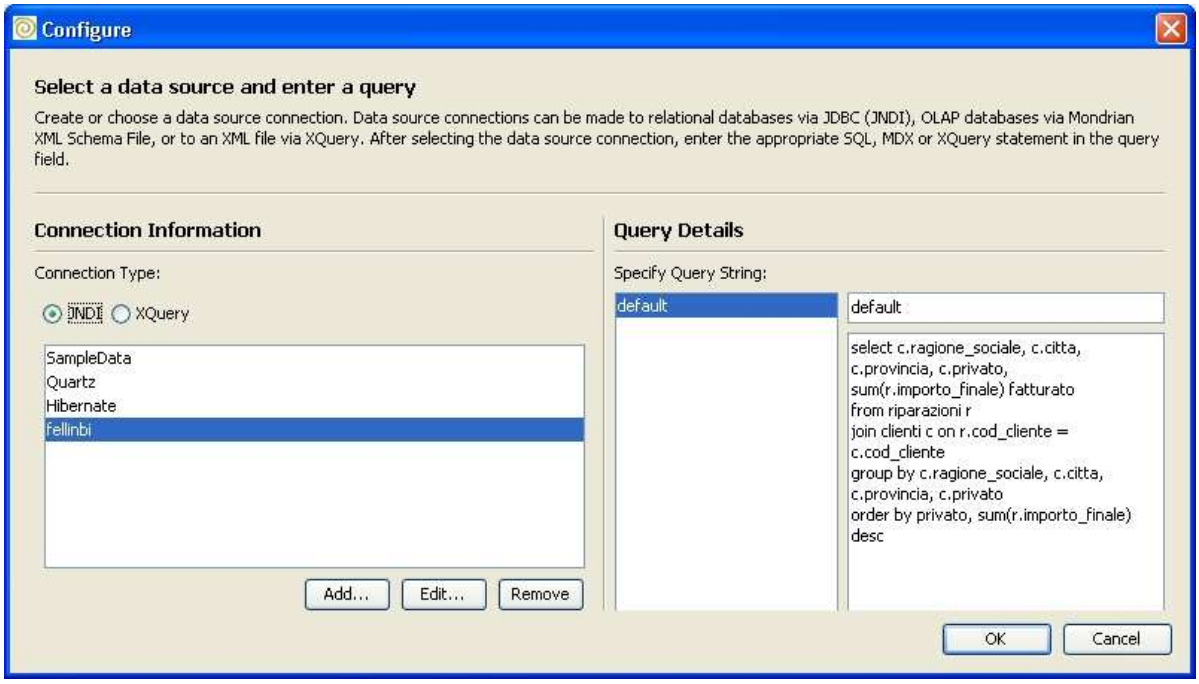


Illustrazione 96: Definizione della query

La logica di funzionamento è ancora una volta a bande ed è previsto l'uso di espressioni nei campi per calcolare il valore durante l'esecuzione. Nella scrittura delle espressioni, che sono in linguaggio JavaScript, si nota l'assenza di strumenti assistiti di compilazione, come ad esempio funzioni per il debug o il semplice elenco di metodi disponibili. Trattandosi di un prodotto non ancora completo e in fase di maturazione la cosa potrebbe risolversi nei futuri rilasci del programma.

Illustrazione 97: Espressione sul campo PRIVATO

La creazione di raggruppamenti avviene aggiungendo una nuova voce all'elenco Groups. Nel progetto è stato creato un gruppo in base al valore assunto dal campo Privato. Leggermente diversa da BIRT e JasperReport è l'aggiunta di campi calcolati come sommatorie o numero di pagina. In Illustrazione 99 si nota come le funzioni siano riportate nell'elenco Properties e, una volta definite, appaiono nel menù ReportFunctions con una etichetta che può essere usata nei campi del report.

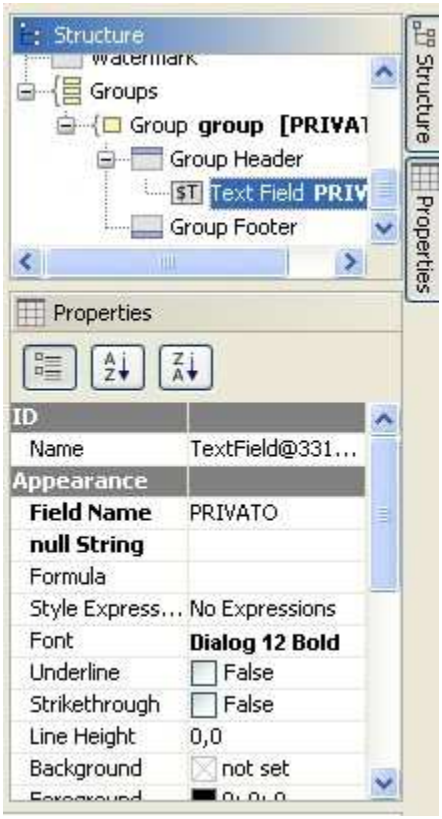


Illustrazione 98: Definizione di un gruppo

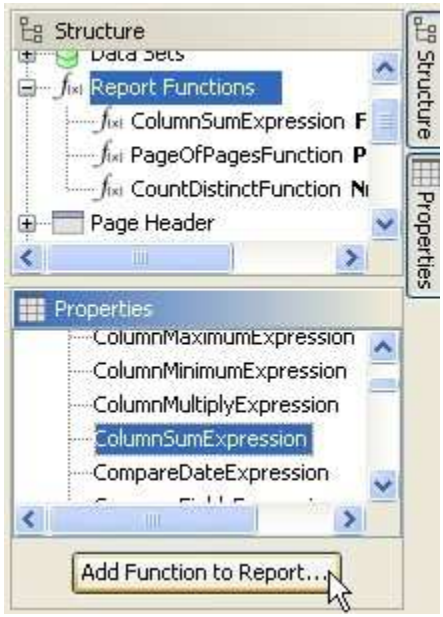


Illustrazione 99: Aggiunta di campi calcolati

Il risultato del lavoro, che può essere salvato in un file con estensione .report, è visibile nella seguente figura.

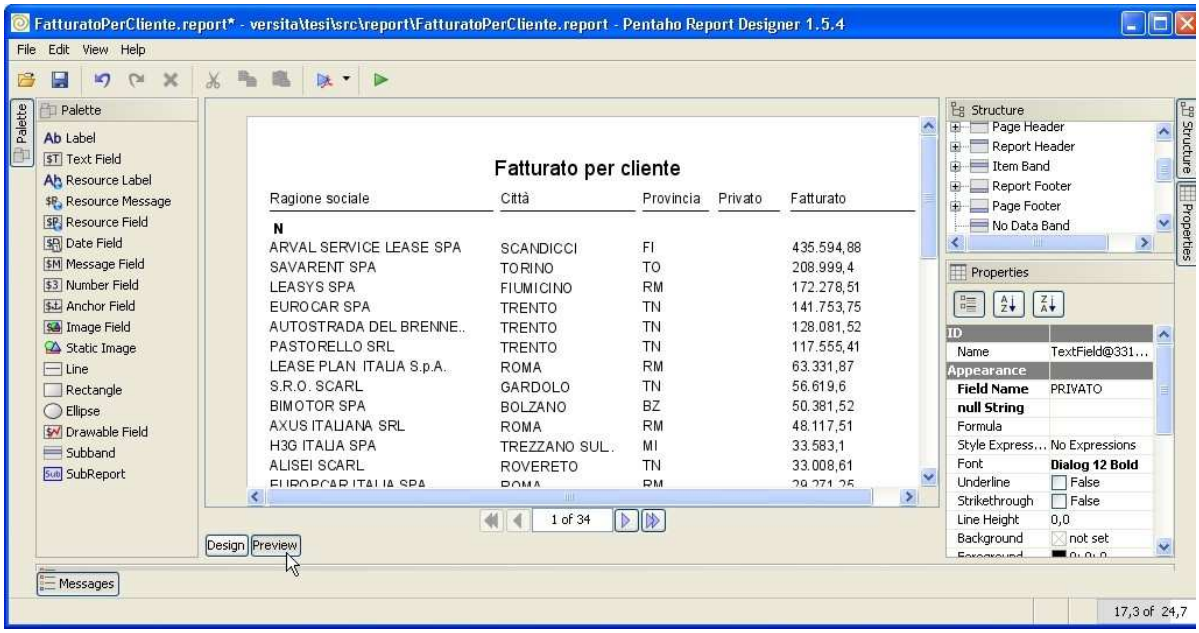


Illustrazione 100: Collaudo del report

5.5 Implementazione di PentahoBI

5.5.1 Installazione

5.5.1.1 Installazione lato server

L'installazione di PentahoBI può essere fatta in modi diversi. Ad esempio si può usare Pentaho-demo che contiene l'installazione stand-alone di un ambiente dimostrativo personalizzabile. Oppure si può procedere installando manualmente i componenti della piattaforma secondo una procedura che varia a seconda dai casi e può essere riassunta in:

- installare uno dei Java Web Server a scelta tra JBoss e Tomcat
- compilare il pacchetto .war (o .ear per JBoss) usando ant per poi copiarlo nella cartella del Web Server
- copiare il pacchetto Pentaho-data contenente l'RDBMS HSQLDB di supporto alla piattaforma. In alternativa si può usare un database MySQL
- copiare la repository delle soluzioni Pentaho-solutions in una cartella accessibile al Web Server
- editare a mano i file di configurazione che variano a seconda del Web Server impiegato e la configurazione scelta

L'installazione manuale è comunque sconsigliata dalla Pentaho stessa per la presenza di molteplici dettagli implementativi e nozioni tecniche da comprendere. Per chi si avvicina per la prima volta alla piattaforma Pentaho BI è disponibile il recente installer per Pentaho 1.2, con cui si ottiene un cartella contenente il Web Server JBoss e tutti i file necessari alla piattaforma per funzionare.

Per questo progetto è stato impiegato PentahoBI 1.2 usando l'installer e per valutare alcune funzionalità in via di sviluppo è stata fatta un'installazione manuale di Pentaho BI 1.5.4 Milestone 4 su Web Server Apache Tomcat 5.5, di cui si rimanda alla documentazione “Manual Deployment of Pentaho” per la procedura dettagliata.

Purtroppo su Apache Tomcat tutte le release in fase di sviluppo dalla 1.5.1 alla 1.5.4 come pure la release di produzione 1.2 hanno problemi ad eseguire analisi OLAP, fatto che ha pregiudicato l'adozione dell'ottimo Apache Tomcat come Application Server di riferimento. Pur consultando la documentazione e i forum non si è riusciti a risolvere il problema.

Illustrazione 101: JPivot su Apache Tomcat da errori di esecuzione

Per quanto riguarda il database viene impiegato Firebird 2.0.1 già precedentemente installato nel server Server01. L'unico passaggio è stato creare il database FellinBI e lo schema relazionale discusso nel paragrafo 4.4.

Una volta installato PentahoBI 1.2 usando l'installer e Pentaho 1.5.4 con la procedura manuale si ha a disposizione un ambiente

funzionante ma che va riadattato alle proprie esigenze. Questi argomenti vengono trattati nel successivo paragrafo.

5.5.1.2 Configurazioni iniziali

Puntando il browser sull'URL http://localhost:8080/Pentaho è possibile accedere all'interfaccia della piattaforma, una volta che questa è stata mandata in esecuzione lanciando lo script start-pentaho.bat o il relativo servizio NT. Per consentire l'accesso ai clienti remoti si modifica il file /jboss/server/default/deploy/pentaho.war/WEB-INF/web.xml (se si usa JBoss oppure /tomcat/webapps/Pentaho/WEB-INF/web.xml se si usa Tomcat) sostituendo la stringa localhost con il nome del server (server01 nel nostro caso).

Rimane da configurare l'ambiente per la connessione al database Firebird verso cui la piattaforma attingerà ai dati. In JBoss l'operazione viene realizzata riportando i parametri di connessione nel file fellinbi-ds.xml che va copiato in \jboss\server\default\deploy. Il contenuto del file è in pratica una connessione JNDI[12] resa disponibile all'intero Web Server JBoss:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
      <local-tx-datasource>
      <jndi-name>fellinbi</jndi-name>
      <connection-url>
      jdbc:firebirdsql:localhost:E:/database/Firebird/FELLINBI.FDB
      </connection-url>
```

12 La tecnologia JNDI consente di definire connessioni a sorgenti dati, identificandole con un nome univoco che può essere richiamato dagli applicativi Java. In tal modo si possono condividere connessioni comuni, senza definire di volta in volta i parametri di accesso.

```
      <driver-class>org.firebirdsql.jdbc.FBDriver</driver-class>
      <user-name>SYSDBA</user-name>
      <password>***</password>
      </local-tx-datasource>
</datasources>
```

Si prosegue modificando il file /jboss/server/default/deploy/pentaho.war/WEB-INF/web.xml riportando tra i tag <web-app> e </web-app> lo spezzone:

```
    <resource-ref>
      <description>fellinbi</description>
      <res-ref-name>jdbc/fellinbi</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
    </resource-ref>
```

Anche il file /jboss/server/default/deploy/pentaho.war/WEB-INF/jboss-web.xml va aggiornato con la seguente stringa:

```
  <resource-ref>
      <res-ref-name>jdbc/fellinbi</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <jndi-name>java:/fellinbi</jndi-name>
  </resource-ref>
```

Per quanto riguarda Apache Tomcat (installazione manuale) è sufficiente modificare il singolo file \tomcat5.5\conf\server.xml aggiungendo all'interno del tag `<Context path="/Pentaho" docbase="webapps/Pentaho/"> </Context>` il codice:

```
<Resource name="jdbc/fellinbi"
      auth="Container" type="javax.sql.DataSource" maxActive="20"
      maxIdle="5" maxWait="10000" username="SYSDBA" password="***"
      factory="org.apache.commons.dbcp.BasicDataSourceFactory"
      driverClassName="org.firebirdsql.jdbc.FBDriver"
      url="jdbc:firebirdsql:localhost:E:/database/Firebird/FELLINBI.FDB"
/>
```

Infine si aggiunge al file \Pentaho-solutions\system\simple-jndi la stringa:

```
fellinbi/type=javax.sql.DataSource
fellinbi/driver=org.firebirdsql.jdbc.FBDriver
fellinbi/url=jdbc:firebirdsql:localhost:E:/database/Firebird/FELLINBI.FDB
fellinbi/user=SYSDBA
fellinbi/password=***
```

Completa l'installazione sia su JBoss che su Tomcat la modifica dei file:

- /Pentaho_solutions/system/smtp-email/email_config.xml riportando i parametri di accesso al server SMTP (funzionalità non adoperata nel progetto)
- /Pentaho-solutions/system/pubblisher_config.xml riportando la password di accesso alla repository per i tools di sviluppo

Navigando tra le pagine della piattaforma si nota che sono creare in lingua inglese e tradotte solo parzialmente in italiano. Il meccanismo con cui PentahoBI traduce il testo è concettualmente semplice:

La piattaforma usa il file messages_it.properties di \jboss\server\default\deploy\pentaho.war\WEB-INF\classes\org\Pentaho\locale\messages_it.properties (in Apache Tomcat \tomcat5.5\webapps\Pentaho\WEB-INF\classes\org\Pentaho\locale\messages_it.properties) ricercando la traduzione in base a un campo chiave. Per le altre lingue il file è presente con il nome message_<codice lingua>.properties. La traduzione è applicabile anche a livello di documento analitico, creando un file <nome_action_sequence>_<codice

lingua>.properties nella cartella in cui è installata l'Action Sequence. Se la piattaforma non trova la versione italiana di questi file utilizzerà la versione inglese. Alcune voci non implementano ancora il meccanismo di traduzione e rimangono in inglese pur aggiungendo le rispettive chiavi nel file della lingua italiana.

Non è documentato come sia possibile creare un nuovo albero delle soluzioni oltre a samples e registrarlo all'interno della piattaforma. Per tale motivo si è creata la sotto cartella FellinBI in samples che ospita i futuri documenti analitici, oltre ai file necessari al al riconoscimento della cartella (index.xml, index.properties e relative icone). È assente al momento una procedura automatica per la creazione di sottocartelle al fine di organizzare i documenti analitici, per cui si deve provvedere a creare manualmente la cartella copiandovi i file index.xml e index.properties prendendoli come spunto dagli esempi a disposizione.

Un'altra configurazione utile è l'accesso utente, disponibile solo nelle versioni sperimentali 1.5 di Pentaho mentre nella versione 1.2 chiunque con accesso al server web può usare la piattaforma.

Illustrazione 102: Accesso utente in Pentaho 1.5

La modifica della maschera benché importante in un'ambiente di produzione reale richiede un intervento diretto alle pagine JSP e al database. Per esempio, per aggiungere l'utente fellin alla piattaforma è necessario prima aggiornare il database con i seguenti passaggi:

1) in un prompt MS-DOS o una shell UNIX avviare il database HSQLDB con il seguente comando:

```
java -cp fellinbi/Pentaho-data/lib/hsqldb.jar org.hsqldb.Server
-database.0 database -dbname.0 database -port 9002
```

2) in un altro prompt aprire il tool per la connessione al database

```
java -cp fellinbi/Pentaho-data/lib/hsqldb.jar org.hsqldb.util.DataManager
```

3) eseguire dal tool appena aperto le query

```
INSERT INTO JBP_USERS VALUES(5, 'fellinbi', NULL, NULL,
'5f4dcc3b5aa765d61d8327deb882cf99', '', '', '2007-05-10
00:00:0.000000000', FALSE, TRUE);
INSERT INTO JBP_ROLE_MEMBERSHIP VALUES(5,1); /*role Admin per fellinbi */
INSERT INTO JBP_ROLE_MEMBERSHIP VALUES(5,2); /*role User per fellinbi */
```

dove la stringa 5f4dcc3b5aa765d61d8327deb882cf99 è l'hash MD5 della password inserita nella maschera di accesso.

Quindi si modifica a mano la pagina login.jsp della cartella pentaho.war\jsp del server cambiando il frammento di codice

```
<select
onchange="document.forms.login.elements.j_username.value=this.options[this
.selectedIndex].value;
document.forms.login.elements.j_password.value='password'">
<option value="" selected><%= Messages.getString("UI.USER_SELECT")
%></option>
<option value="suzy">Suzy</option>
<option value="joe">Joe (<%= Messages.getString("UI.USER_ADMIN")
%>)</option>
<option value="pat">Pat</option>
```

```
<option value="tiffany">Tiffany</option>
</select>
```

nel seguente frammento:

```
<select
onchange="document.forms.login.elements.j_username.value=this.options[this
.selectedIndex].value">
<option value="fellinbi">FellinBI</option>
</select>
```

Al riavvio della piattaforma si può accedere con il nuovo utente FellinBI:

Illustrazione 103: Personalizzazione login in Pentaho 1.5

Ultimate le modifiche la piattaforma è pronta per installare nuovi applicativi di Business Intelligence.

5.5.1.3 Installazione lato client

L'installazione lato client gode dei vantaggi amministrativi delle soluzioni web-based, per cui l'unico passaggio necessario è installare un browser web di ultima generazione. Sui client aziendali è già installato Internet Explorer versioni 6 e 7, ma teoricamente qualsiasi browser moderno è adatto allo scopo.

I report in formato PDF richiedono l'apposito plugin del browser Adobe Reader, mentre per chi volesse provare i grafici in stile Adobe Flash presenti nelle versioni di sviluppo 1.5 è necessario l'installazione del plugin Adobe Flash Player. Non è richiesto dal lato client il supporto a Java.
Nel nostro caso l'unico intervento tecnico è stato aggiungere al desktop degli utenti il link che punti a http://<host server>:8080/pentaho.

5.5.1.4 Installazione ambiente di sviluppo

Per la macchina di sviluppo sono richiesti l'installazione dei seguenti pacchetti:

- Pentaho Design Studio per lo sviluppo delle Action Sequence
- Pentaho Report Designer per creare reports JFreeReport in maniera più articolata rispetto alla procedura assistita via web
- Pentaho Cube Designer o il Mondrian Schema Workbench per la definizione dello schema multidimensionale Mondrian
- Strumenti di reportistica come iReport per JasperReport o il plug-in BIRT per Eclipse

Essendo programmi scritti in Java la loro installazione è molto semplice e si riduce il più delle volte a scompattare un archivio compresso all'interno della cartella di lavoro.

Come interfaccia al database Firebird si può utilizzare l'ottimo editor IBExpert di HK Software, disponibile in versione Personal in forma gratuita per uso non commerciale.

Il testing degli applicativi risulta un'operazione molto importante e, non disponendo PentahoBI di cicli di approvazione e collaudo dei documenti analitici come invece avviene in SpagoBI, la soluzione

migliore sta nell'installare PentahoBI nella macchina di utilizzo per poi aggiornare il server di produzione con gli applicativi collaudati. Particolarmente adatto allo scopo di testing sarebbe una installazione su Apache Tomcat, molto più leggera e veloce dell'Application Server JBoss[13]. Per la presenza di un baco non risolto che pregiudica il funzionamento di analisi OLAP con Mondrian e JPivot si è il più delle volte costretti ad utilizzare JBoss, soluzione che incide sulle prestazioni delle macchine non particolarmente veloci.

5.5.2 ETL per il caricamento dei dati

Il caricamento dei dati avviene in automatico ad intervalli prefissati. E' comunque buona norma consentire all'utente di eseguire l'aggiornamento direttamente dal portale, predisponendo un apposito comando.

Usando Pentaho Design Studio la cosa si riduce alla creazione di una nuova Action Sequence chiamata AggiornaFellinBI e salvata nella cartella /pentaho-solutions/sample/fellinbi/strumenti. Nell'Illustrazione 104 si possono vedere i parametri di registrazione nella repository, mentre la parte esecutiva è quella dell' Illustrazione 105.

13 sulla macchina di sviluppo Tomcat 5.5 con Pentaho BI usa circa 80 Mb di RAM, che diventano circa 190 Mb di RAM quando si usa JBoss. Inoltre le prestazioni di Tomcat sono notoriamente migliori di JBoss per questioni legate alle scelte architetturali.

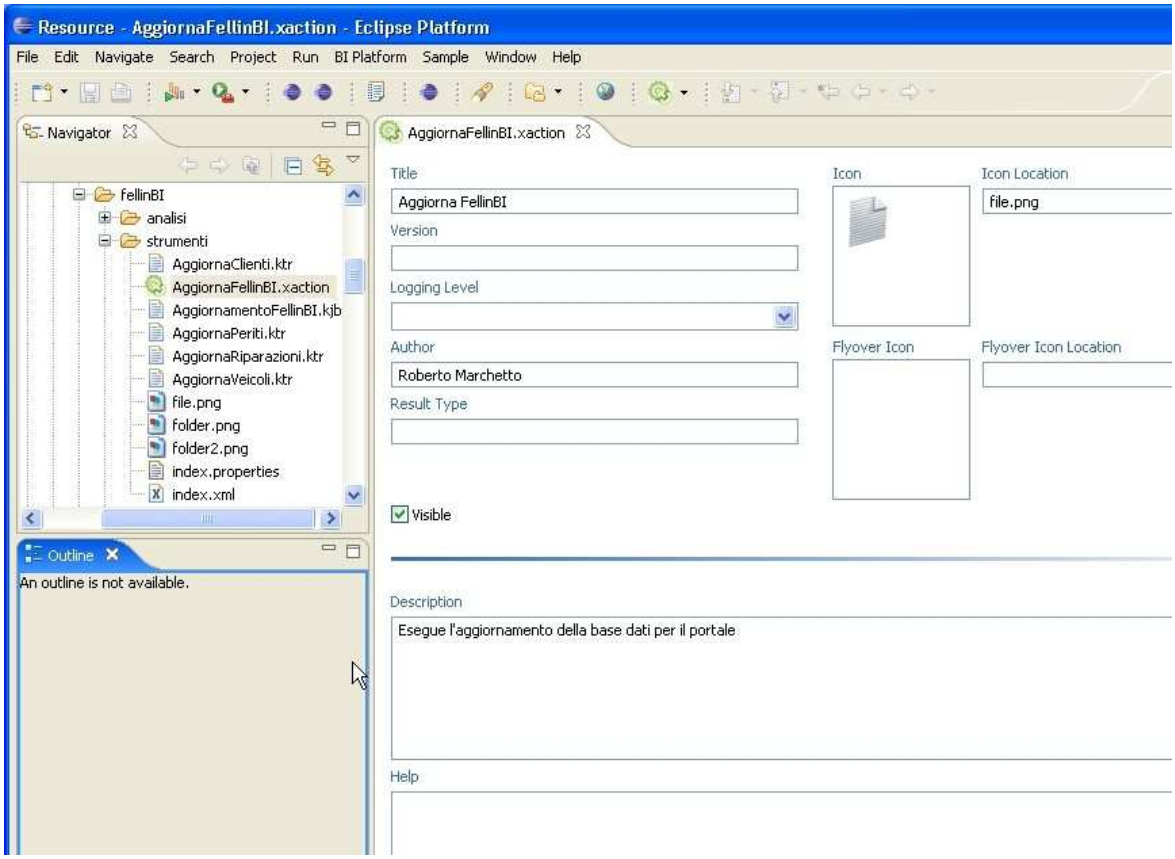


Illustrazione 104: Informazioni di facciata di AggiornaFellinBI

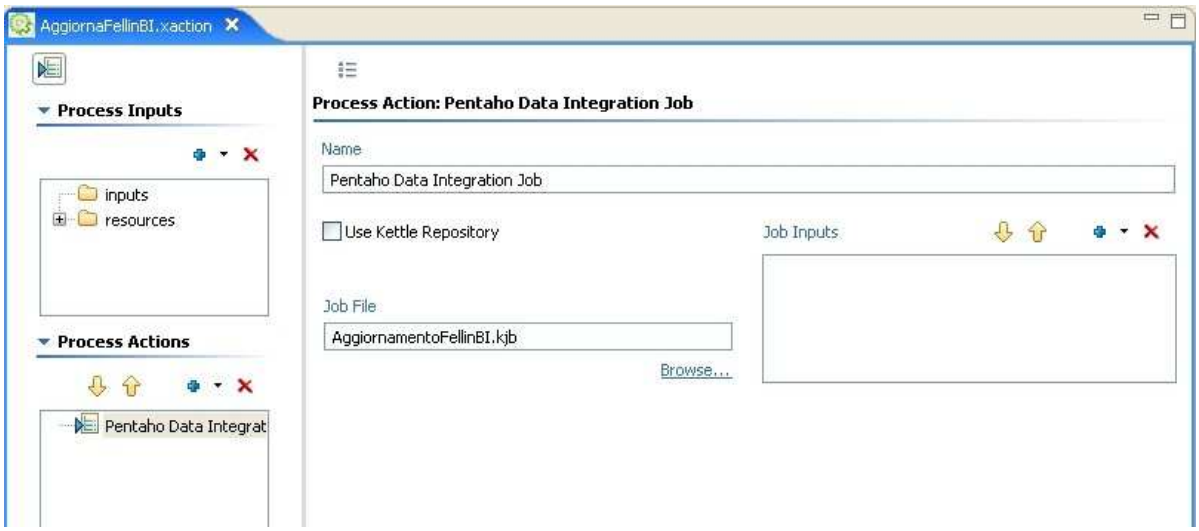


Illustrazione 105: Definizione processo di AggiornaFellinBI

In questo modo l'utente può lanciare l'aggiornamento della base dati semplicemente cliccando sul comando AggiornaFellinBI nella

repository dei documenti analitici.

Illustrazione 106: L'Action Sequence per l'aggiornamento della base dati

5.5.3 Reportistica

Si tratta di registrare nella piattaforma il report del paragrafo 5.4.1, cominciando a compilare una nuova Action Sequence con le seguenti informazioni di facciata (scheda General):

Title	Fatturato per cliente
Version	1.0
Loggin Level	ERROR
Icon Location	file.png
Visible	Spuntato
Description	Fatturato per cliente, diviso tra clienti privati e aziendali

Il file una volta salvato sulla cartella /pentaho-solutions/sample/fellinbi/reportistica dovrebbe dare il risultato seguente:

Illustrazione 107: L'Action Sequence registrata

Il report Fatturato per cliente prevede l'uso del parametro Privato con valori Y/N in modo da distinguere i clienti clienti privati (valore Y) da quelli non privati (valore N). Per aggiungere un nuovo parametro si utilizza l'elenco “Process Inputs”, dove va anche definito l'elenco dei valori Y/N che lo costituiscono.

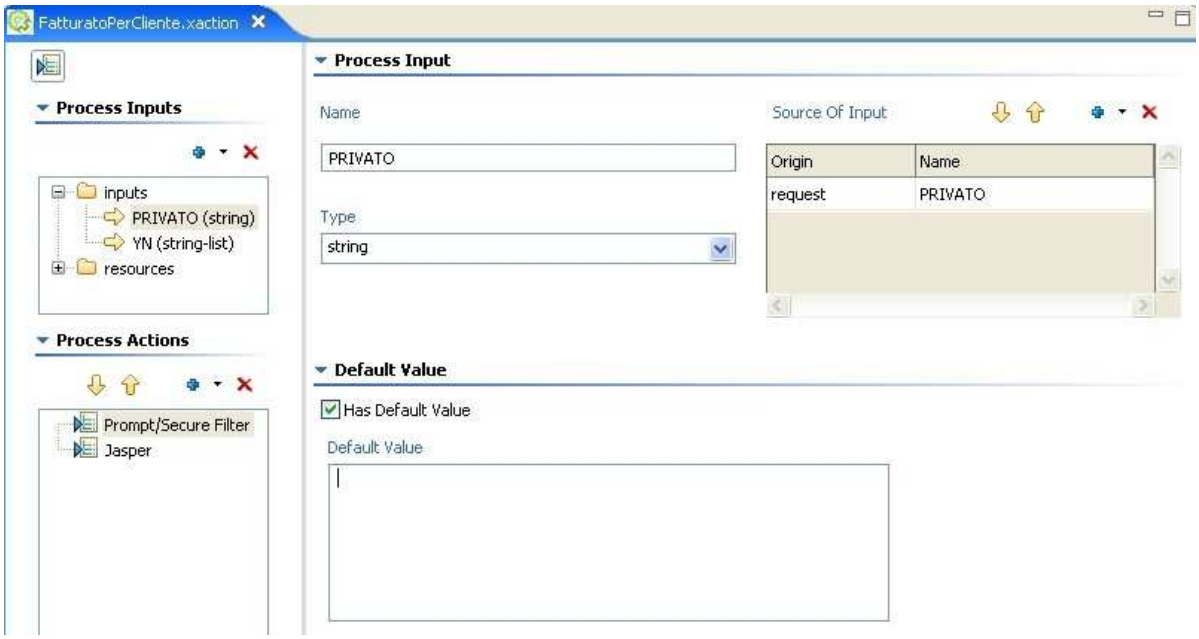


Illustrazione 108: Definizione di un nuovo parametro

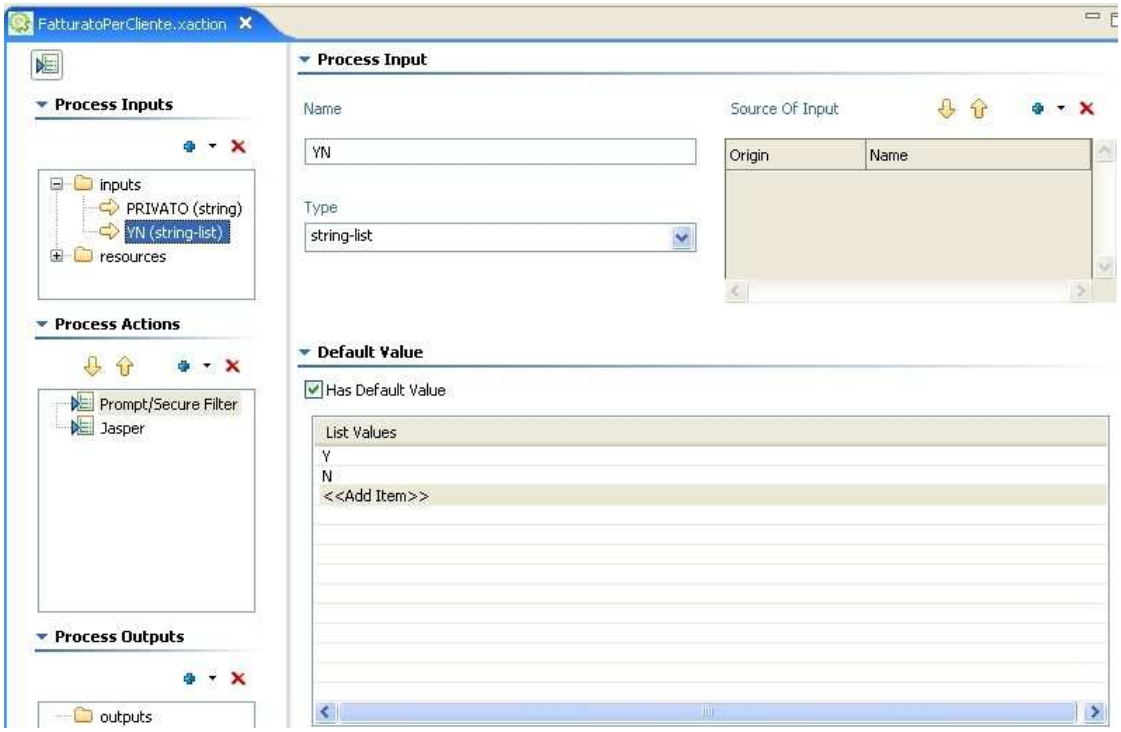


Illustrazione 109: Definizione di un elenco valori

Aggiungendo un componente “Prompt/Secure Filter” all'elenco “Process Actions” verrà chiesto all'utente in fase di esecuzione il valore da attribuire al parametro PRIVATO.

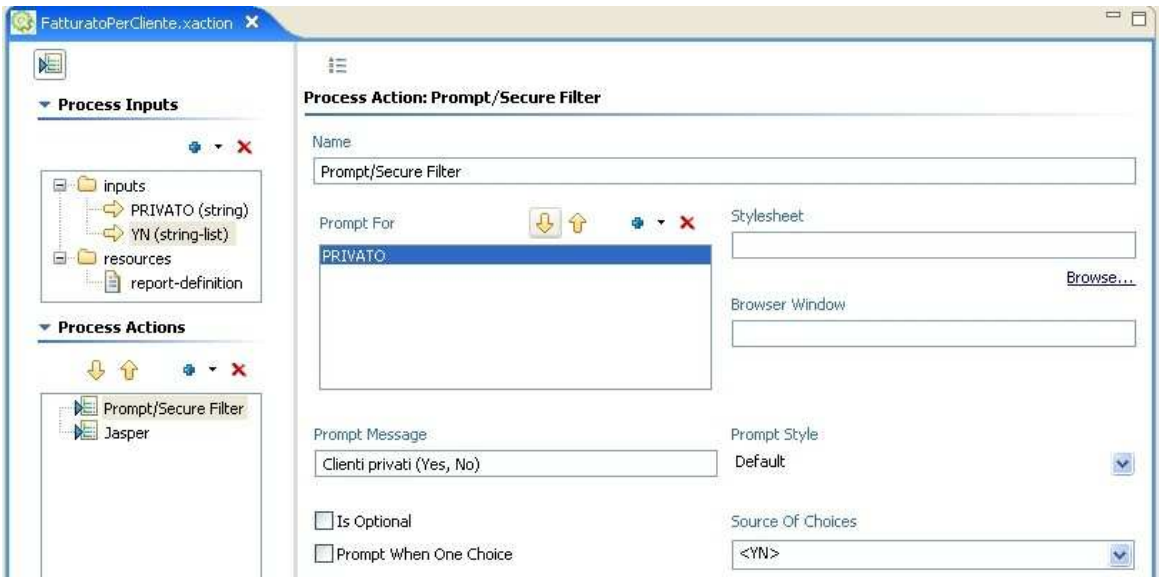


Illustrazione 110: Componente per la compilazione del parametro

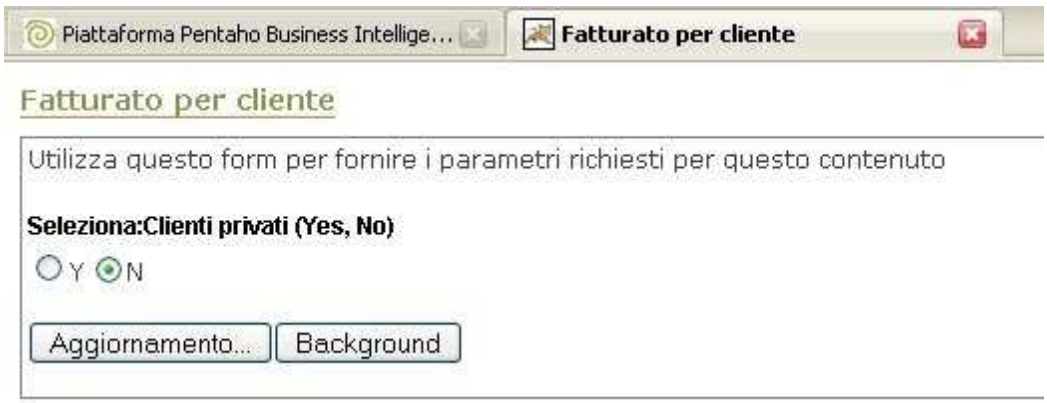


Illustrazione 111: Form di richiesta dei parametri

Completa l'Action Sequence la registrazione del file del report .jrxml, usando un componente “Jasper”.

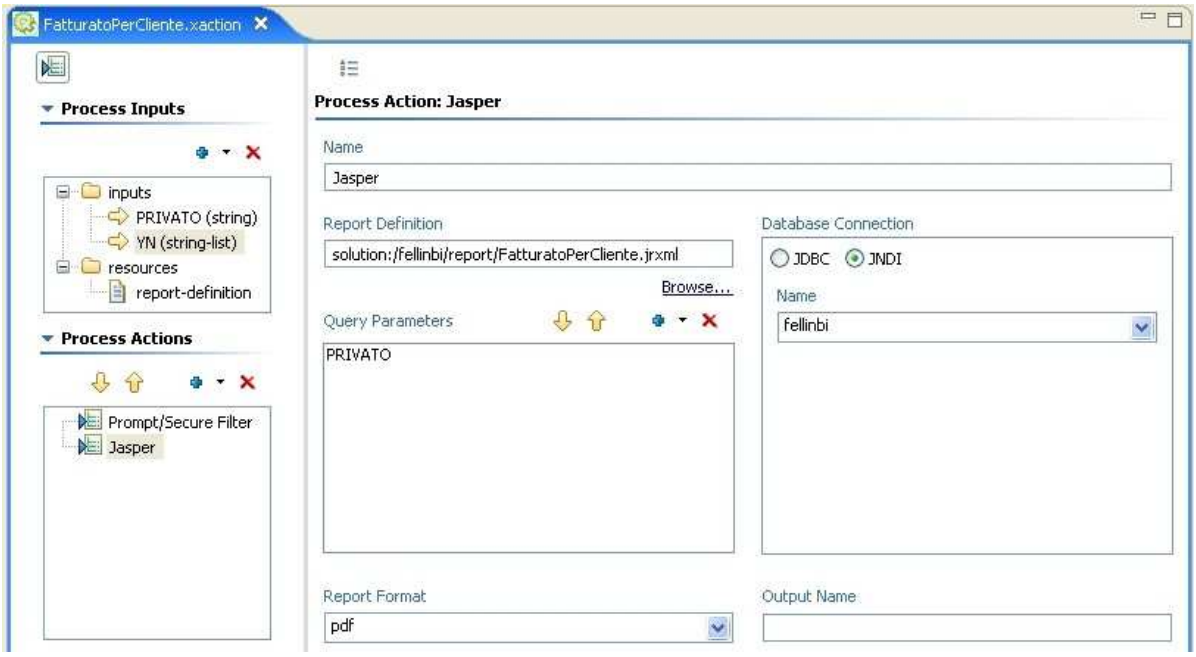


Illustrazione 112: Definizione del collegamento al report

Il risultato finale è visibile nella seguente figura:

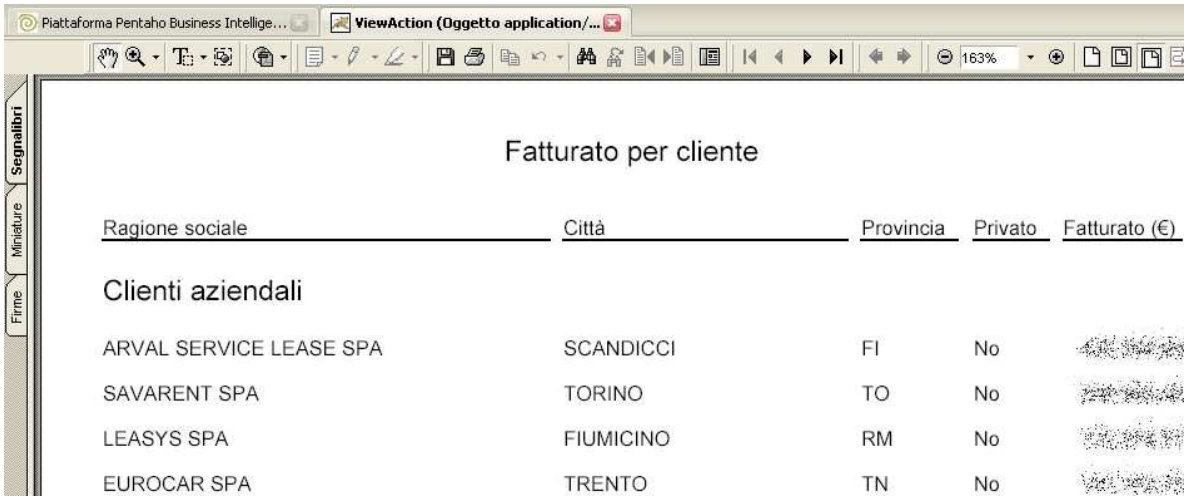


Illustrazione 113: Report eseguito in PentahoBI

Anche in questo caso come nella registrazione dell'analisi OLAP si è sentita la carenza di menù a tendina funzionanti e procedure assistite, mentre le informazioni di debug stampate durante il test dell'applicativo non si sono dimostrate sufficientemente chiare per risolvere gli eventuali problemi di esecuzione.

Paragonabile a JasperReport è il risultato ottenuto caricando i report BIRT e JFreeReport, la cui procedura di registrazione è simile al caso di JasperReport e viene tralasciata.

Illustrazione 114: Report fatturato per cliente nella variante BIRT

Illustrazione 115: Report fatturato per cliente nella variante JFreeReport

5.5.4 Analisi OLAP

Per registrare il cubo multidimensionale del paragrafo 5.3 in Pentaho si procede creando una nuova Action Sequence con le seguenti informazioni della scheda General:

Title	OLAP su Riparazioni
Version	1.0
Loggin Level	ERROR
Icon Location	file.png
Visible	Spuntato
Description	Analisi OLAP sulle riparazioni

Il documento analitico nella repository appare come nella seguente immagine:

Illustrazione 116: Elenco dei documenti analitici

Per la parte esecutiva l'Action Sequence va compilata come segue:

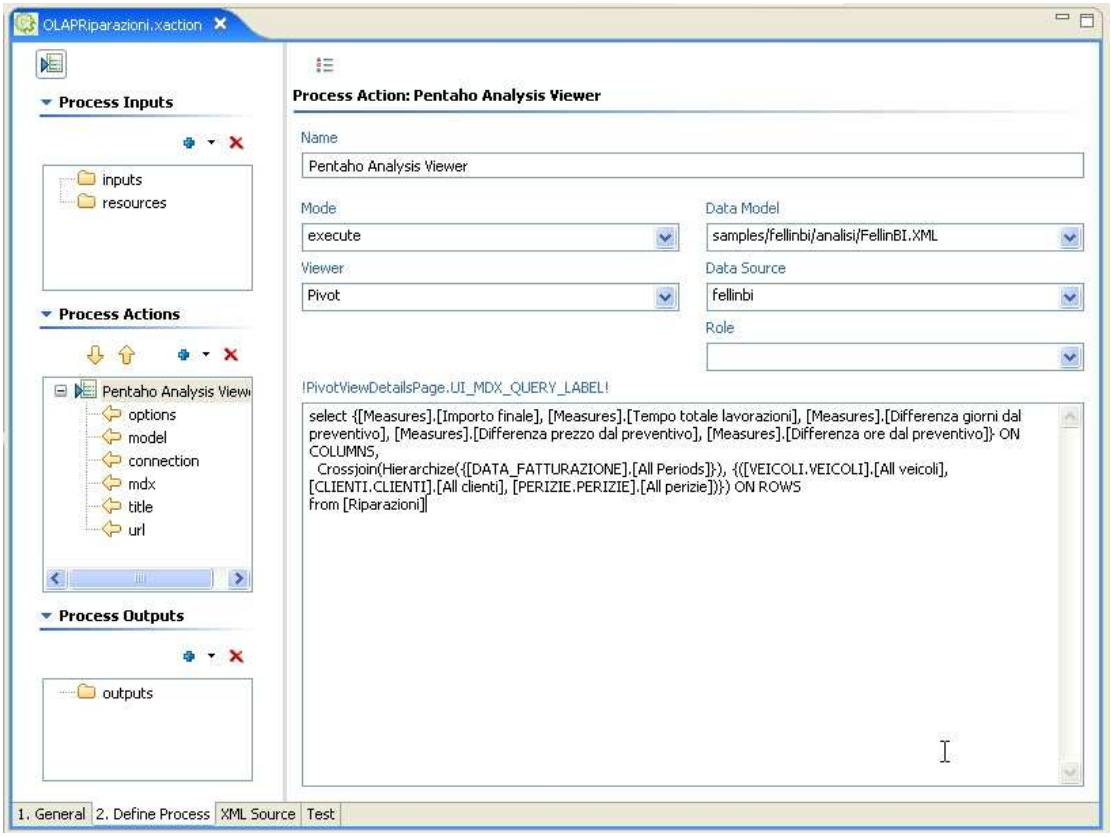
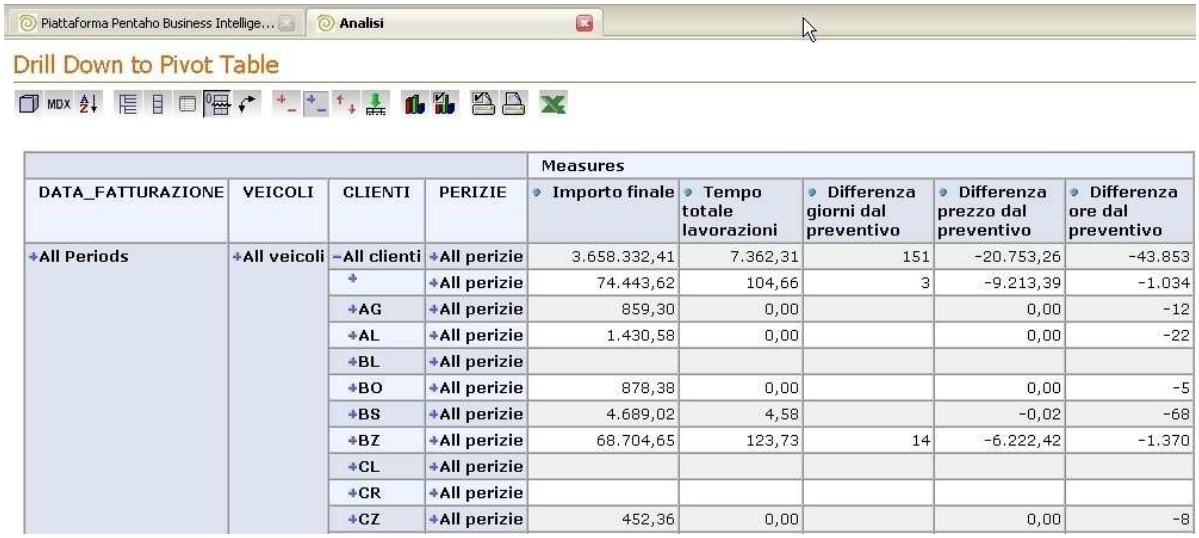


Illustrazione 117: Definizione dell'analisi OLAP

Non esiste al momento una procedura guidata per la selezione delle dimensioni e i fatti del cubo da visualizzare, per cui è necessario scrivere a mano la query MDX che reperisce i dati iniziali:

```
select {[Measures].[Importo finale], [Measures].[Tempo totale
lavorazioni], [Measures].[Differenza giorni dal preventivo],
[Measures].[Differenza prezzo dal preventivo], [Measures].[Differenza ore
dal preventivo]} ON COLUMNS,
Crossjoin(Hierarchize({[DATA_FATTURAZIONE].[All Periods]}),
{([VEICOLI.VEICOLI].[All veicoli], [CLIENTI.CLIENTI].[All clienti],
[PERIZIE.PERIZIE].[All perizie])}) ON ROWS
from [Riparazioni]
```

Una volta lanciato l'applicativo vengono estratti i dati a seconda della query MDX iniziale ed è possibile esplorare il cubo multidimensionale usando lo strumento JPivot:



				Measures				
DATA_FATTURAZIONE	VEICOLI	CLIENTI	PERIZIE	Importo finale	Tempo totale lavorazioni	Differenza giorni dal preventivo	Differenza prezzo dal preventivo	Differenza ore dal preventivo
All Periods	All veicoli	All clienti	All perizie	3.658.332,41	7.362,31	151	-20.753,26	-43.853
			All perizie	74.443,62	104,66	3	-9.213,39	-1.034
		AG	All perizie	859,30	0,00		0,00	-12
		AL	All perizie	1.430,58	0,00		0,00	-22
		BL	All perizie					
		BO	All perizie	878,38	0,00		0,00	-5
		BS	All perizie	4.689,02	4,58		-0,02	-68
		BZ	All perizie	68.704,65	123,73	14	-6.222,42	-1.370
		CL	All perizie					
		CR	All perizie					
		CZ	All perizie	452,36	0,00		0,00	-8

Illustrazione 118: Documento analitico in esecuzione

Durante la realizzazione dell'Action Sequence, come negli altri casi finora svolti, si è sentita la mancanza di strumenti guidati per la compilazione dei valori (di fatto i menù di scelta a tendina non funzionano e non tutti i valori sono facilmente intuibili), mentre la documentazione non è esaustiva in questo senso. Questa lacuna, unita alla necessità di ricaricare manualmente l'elenco degli applicativi registrati nella piattaforma ha allungato notevolmente i tempi di sviluppo.

5.6 Implementazione di SpagoBI

5.6.1 Installazione

5.6.1.1 Installazione lato server

Engineering Ingegneria Informatica mette a disposizione un comodo installer per la sua piattaforma SpagoBI. Possono venire installati

tutti o solo alcuni dei moduli a disposizione, che vengono dapprima scaricati sulla cartella di lavoro. Il procedimento è agevole e ben documentato dalla guida “SpagoBI Installation Manual”.

Come ambiente di esecuzione è richiesta la presenza di un Server Web già installato nel sistema con estensioni eXo-Portal a scelta tra Tomcat, JBoss, JonAS e IBM Web Sphere[14].

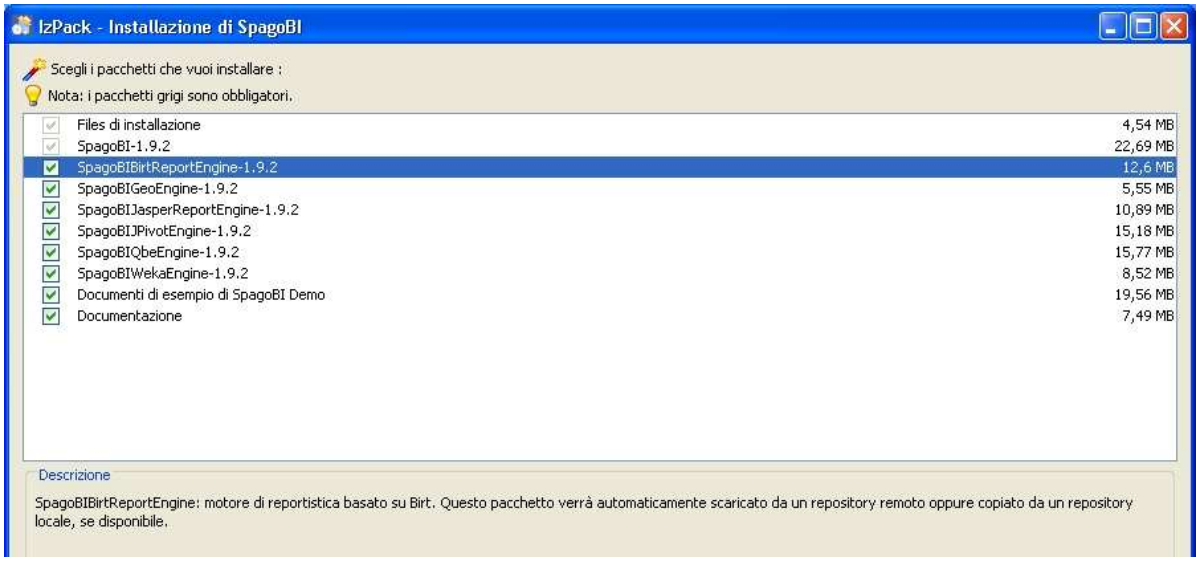


Illustrazione 119: Selezione dei moduli da installare

Durante la fase di installazione è possibile configurare la connessione all'RDBMS del DataWarehouse. I driver disponibili sono quelli per HSQLDB, Oracle, MySQL e Postgre SQL. Per connessioni a database diversi, come nel nostro caso che utilizzeremo Firebird, è necessario intervenire a installazione ultimata sui file di configurazione come riportato nel paragrafo successivo.

Sia il programma di installazione che i moduli sono compilati in Java, rendendo la procedura univoca nelle piattaforme Windows, Linux e Macintosh.

Al termine della procedura la piattaforma SpagoBI viene caricata sul

14 Lo sviluppo basato su protlet JSR-168 consente comunque di installare SpagoBI teoricamente su qualunque altro portal container che aderisca a tale standard.

Server Web precedentemente installato e può essere avviata dal collegamento sul menù Start oppure come servizio Windows NT.

5.6.1.2 Configurazioni iniziali

Come prima operazione si è provveduto a registrare la connessione JNDI[15] al database Firebird procedendo come segue:

- copiare il driver JDBC nella cartella del server /exo-portal/common/lib
- definire il datasource JNDI aggiungendo tra i tag <GlobalNamingResorce> e </GlobalNamingResorce> del file /expo-portal/conf/server.xml il seguente frammento:

```
<Resource name="jdbc/spagobi" auth="Container"
type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/spagobi">
      <parameter>
            <name>factory</name>
            <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
      <parameter>
            <name>driverClassName</name>
            <value>org.hsqldb.jdbcDriver</value>
      </parameter>
      <parameter>
            <name>url</name>
            <value>jdbc:hsqldb:hsql://localhost:9002/spagobi</value>
      </parameter>
      <parameter>
            <name>username</name>
            <value>sa</value>
      </parameter>
      <parameter>
```

15 La tecnologia JNDI consente di definire connessioni a sorgenti dati, identificandole con un nome univoco che può essere richiamato dagli applicativi Java per condividere connessioni comuni, senza definire di volta in volta i parametri di accesso.

```
            <name>password</name>
            <value></value>
      </parameter>
      <parameter>
            <name>maxActive</name>
            <value>20</value>
      </parameter>
      <parameter>
            <name>maxIdle</name>
            <value>10</value>
      </parameter>
      <parameter>
            <name>maxWait</name>
            <value>-1</value>
      </parameter>
</ResourceParams>
```

- registrare la connessione JNDI nella piattaforma SpagoBI aggiungendo al file /exo-portal/webapps/spagobi/WEB-INF/conf/data-access.xml il frammento:

```
<CONNECTION-POOL connectionPoolName="fellinbi" connectionPoolFactoryClass
=     "it.eng.spago.dbaccess.pool.JNDIConnectionPoolFactory"
      connectionPoolType="native"
      connectionDescription="FellinBI Database">
      <CONNECTION-POOL-PARAMETER parameterName="initialContext"
                  parameterValue="java:comp/env"/>
      <CONNECTION-POOL-PARAMETER parameterName="resourceName"
                  parameterValue="jdbc/fellinbi"/>
      <CONNECTION-POOL-PARAMETER parameterName="driverVersion"
                  parameterValue="2.1.1" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="sqlMapperClass"

      parameterValue="it.eng.spago.dbaccess.sql.mappers.OracleSQLMapper"
            parameterType=""/>
</CONNECTION-POOL>
```

- all'interno del tag <Context> del file /exo-

portal/conf/Catalina/localhost aggiungere:

```
<ResourceLink name="jdbc/fellinbi" type="javax.sql.DataSource"
global="jdbc/fellinbi" />
```

- aggiungere il frammento seguente

```
<CONNECTION name="fellinbi"
            isDefault="true"
            isJNDI="true"
            initialContext="java:comp/env"
            resourceName="jdbc/fellinbi"/>
```

ai vari moduli della piattaforma, in particolare nei file

- exo-home/webapps/SpagoBIJPivotEngine/WEB-INF/classes/connections-config.xml (per JPivot)
- exo-home/webapps/SpagoBIJasperReportEngine/WEB-INF/classes/engine-config.xml (per JasperReport)
- exo-home/webapps/SpagoBIBirtReportEngine/WEB-INF/classes/engine-config.xml (per BIRT)
- exo-home/webapps/SpagoBIGeoEngine/WEB-INF/conf/spago/data-access.xml (per Weka)
- exo-home/webapps/SpagoBIQbeEngine/WEB-INF/conf/data-access.xml (per QbE)
- exo-home/webapps/SpagoBIGeoEngine/WEB-INF/conf/spago/data-access.xml (per GeoEngine)

- per ogni componente di SpagoBI sopra elencato registrare la connessione JNDI nel relativo file /exo-portal/webapps/<applicazione>/WEB-INF/web.xml aggiungendo nei tag <webapp></webapp> aggiungendo:

```
<resource-ref>
     <description>fellinbi</description>
```

```
      <res-ref-name>jdbc/fellinbi</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
</resource-ref>
```

Configurata la connessione al database si passa all'aggiunta del nuovo utente fellinbi e la personalizzazione dell'interfaccia del portale. L'operazione avviene completamente tramite le pagine web della piattaforma, entrando con l'utente amministrativo exoadmin. Qui è possibile personalizzare anche i gruppi utente per consentire una gestione flessibile e completa dei diritti di accesso. Ogni utente inoltre può avere a disposizione un ambiente personalizzato grazie all'editor delle interfacce e dei menù di navigazione.

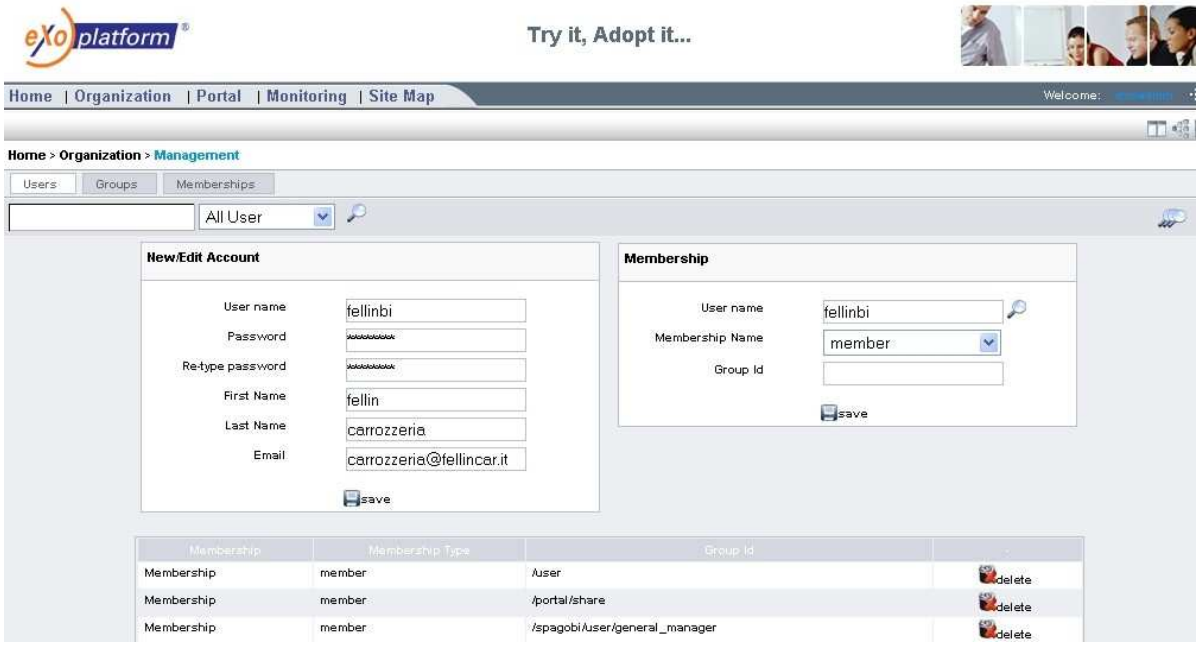


Illustrazione 120: Definizione di un nuovo utente

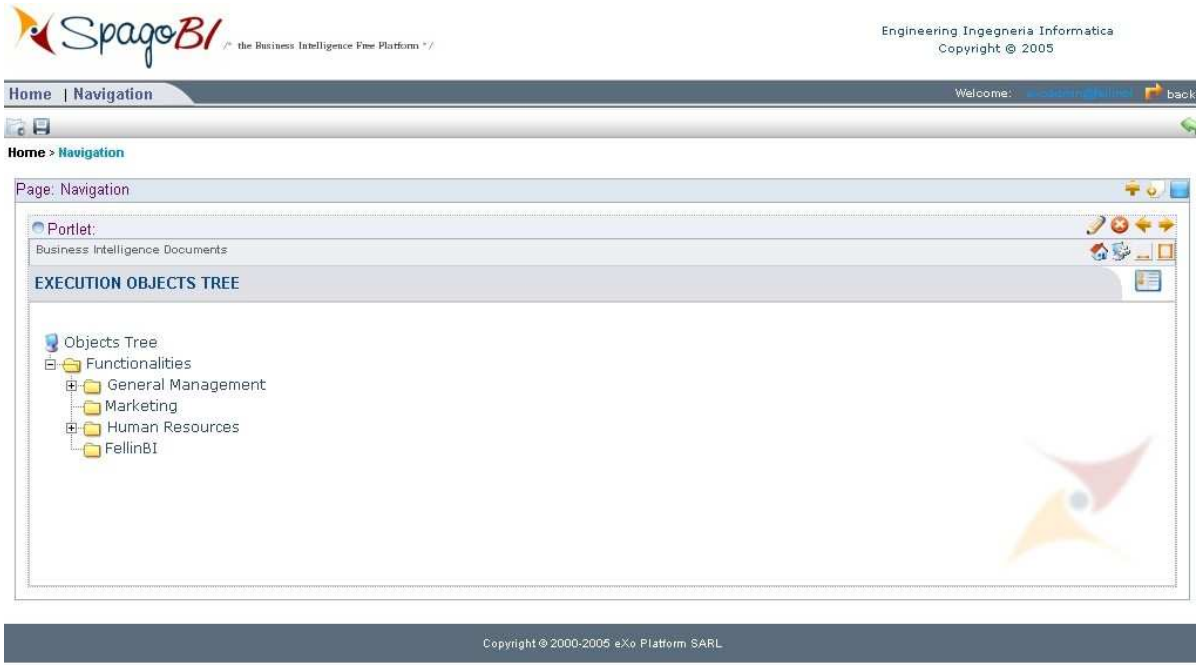


Illustrazione 121: Personalizzazione interfaccia utente

Nel nostro progetto è stata tolta la schermata iniziale predefinita di SpagoBI che elenca tutti gli utenti e le rispettive password. Inoltre per l'utente fellinbi è stata lasciata la sola sotto cartella fellinbi della repository dei documenti analitici. Considerando che l'architettura è basata sull'uso di Portlet JSR-168 è consentito l'utilizzo di componenti di terze parti raggiungendo un elevato grado di personalizzazione.

Per la traduzione delle pagine in italiano si va a modificare il file spagobi.xml contenuto nella cartella \exo-portal\webapps\spagobi\WEB-INF\conf\spagobi aggiornando il tag <LANGUAGE_SUPPORTED> come segue:

```
<LANGUAGE_SUPPORTED>
      <LANGUAGE default="true" language="it" country="IT" />
      <LANGUAGE default="false" language="en" country="US" />
      <LANGUAGE default="false" language="fr" country="FR" />
</LANGUAGE_SUPPORTED>
```

Nuove lingue possono essere aggiunte creando un file di traduzione che va registrato all'interno della piattaforma seguendo le indicazioni della documentazione ufficiale.

5.6.1.3 Installazione lato client

L'unico requisito dei client è un browser web di ultima generazione. Per stampare i report su PDF e per i cruscotti basati su Flash è necessario installare i relativi plugin Adobe Reader e Adobe Flash Player. Non è richiesta altra configurazione aggiuntiva, se non l'aggiunta del link web alla piattaforma SpagoBI che è del tipo

```
http://<nomeserver>:8080/sbiportal
```

5.6.2 ETL per il caricamento dei dati

Kettle, software realizzato dalla Pentaho stessa, è fortemente integrato nella piattaforma PentahoBI. La controparte per SpagoBI è Talend Open Studio, le quali procedure ETL possono venire importate ed eseguite dalla piattaforma.

Direttamente da Talend Open Studio è possibile configurare il collegamento al server dove gira SpagoBI, il tutto tramite il menu Window -> Preferences e sulla mascherina che appare scegliendo la voce Talend -> SpagoBI Server. Questa videata consente di definire nuovi collegamenti a SpagoBI semplicemente cliccando sul pulsante New e compilando le informazioni come nella seguente figura:

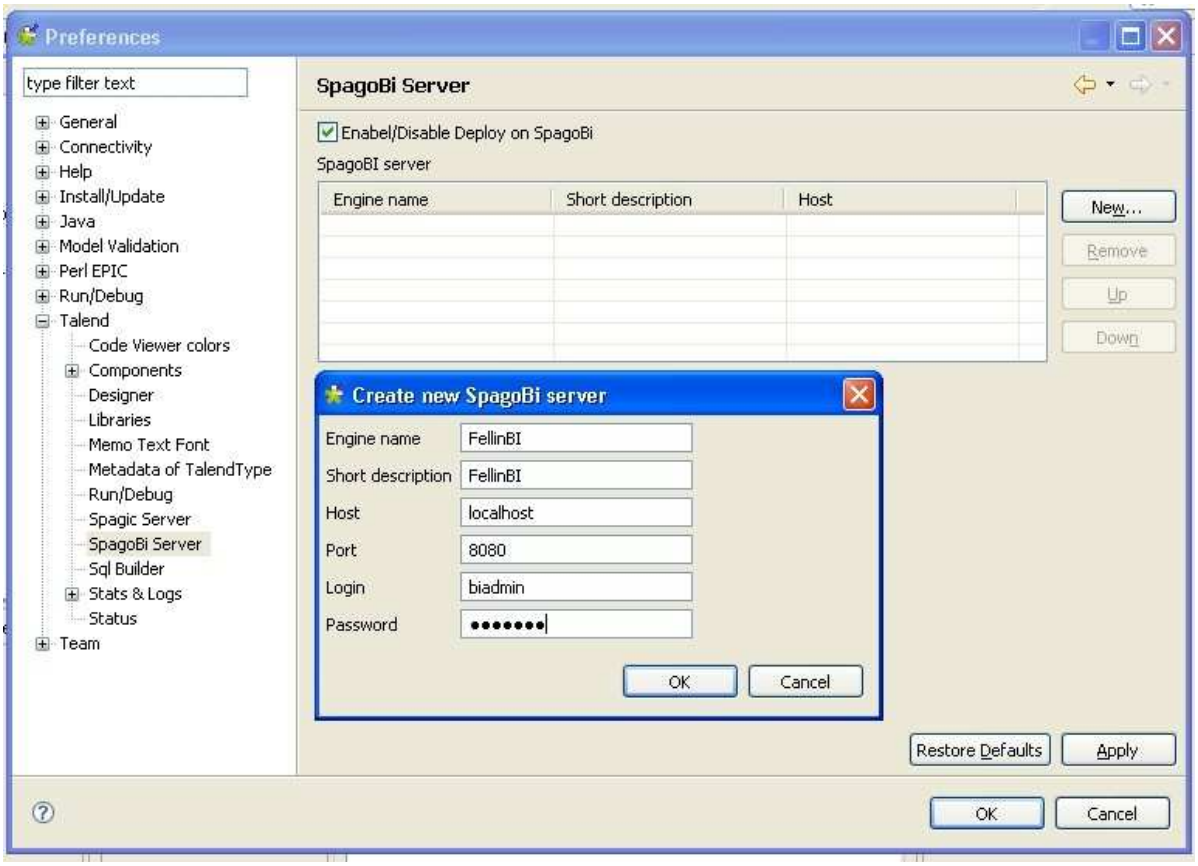


Illustrazione 122: Registrazione del server SpagoBI

Una volta confermate le nuove impostazioni, cliccando con il pulsante destro del mouse sull'elenco “Job Designs” apparirà la voce “Deploy on SpagoBI“. La mascherina di Illustrazione 123 consentirà quindi di installare il Job all'interno delle piattaforma SpagoBI.

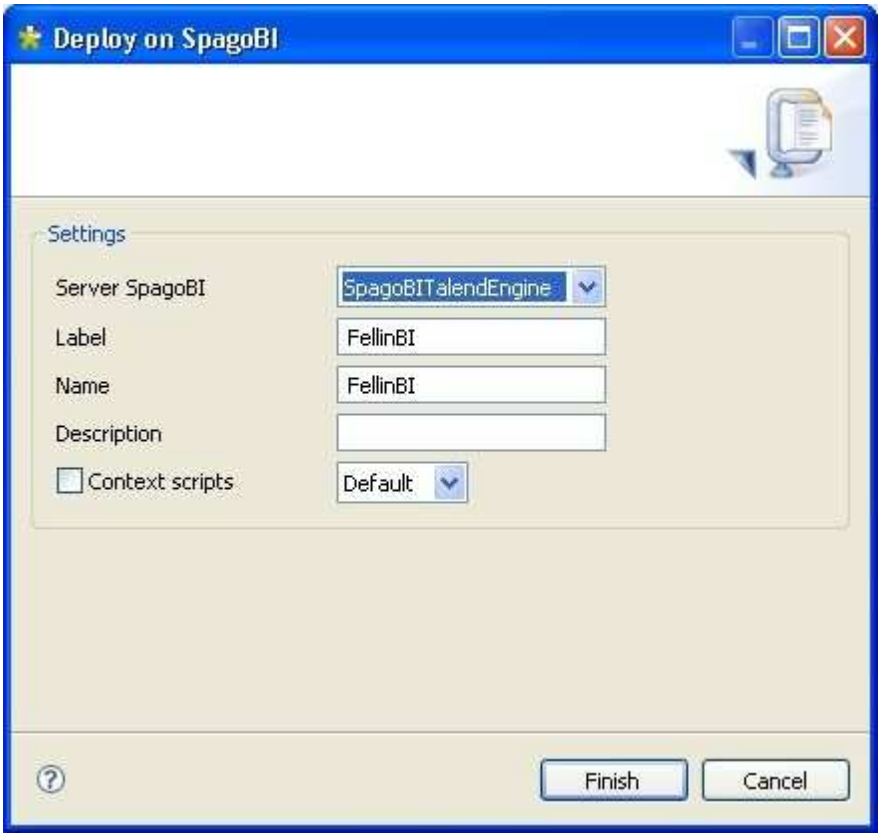


Illustrazione 123: Installazione di un job in SpagoBI

Nel nostro caso è stata installata la procedura FellinBI che raggruppa gli aggiornamenti per le tabelle Riparazioni, Clienti, Periti e Veicoli. Resta escluso dalla procedura FellinBI il job FellinBIDimensioneData per l'aggiornamento della tabella Dimensione_data, in quanto costituisce un'operazione che viene eseguita una tantum dall'utente.

Con questi passaggi Talend Open Studio prepara un pacchetto con la procedura ETL e lo installa nella cartella del server. Viene inoltre copiato un file XML con le informazioni necessarie alla registrazione della procedura nella piattaforma. Quello che rimane ora è la creazione di un nuovo documento analitico che mandi in esecuzione i job direttamente da SpagoBI, operazione questa che viene agevolata dalla presenza di maschere guidate direttamente tra le pagine della piattaforma. Solitamente la pubblicazione di nuovi

documenti analitici si riduce alla compilazione dei seguenti campi:

- nome e descrizione del documento analitico
- file template generato dallo strumento di design (ad esempio Open Studio, iReport, BIRT, eccetera)
- eventuali parametri da richiedere all'utente prima dell'esecuzione del processo
- repository dei documenti analitici dove pubblicare l'applicativo

Per registrare la procedura di ETL, una volta che è stata esportata da Talend Open Studio, si deve semplicemente accedere con i diritti di sviluppatore all'elenco dei documenti analitici di Illustrazione 124 per poi cliccare sull'icona a forma di lente di ingrandimento.

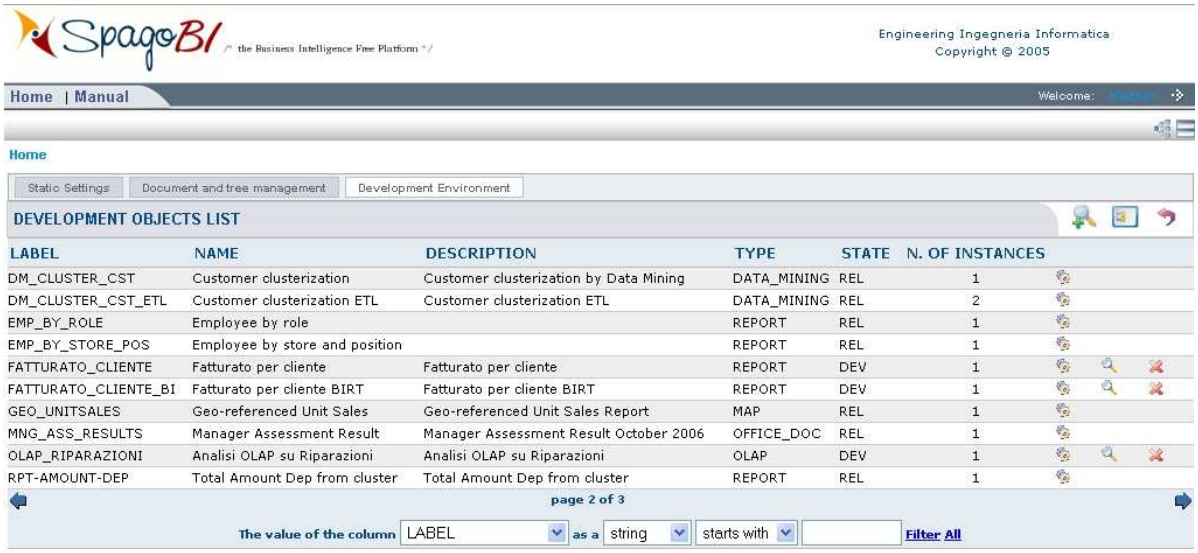


Illustrazione 124: Elenco documenti analitici in SpagoBI

Apparirà una mascherina da completare come nella seguente figura:

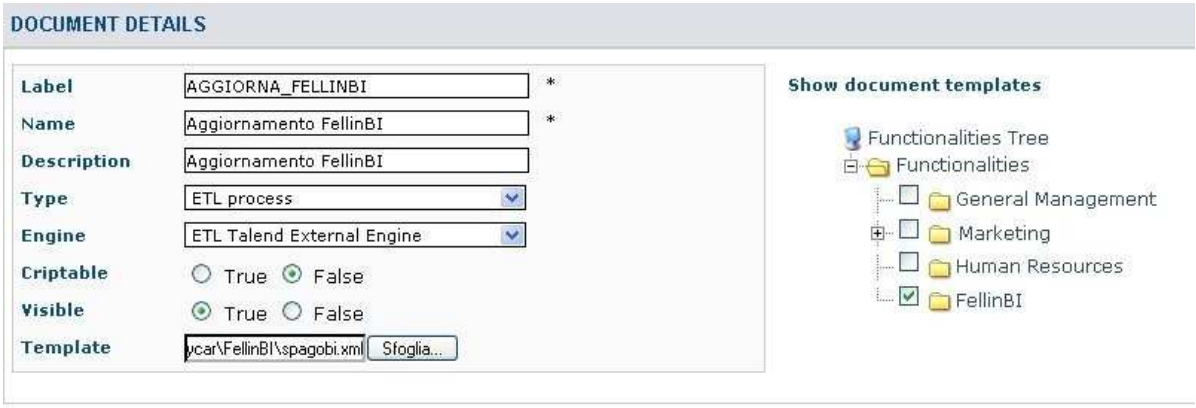


Illustrazione 125: Registrazione della procedura ETL FellinBI

Il file Template è il file XML generato da Open Studio, che nell'installazione predefinita viene salvato in /exo-portal/webapps/SpagoBITalendEngine/RuntimeRepository/mycar.

La stessa operazione viene ripetuta per registrare la procedura FellinBIDimensioneData, creando un nuovo documento con le seguenti impostazioni:

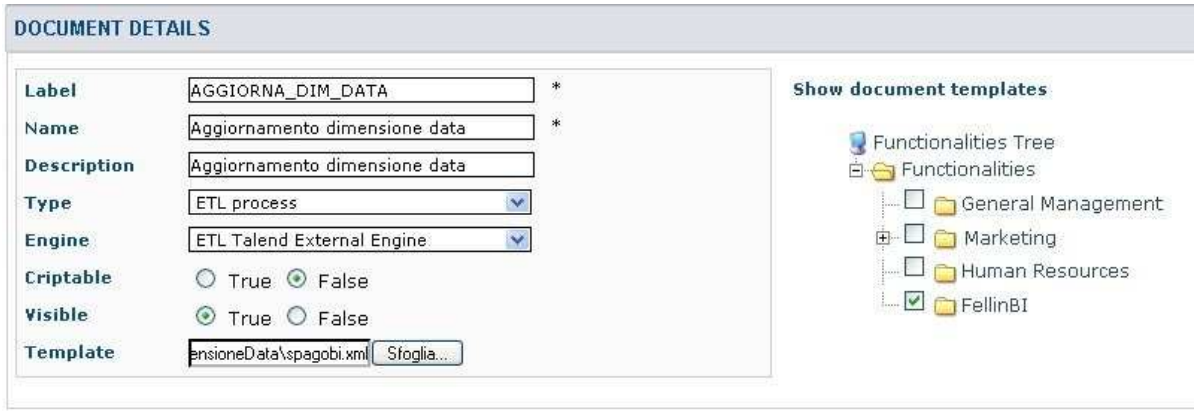


Illustrazione 126: Registrazione della procedura ETL FellinBIDimensioneData

I documenti così creati rimangono nello stato development, solitamente nascosti ai comuni utenti ma accessibili agli sviluppatori per collaudarne le funzionalità. In SpagoBI nella maschera ”Documents Configuration” di “Document and tree management“ vengono elencati tutti i documenti analitici registrati

e cliccando sull'icona a forma di lente di ingrandimento appare una maschera simile a quella vista per la registrazione di nuovi documenti. Con questa maschera è inoltre possibile cambiare lo stato dei documenti tra le opzioni:

- Suspended
- Development
- Test
- Released

Impostato lo stato a Released anche l'utente fellinbi potrà eseguire l'applicativo direttamente dalla repository FellinBI di Illustrazione 121.

Oltre all'esecuzione dei documenti analitici ETL dalla piattaforma è consigliabile installare e configurare lo schedulatore cron in modo che esegua queste procedure a intervalli prefissati, nel nostro caso ogni 12 ore. Come discusso nel paragrafo 5.2.2.6 Talend Open Studio prevede l'apposita scheda Scheduler per la configurazione di cron attraverso un file batch che viene generato in fase di esportazione del job.

5.6.3 Reportistica

Per aggiungere un report, una volta creato con il rispettivo strumento di sviluppo come iReport (vedi paragrafo 5.4.1), si deve accedere con i diritti di sviluppatore alla videata che elenca i documenti analitici e si prosegue come nella registrazione delle procedure ETL. In questo caso la maschera viene configurata come nella seguente figura:

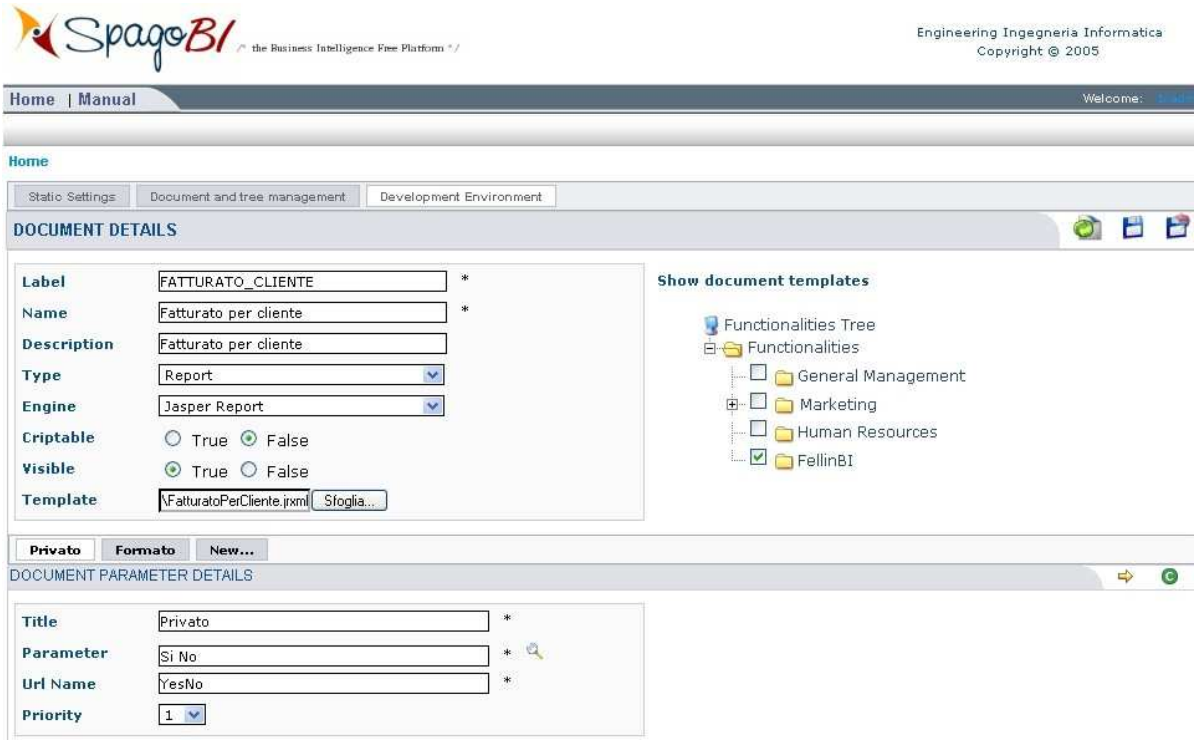


Illustrazione 127: Registrazione report in SpagoBI

Oltre ai dati di registrazione vengono aggiunti due parametri richiesti all'utente in fase di esecuzione. Uno è il parametro privato con cui filtrare il report e l'altro è il formato di stampa, a scelta tra HTML, PDF, XLS, CVS, TXT, XML e JPG.

Illustrazione 128: Parametri chiesti all'utente durante l'esecuzione

Mentre il parametro formato era già disponibile nella piattaforma,

per aggiungere il parametro privato con valori a scelta Si/No si deve prima definire un nuovo elenco di valori “List Of Values” (LOV) che andrà a costituire un nuovo parametro dell'elenco parametri.

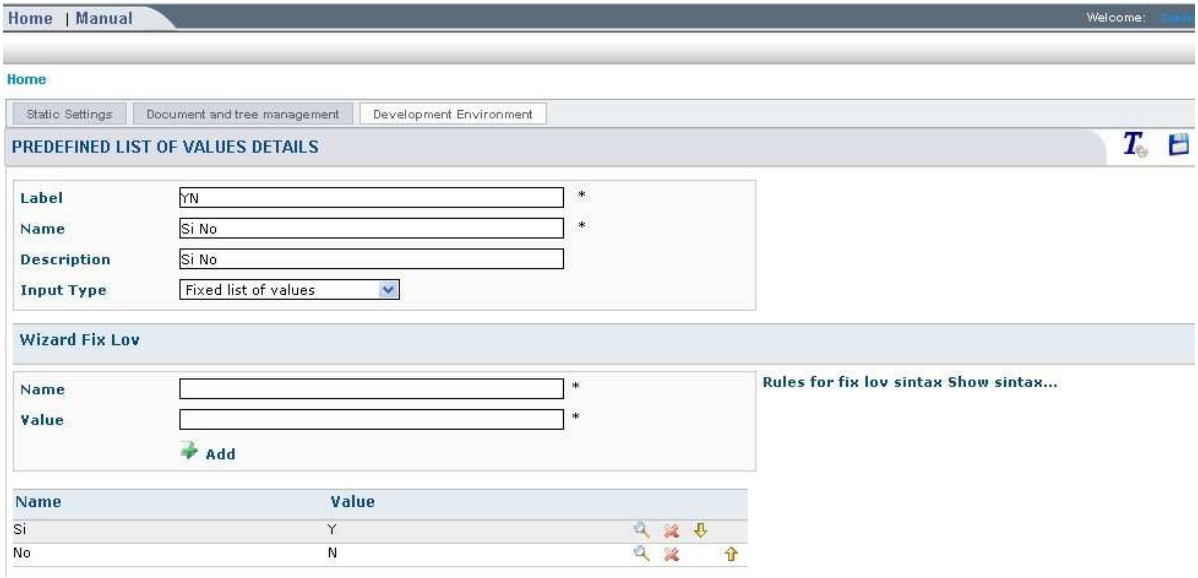


Illustrazione 129: Definizione di un elenco di valori (LOV)

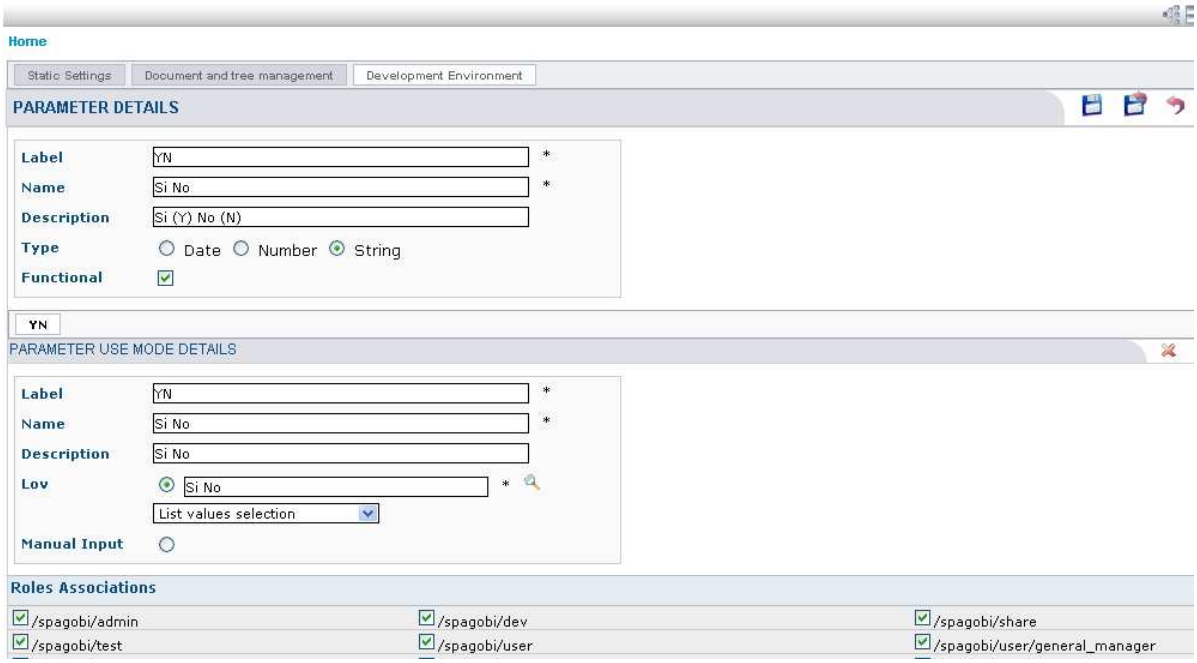


Illustrazione 130: Definizione di un parametro in SpagoBI

La scelta progettuale di creare nuovi parametri dalla combinazione

di elenchi di valori (List Of Values) e criteri di validazione (Predefined Values Constraints) consente un elevato riutilizzo di funzioni e una elevata manutenibilità nei possibili sviluppi futuri. Ad ogni parametro è anche possibile attribuire permessi e modificarne il comportamento a seconda dell'utente che accede alla piattaforma.

Il documento analitico realizzato compare nella cartella fellinbi della repository e durante l'esecuzione, una volta compilata la form che chiede l'inserimento dei parametri, il report appare come segue:

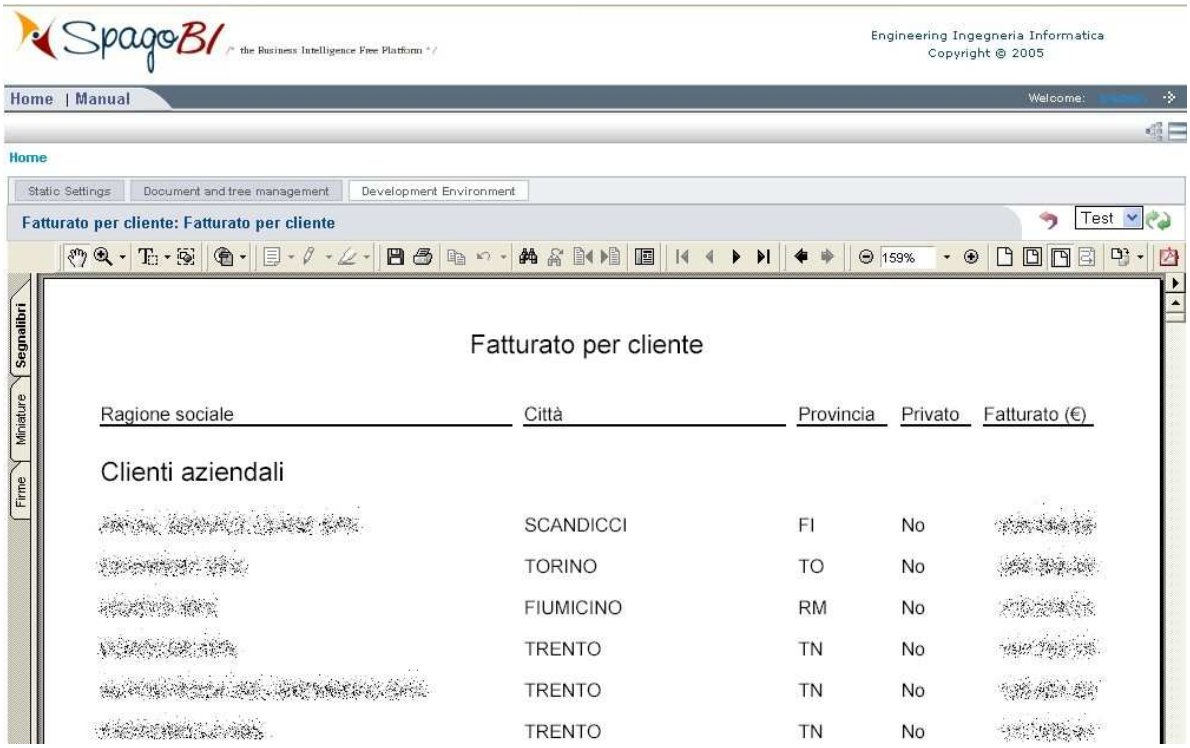


Illustrazione 131: Esecuzione del report Fatturato per cliente

Un risultato simile si ottiene implementando il report con BIRT di cui si tralasciano i passaggi che sono gli stessi della variante JasperReport.

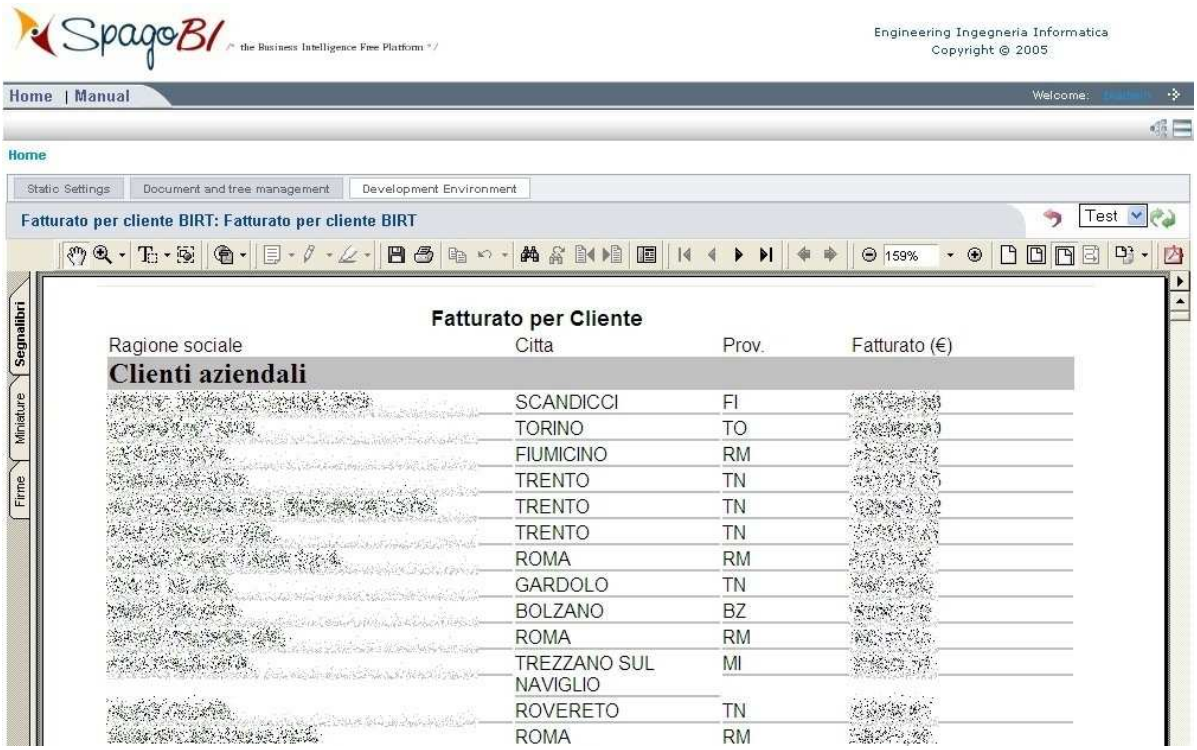


Illustrazione 132: Report Fatturato per cliente realizzato con BIRT

5.6.4 Analisi OLAP

L'analisi OLAP richiede come primo passaggio la copiatura del file schema Mondrian definito nel paragrafo 5.3 all'interno della cartella \exo-portal\webapps\SpagoBIJPivotEngine\WEB-INF\queries.

Come tutti i documenti analitici, SpagoBI fornisce una semplice interfaccia per registrare nuovi applicativi. Nel caso di applicativi OLAP la registrazione avviene nella videata seguente:

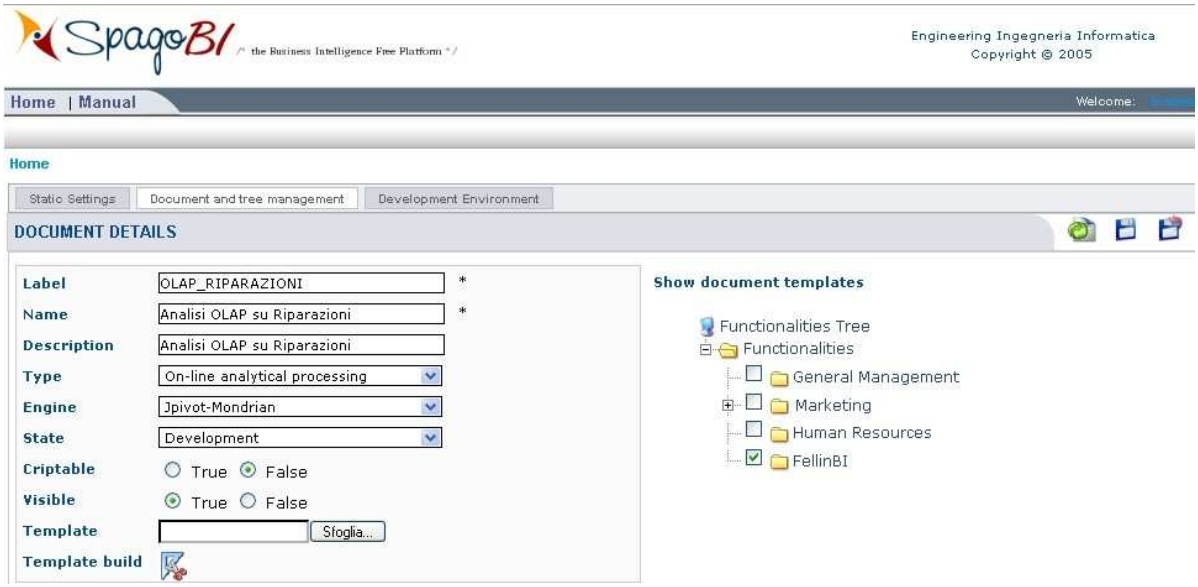


Illustrazione 133: Registrazione dell'analisi OLAP

Registrate le informazioni generali del documento analitico rimane da definire la query MDX per l'interrogazione del cubo Riparazioni. Con le recenti versioni di SpagoBI è a disposizione all'interno delle stesse videate uno strumento per scegliere gli elementi da visualizzare del cubo e generare la query MDX in automatico.

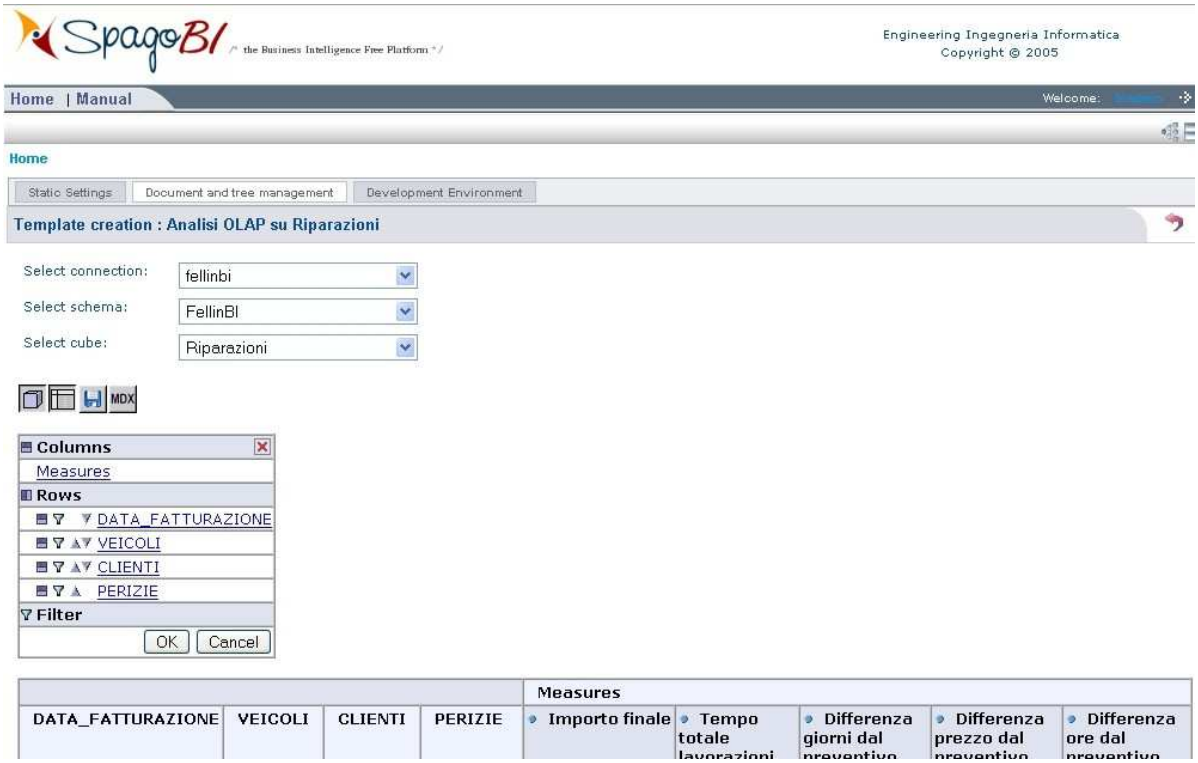


Illustrazione 134: Definizione della query per l'analisi OLAP

Rimane comunque possibile intervenire direttamente sulla query per personalizzare l'analisi in base alle proprie esigenze. Una volta salvato il documento analitico il risultato è il seguente:

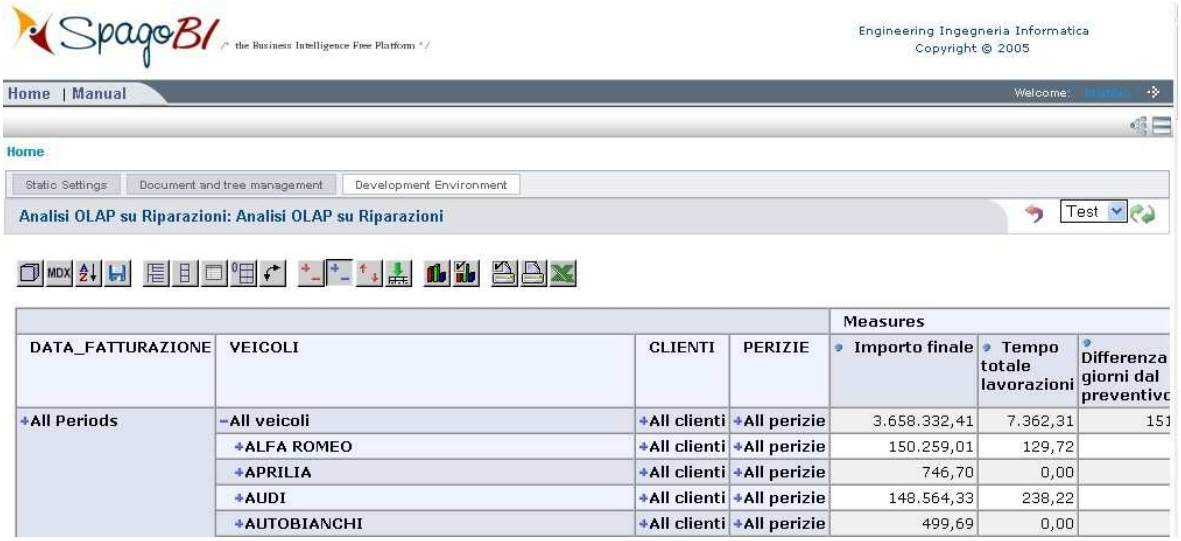


Illustrazione 135: Esecuzione dell'analisi OLAP

6 Breve comparativa

L'analisi comparativa fra due piattaforma di Business Intelligence è di per se un argomento complesso e delicato. Molti sono i fattori da tenere in considerazione come l'architettura software, le prestazioni, la metodologia di sviluppo, il supporto tecnico, eccetera. La cosa si complica ancor di più dato che le due piattaforme sono il risultato di un insieme di altri software e che ogni valutazione comparativa andrebbe supportata da risultati oggettivi.

Considerando il fatto che le due piattaforme non sono state esaminate in tutte le loro funzionalità e considerando che l'obiettivo finale è valutare la qualità di due ambienti di Business Intelligence, tali valutazioni comparative rimangono generali senza per questo pregiudicare il lavoro svolto.

6.1 Architettura software

Le due piattaforme sono entrambe sviluppate con il linguaggio di programmazione Java, funzionano su web server simili e condividono molti componenti software Open Source come Mondrian/JPivot, Weka, BIRT, eccetera. Ciononostante l'architettura di base delle due piattaforme e il modo con cui i componenti software sono integrati è decisamente diverso. Mentre Pentaho BI tende a fondere assieme gli strumenti che utilizza, Spago BI preferisce integrarli creando interfacce software come ad esempio drivers. Spago BI funziona interamente in modalità a portale facendo uso del framework applicativo Spago e l'ambiente eXo Portal. Pentaho BI invece ha come interfaccia predefinita quella a sito classico, pur consentendo in JBoss una modalità a portale poco

documentata e utilizzata.

Le conseguenze più rilevanti dal punto di vista dello sviluppo sono un maggior riutilizzo di codice ed estensibilità per Spago BI e un maggior controllo e integrazione per Pentaho BI.

Caratteristica	Pentaho BI	SpagoBI
Application Server	A scelta fra Tomcat[16] e JBoss come installazione di default	A scelta fra Tomcat, JBoss, JonAS, IBM WebSphere, Tomcat è l'installazione di default
Portal Server	JBoss Portal, non richiesto nella modalità a interfaccia classica di default	Exo Portal, IBM WebSphere Portal o teoricamente qualunque altro portal container con supporto a JSR-168
Application Framework	Nessuno in particolare	Spago
Repository Contenuti	Nessuno in particolare	Jackrabbit, Exo JCR o qualunque altro repository conforme allo standard JSR-170
Motore Collaboration	Non implementato	jBPM per il Workflow, OpenOffice Impress per i dossier analitici
Reportistica	JasperReport, BIRT, JFreeReport	JasperReport, BIRT,Business Objects (WebI)

16 Su Tomcat Pentaho BI non consente di eseguire analisi OLAP a causa di errori interni, errori che rimangono irrisolti dalla release ufficiale 1.2 fino all'ultima release 1.5.4 Milestone 4

Motore/client OLAP	Mondrian/JPivot	Mondrian/JPivot, XMLA Server/JPivot
Motore di Data Mining	Weka[17]	Weka
GIS	SVG, Interfacciamento con Google Maps	CartoWeb, MapServer, SVG
Supporto QbE	Assente	Strumento basato su Hibernate
Software di ETL proposto	Pentaho Data Integration (meglio conosciuto come Kettle)	Talend Open Studio
Motore Workflow	Shark	JBPM
Schedulatore	Quartz	Quartz
Grafici	JFreeChart, XML/SWF Charts	OpenLaszlo
Altri software usati	Nessuno in particolare	OpenOffice Impress

6.2 Funzionalità

La seguente tabella riassume le funzionalità implementate, indicando con la + se sono presenti, x se sono assenti oppure con il rettangolo grigio se sono incomplete. Della piattaforma PentahoBI non viene considerata l'interfaccia a portale che tra l'altro ha le stesse funzioni se non per una gestione degli accessi utente leggermente migliore.

17 Non ancora integrato nella piattaforma

Funzionalità	SpagoBI	PentahoBI 1.2	PentahoBI 1.5.4 (testing)
Grafici			
Analisi OLAP			
Report			
GIS			
Data Mining			
Query-by-example			
Login e profili accesso utente			
Ciclo approvazione documenti e testing			
Salvataggio analisi OLAP svolte			
Schedulazione esecuzione documenti analitici			

Come si può notare la piattaforma SpagoBI implementa un maggior numero di funzionalità accessorie come un buon meccanismo di permessi utente e testing degli applicativi. In Pentaho BI alcune caratteristiche sono ancora in fase di implementazione come ad esempio il Data Mining. Entrambi i software permettono di realizzare i più comuni documenti analitici.

6.3 Supporto e documentazione

Al momento della redazione della tesi la documentazione principale di Spago BI è costituita dalle guide QuickStart v.0.9.3 e Spago BI How To v.1.7 che si dimostrano un valido strumento per iniziare a lavorare nonostante ci siano paragrafi in fase di redazione. L'interfaccia grafica di Spago BI, dopo aver preso confidenza con la particolare disposizione degli elementi, risulta sufficientemente chiara e intuitiva per compiere molte operazioni anche senza documentazione a supporto.

Chi volesse personalizzare il login e le pagine utente deve accedere alla modalità amministrativa di eXo Portal, di cui esiste una ricca documentazione nel sito www.exoplatform.com.

Lo sviluppatore intenzionato ad estendere la piattaforma può cominciare con la guida SpagoBI Architecture, per poi accedere alle risorse che si trovano nel sito di SpagoBI come il bug tracking system Atlassian Jira e il source code repository basato su Subversion.

Completa il supporto la presenza di un forum e mailing list per mettersi a contatto con gli altri sviluppatori e tecnici. Per gli strumenti esterni come il software di reportistica BIRT o JasperReport, gli schemi Mondrian, i grafici OpenLaszlo eccetera si dovrà consultare la documentazione nei rispettivi siti. Durante lo sviluppo degli applicativi e l'implementazione della piattaforma Spago BI la documentazione si è dimostrata adeguata alle esigenze e competenze di un tipico sviluppatore per la Business Intelligence.

Per quanto riguarda la piattaforma Pentaho BI è a disposizione una

quantità numerosa di documenti suddivisi per argomento e quasi tutti identificati con il numero di versione 1.5.4. Per l'amministrazione di sistema, anche se si decide di usare l'applicativo auto installante, è utile consultare la guida “Manual Deployment of Pentaho” per avere un'idea di quali siano i file di configurazione più importanti. Una conoscenza del web server su cui si installerà la piattaforma può agevolare di molto le fasi di amministrazione, dati i frequenti accessi al file system del server e ai file di configurazione in esso contenuti.

Lo sviluppatore di documenti analitici può orientarsi con le guide “Getting Started with the BI Platform”, “Creating Pentaho Solutions” e “Pentaho Integrating Birt and Jasper”. La creazione di dashboards rimane un'operazione per programmatori esperti anche consultando il documento “Pentaho Dashboard Building” che illustra i concetti essenziali nell'uso di Action Sequence, codice JSP, JavaScript e HTML per realizzare cruscotti. Ai sviluppatori della piattaforma è infine dedicata la guida “Pentaho Building Components”, il bug tracking system Atlassian Jira e la repository delle sorgenti basta su Subversion.

La documentazione presenta spesso paragrafi in via di redazione e solo in alcuni casi si è dimostrata completa. La traduzione in italiano, poi, risale a versioni molto vecchie di Pentaho e costringe a consultare la versione inglese. Anche gli esempi di documenti analitici a corredo risultano talvolta disallineati con le ultime versioni rilasciate, ad esempio i grafici vengono realizzati senza l'uso di Action Sequence bensì con file XML non documentati.

L'utente che non trova le informazioni necessarie nella guida può cercare supporto nel forum di Pentaho.

In conclusione la documentazione e il supporto di SpagoBI risulta di per sé soddisfacente anche se rimangono delle parti da completare. La maturità di SpagoBI e le maschere autoguidate poi consentono di prendere familiarità con l'ambiente in tempi piuttosto rapidi. Per PentahoBI invece lo sviluppatore si trova spesso privo di un soddisfacente supporto informativo che unito a una piattaforma poco intuitiva e non completamente stabile lasciano all'utente solo le alternative del forum o l'acquisto del supporto tecnico.

6.4 Soddisfazione dei requisiti

Nel paragrafo 4.5 è stato stilato un elenco certamente non esaustivo dei requisiti per una piattaforma di Business Intelligence. Trattandosi di parametri non oggettivi è possibile dare un confronto sommario del loro raggiungimento nei due software:

requisito	valutazione
Ambiente user-friendly	Sia SpagoBI che PentahoBI pongono particolare rilevanza a tale aspetto, soprattutto nell'interfaccia rivolta all'utente finale. SpagoBI ha una interfaccia un po' particolare nelle pagine del portale, che richiede un breve ambientamento iniziale ma risulta sicuramente efficace
Possibilità di variare le analisi di base, applicando filtri o modificando la prospettiva	Sia SpagoBI che PentahoBI prevedono come soluzione la parametrizzazione dei documenti analitici
Frequenza di aggiornamento	Dipende dal software di ETL usato.

adeguata e trasparente	Sia Kettle sia Talend Open Studio consentono l'aggiornamento schedulato e trasparente all'utente
Disponibilità di strumenti di Business Intellingence potenti e flessibili (reportistica, Data Mining, analisi OLAP, ecc.)	Entrambe le piattaforme hanno una ricca gamma di software analitici. Il Data Mining è disponibile in SpagoBI e in via di implementazione in PentahoBI
Tempi ragionevoli nell'elaborazione dei dati	Trattandosi di soluzioni Java, i tempi di esecuzione sono relativamente lunghi ma pur accettabili. Nella navigazione tra le pagine e gli applicativi si è notata una migliore reattività di SpagoBI rispetto a PentahoBI
Visione chiara e completa in un numero ridotto di schermate	Entrambe le piattaforme sono curate dal punto si vista della navigazione tra le pagine
Possibilità di stampare i risultati e esportarli in formati comuni, come fogli Excel o file pdf	Questo dipende dal software usato, ma in linea generale i documenti analitici consentono tali funzionalità
Possibilità di salvare l'analisi OLAP svolta e riprenderla in seguito	E' previsto in SpagoBI, non in PentahoBI
Restrizione d'accesso a persone autorizzate	La gestione degli utenti in SpagoBI è molto buona, in PentahoBI è ancora immatura per essere applicata in ambienti che la richiedono
Integrazione dell'ambiente	Entrambe le piattaforme consentono

con altri software, come pacchetti di office automation	di esportare e importare dati verso formati Excel. Al contrario di alcuni software OLAP non c'è un interfacciamento diretto con Excel per l'esplorazione del cubo multidimensionale. SpagoBI consente di stampare il risultato delle analisi direttamente su un file Power Point.
Utilizzo di sistemi hardware/software già presenti o facilmente reperibili	Trattandosi di soluzioni Java la portabilità e la compatibilità hardware/software è molto elevata.
Percepire rapidamente situazioni anomali, con l'utilizzo ad esempio di sistemi di alert o cruscotti interattivi	Sono previsti cruscotti e la definizione di alert. In PentahoBI definire un cruscotto è un'operazione ancora difficile da realizzare, mentre gli alert per entrambe le piattaforme potrebbero essere resi più percepibili all'utente finale
Software robusto e affidabile	L'uso di ambienti ben collaudati come i web service JBoss e Tomcat, il linguaggio di programmazione Java e software quali JasperReport o Mondrian rendono le piattaforme molto affidabili una volta che i documenti analitici sono realizzati
Presenza di tools di configurazione che evitino la	In SpagoBI la modifica di file testuali è ridotta a particolari configurazione

modifica di file testuali	come l'aggiunta di una nuova connessione al database. In PentahoBI è frequente accedere a file xml sia durante la configurazione della piattaforma sia nella registrazione di nuovi documenti analitici
Documentazione chiara e completa	Per chi conosce l'inglese SpagoBI è sufficientemente adeguato sotto questo punto di vista. PentahoBI ha ancora molta documentazione incompleta, documentazione che tra l'altro è sparsa e aggiornata solo in alcune lingue
Supporto esterno rapido in caso di malfunzionamenti (forum, supporto tecnico, ecc.)	Sul sito di SpagoBI e di PentahoBI sono disponibili i rispettivi forum, ben frequentati dagli utenti e sviluppatori. Per entrambe le piattaforme è inoltre disponibile il supporto tecnico a pagamento
Tempi di installazione ragionevoli	E' disponibile sia per PentahoBI che per SpagoBI un comodo programma di installazione. Qualora si volesse installare gli ambienti manualmente la procedura diventa più complicata, ma pur sempre percorribile
Ridotta manutenzione dell'ambiente una volta installato	Una volta installate le due piattaforme richiedono pochissima manutenzione
Facilità nell'effettuare backup e ripristini	Il backup del database dipende dal software RDBMS usato, mentre per il

	backup e il ripristino della piattaforma è sufficiente copiare le cartelle di installazione
Tools user-friendly per creare applicativi	In SpagoBI si è raggiunto un buon livello di semplicità nella registrazione dei documenti analitici. In PentahoBI si sente spesso la mancanza di strumenti assistiti di compilazione, che verranno probabilmente implementati con la maturazione del codice
Facilità nell'installare e testare gli applicativi nella piattaforma	Per testare nuovi applicativi in PentahoBI bisogna registrare il documento rendendolo disponibile a tutti gli utenti e bisogna aggiornare di volta in volta la repository degli applicativi. Decisamente più completa è la gestione dei documenti analitici in SpagoBI, dove sono previste procedure di validazione dei documenti complete di diritti utente
Utilizzo di linguaggi di programmazione diffusi o comunque facilmente comprensibili	Lo scripting nei report segue la sintassi molto comune di JavaScript. Alcuni documenti analitici richiedono la conoscenza di XML e in PentahoBI di MDX

Facilità nell'apportare estensioni e modifiche agli applicativi già creati	In entrambi i casi per modificare i documenti analitici è sufficiente aprire i relativi strumenti di sviluppo
Possibilità di debugging degli applicativi sia in caso di malfunzionamenti una volta installati sia durante la realizzazione prima di pubblicarli	PentahoBI prevede il debugging dei documenti analitici con diversi livelli di dettaglio. SpagoBI non prevede il debugging degli applicativi se si esclude la stampa delle eccezioni Java, ma gli strumenti assistiti di compilazione dei documenti analitici risolvono agevolmente questa mancanza

Una ulteriore nota va a riguardo delle prestazioni, dove la macchina test si è dimostrata inadeguata quando venivano lanciati più applicativi contemporaneamente. Il Web Service Apache Tomcat dimostra per le scelte architetturali performance migliori a JBoss ma in PentahoBI la presenza di errori di esecuzione di analisi OLAP ne pregiudica l'adozione.

7 Conclusioni

Fin qui abbiamo esaminato senza entrare troppo nel dettaglio le funzionalità di due piattaforme, implementando due ambienti di Business Intelligence seguendo le esigenze reali di una PMI italiana. Un'esame completo richiederebbe risorse che vanno al di là degli scopi di questa tesi, ma il lavoro svolto è già sufficiente a delinearne le caratteristiche generali.

I due software, pur condividendo componenti simili come motori

analitici, web server e linguaggi di programmazione hanno caratteristiche strutturali molto diverse. In Pentaho BI la via scelta è quella della fusione di software in un'unica soluzione, cosa che dovrebbe permettere un maggior controllo sui componenti e una maggior integrazione. SpagoBI, essendo una piattaforma di integrazione che non interagisce con i singoli motori analitici dovrebbe essere svantaggiata da questo punto di vista. La realtà sembra capovolgere le parti a favore di SpagoBI che ha dimostrato una buona stabilità, semplicità di utilizzo e integrazione dei componenti software.

L'implementazione di PentahoBI ha creato non poche difficoltà sia nella parte di amministrazione causa l'accesso frequente ai file di configurazione in un web server complesso come nel caso di JBoss, sia nella parte di sviluppo per la mancanza di strumenti assistiti e non sempre intuitivi. Pentaho BI a discapito del numero di versione e del messaggio di marketing ha le caratteristiche di un software ancora in fase di sviluppo. Acquisire software Open Source non completamente maturi (vedi caso JFreeReport, Kettle, eccetera) e integrarli in una piattaforma con numerose funzionalità richiede d'altra parte risorse di sviluppo notevoli, che in SpagoBI è stata ben bilanciata reimpiegando progetti già esistenti come eXo Portal e lasciando i motori analitici come componenti esterni.

PentahoBI richiede nelle versioni attuali tempi di sviluppo e competenze tecniche notevoli, tali far tenere in forte considerazione l'acquisto del supporto tecnico a pagamento. La situazione è decisamente migliore in SpagoBI, dimostratosi un prodotto più robusto, completo e facile da usare.

Entrambi i prodotti richiedono una certa competenza tecnica nella loro implementazione che non consente la completa autosufficienza alle tipiche PMI italiane e va ricercata, come anche nel caso dei

tipici software per la Business Intelligence, nelle società di consulenza informatica o nei CED (Centri Elaborazione Dati) aziendali.

Un punto di demerito per entrambe le soluzioni sta nell'inserimento di parametri di tipo data, spesso importanti nei documenti analitici ma che richiedono soluzioni discutibili per realizzarli.

Una volta implementata e funzionante la piattaforma SpagoBI ben si adatta alle tipiche esigenze delle PMI italiane, rendendo di fatto necessaria solo una formazione degli utenti finali per l'interfaccia grafica. L'applicabilità a grandi imprese andrebbe esaminata con studi specifici, anche se la scalabilità della piattaforma è buona in questo senso.

L'implementazione di PentahoBI invece richiede competenze tecniche e uno sforzo maggiore da parte degli sviluppatori. Una volta installata e funzionante la piattaforma PentahoBI può adattarsi, anche con correzioni a certe lacune come l'accesso agli utenti, alle imprese con esigenze modeste per la Business Intelligence, come le piccole imprese con potere manageriale ristretto a poche persone. La situazione può certamente migliorare acquistando la versione a pagamento, che non rientra nel campo dell'Open Source considerato in questa tesi.

8 Bibliografia

- S.Dulli, Favero V., “Modelli e Strutture per il Data Warehousing”, DIADE_CUSL, Padova, 2000

- M.Golfarelli, S.Rizzi, “Data Warehouse – teoria e pratica della progettazione”, McGraw-Hill, 2006

- C.Larman, “Applicare UML e i Pattern – Analsi e progettazione orientata agli oggetti”, Prentice Hall, 2005