UNIVERSITÀ DEGLI STUDI DI SALERNO

FACOLTÀ DI SCIENZE NATURALI FISICHE E MATEMATICHE

Corso di laurea in Informatica

Tesina di Basi di Dati 2

GESTIRE ED INTERROGARE UN DATAWAREHOUSE CON MYSQL + MONDRIAN, INCLUSE LE OPERAZIONI DI OLAP E GLI STRUMENTI ETL

Professori

Genoveffa Tortora

Michele Risi

Studente

Costante Luca

Matr. 0521000838

INDICE

INTRODUZIONE

La Business Intelligence è la capacità di utilizzare l'informazione (prodotta da Sistemi Informativi eterogenei) in maniera costruttiva nei processi decisionali.

Per raggiungere questo obiettivo dobbiamo tenere a mente quattro punti chiave:

- ✓ Comprendere il processo decisionale all'interno dell'azienda e, in generale, tutti i processi che producono informazione
- ✓ Organizzare l'informazione aziendale utile alla Business Intelligence in un DataWarehouse
- ✓ Definire i flussi di dati dai sistemi informativi verso il DataWarehouse
- ✓ Utilizzare strumenti idonei ad esplorare l'informazione nel Data Warehouse.

Il primo punto è quello fondamentale e ricade nella fase di analisi. Disegnare schemi dei processi, delineare gli attori coinvolti nei processi, scoprire dove risiede l'informazione e come questa è strutturata. Kimball riassume il successo di questo punto in alcune affermazioni di cui riporto le principali:

- Comprendi chi sono gli utenti finali e le loro intolleranze all'informatica.
- Scopri quali sono le decisioni in cui l'utente vuole essere assistito.
- Cerca nuovi utenti potenziali che possono essere interessati dall'applicazione.
- Scegli un sottoinsieme ragionevole dei dati presenti nell'azienda.
- Crea un interfaccia più semplice possibile per accedere ai dati.
- Assicurati dell'accuratezza dei dati.
- Continua a mantenere monitorati l'accuratezza dei dati e dei report.
- Mantieni la fiducia dell'utente.
- Cerca sempre nuove fonti di informazioni che possano migliorare la qualità del tuo dato.

Il punto due, viene solitamente visto come il più costoso, in termini di risorse umane ed economiche.

Il terzo punto dipende da dove l'informazione si trova; in molti casi è distribuita a livello nazionale quindi si avrà un accesso ai dati tramite web-services con un'architettura SOA oppure in altri casi può essere locale.

Quarto punto: esistono sistemi molto costosi per analizzare i dati e inserirli nei processi decisionali, altri open-source.

1.1. DataWarehouse

Il DataWarehousing (DW) può essere definito come un ambiente con strutture dati finalizzate al supporto delle decisioni, fisicamente separato dai sistemi operazionali. Il DW, quindi, descrive il processo di acquisizione, trasformazione e distribuzione di informazioni presenti all'interno o all'esterno delle aziende come supporto ai "decision maker".

Il DW consente dunque agli analisti e ai manager d'azienda di accedere ai dati in modo veloce, consistente e interattivo avendo a disposizione un'ampia varietà di visioni delle informazioni. In questo modo si riflette la reale dimensionalità dell'impresa a seconda della richiesta dell'utente.

Il Data Warehouse viene strutturato su quattro livelli architetturali:

- trasformazione dei dati: è il livello che presiede all'acquisizione e alla validazione dei dati nel data warehouse
- preparazione e "stoccaggio" dati: è il livello che presiede alla delivery dei dati agli utenti e alle applicazioni analitiche
- interpretazione e analisi dei dati: è il livello, ad elevato valore aggiunto, che presiede alla trasformazione dei dati in informazioni ad elevato valore strategico
- presentazione dei dati: è il livello, a basso valore aggiunto, che presiede alla presentazione finale agli utenti delle informazioni e quindi delle risposte cercate.

1.2. OLAP

OLAP è l'analisi di grandi quantità di dati organizzati in modo denormalizzato in tabelle "dimensionali" (dimension tables) e "fatti" (fact tables). Una struttura OLAP creata per questo scopo è chiamata "cubo" multidimensionale. La tabella dei "fatti" elenca i principali elementi su cui sarà costruita l'interrogazione, collegate a questa tabella vi sono le tabelle delle "dimensioni" che specificano come i dati saranno aggregati. La tabella dei fatti racchiude dati generalmente in forma numerica (quantità, vendite, guadagni, perdite…) che vengono aggregati tramite funzioni (somma, media…) in base alle dimensioni specificate.

1.3. Mondrian

Mondrian è la parte di Pentaho utilizzata per il DataWarehousing. È un server OLAP (Online Analytical Processing) Open Source interamente scritto in Java che si interfaccia a database relazionali.

Implementa le funzionalità indispensabili all'analisi dati (aggregazione, drilldown/through, slicing, dicing) ed è in grado di eseguire query espresse in MDX (Multidimensional Expressions) leggendo i dati da un RDBMS e presentando i risultati in forma multidimensionale per mezzo di una API Java.

La connessione alla base di dati di Data Warehouse avviene via JDBC, il che rende indipendente Mondrian dal particolare RDBMS utilizzato. Lo schema multidimensionale della base dati può essere sia star che snowflake, e la sua descrizione viene fornita al motore sotto forma di file XML.

Mondrian è progettato per delegare all'RDBMS tutte le funzionalità che questo è in grado di eseguire al meglio, in particolare l'aggregazione e l'utilizzo, ove consentito, di materialized views per ottimizzare la velocità di risposta. Le raffinate strategie di caching consentono buone prestazioni in termini di velocità di esecuzione.

Mondrian viene utilizzato per:

- ✓ analizzare il rendimento aziendale attraverso l'analisi di grandi o piccoli volumi di informazioni
- ✓ l'esplorazione "dimensionale" dei dati, ad esempio analizzando i volumi di vendita per linee di prodotto, regioni geografiche o periodi di tempo
- ✓ trasformare il linguaggio MDX in SQL (Structured Query Language) per rintracciare le risposte alle domande "dimensionali"
- ✓ query ad alta velocità con l'uso delle tabelle aggregate nel RDBMS (Relational Data Base Management System)
- ✓ effettuare calcoli avanzati usando le espressioni di calcolo del linguaggio MDX.

Le sue principali caratteristiche sono:

- ✓ facilità di navigazione
- ✓ visualizzazione dei grafici sincronizzata con il percorso di navigazione
- ✓ estensibilità dell'interfaccia utente
- ✓ semplicità di integrazione
- ✓ indipendenza dalla piattaforma.

Scheda tecnica

- ✓ linguaggio di programmazione: Java
- ✓ database utilizzati: JDBC
- ✓ licenza: Common Public License
- ✓ lingue disponibili: Inglese, Tedesco
- ✓ sistema operativo: indipendente dal SO (Linux, Windows, MacOS, Solaris).

1.4. JPIVOT

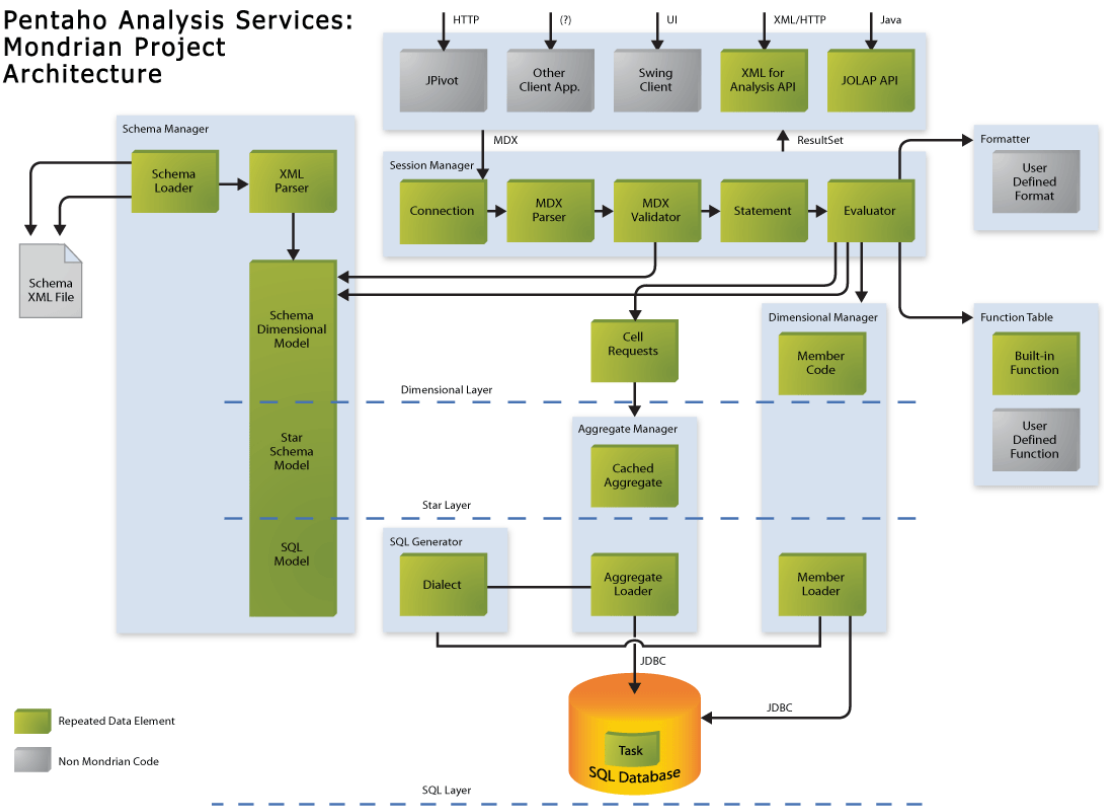
Jpivot è una libreria di tag JSP custom che consente di visualizzare una tabella OLAP ed eseguire su di essa le operazioni tipiche di navigazione, quali slice/dice, drill e roll-up.

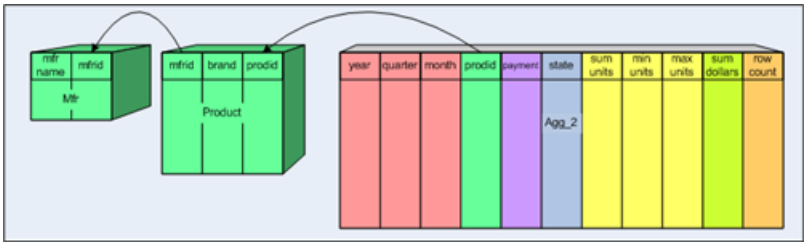
Utilizza Mondrian come motore OLAP, ma può interagire anche con sorgenti dati XMLA (XML for Analysis). Jpivot si basa sulla libreria WCF (Web Component Framework) per il rendering degli oggetti grafici dell'interfaccia utente e sul noto pacchetto JfreeChart per il tracciamento di grafici.

1.5. Architettura Mondrian-Jpivot

Mondrian-Jpivot sono implementati tramite un pool di servlet Java e tag libraries JSP. La comunicazione con l'RDBMS (Relational Data Base Managment System) di Data Warehouse avviene via JDBC, il che li rende indipendenti dal database utilizzato e dalla sua collocazione fisica. È possibile autenticare gli utenti e fornire loro viste parziali e funzionalità di analisi limitate in base ai loro privilegi.

Il deployment avviene in un Application Server, ed è semplice l'integrazione in una web-application preesistente o in un portale.





Pregi dell'architettura:

- ✓ elevata scalabilità: le prestazioni dipendono dall'RDBMS e dall'Application Server adottati
- ✓ flessibilità di installazione: RDBMS, motore OLAP, strato di presentazione possono essere installati su macchine diverse
- ✓ indipendenza dal sistema operativo.

Nell'esempio studiato, il database che sta alla base del DataWarehouse è stato appena realizzato in MySql. Ipotizziamo la situazione in cui 2 aziende vengano fuse. Si ha la necessità che entrambi i database aziendali vengano condivisi per progettare un DataWarehouse che possa essere di supporto alle politiche di marketing da adottare. In questo caso bisognerebbe provvedere alla operazione di riconciliazione ecc.

1.6. ETL

ETL è l'acronimo, in inglese, di Extract Transform e Load. Indica quella tipologia di strumenti atti ad estrarre informazione da diverse fonti, elaborarla e solitamente memorizzarla in un Data Warehouse. I dati possono essere estratti da sistemi sorgenti quali database transazionali, comuni files di testo o da altri sistemi informatici.

Fra gli strumenti Open Source presenti sul mercato ho provato Pentaho Data Integration. Il tool è scaricabile dal sito di Pentaho e si tratta di un'applicazione scritta in Java avviabile appena unzippata.

CASO DI STUDIO: CATENA DI NEGOZI PER LA VENDITA DI COMPUTER

2.1. Requisiti di sistema

Per realizzare questo lavoro, sono stati utilizzati i seguenti strumenti:

- mondrian-3.1.1.12687.zip
- mondrian-3.1.1.12687-derby.zip
- MySQL Server
- MySQL JDBC driver
- Tomcat 5.0.28
- ETL: Pentaho DataIntegration

2.2. Installazione di Tomcat e MySql

Per prima cosa bisogna installare Java 5 o superiore, MySql ed Apache Tomcat. Assicurarsi di avere nel path TOMCAT_HOME/lib la libreria per MySQL Driver JDBC altrimenti scaricarla dal sito (http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.0.7.zip) e copiarla in TOMCAT_HOME/lib. Settare la variabile di ambiente JAVA_HOME in C:\Programmi\Java\jdk1.6.0_06 (o diversamente, dipende da dove lo avete installato). Nella fase successiva andiamo a creare ed a popolare il nostro database. Per l'esempio è stato realizzato un database basato sulla vendita di computer. Nel seguito troviamo la struttura ed il popolamento del DB.

```
-- MySQL Administrator dump 1.4
-- ------------------------------------------------------
-- Server version       5.0.27-community-nt
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

--
-- Create schema computer
--

CREATE DATABASE IF NOT EXISTS computer;
USE computer;

--
-- Definition of table `negozio_dt`
--

DROP TABLE IF EXISTS `negozio_dt`;
CREATE TABLE `negozio_dt` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `regione` varchar(255) NOT NULL,
  `codistat_regione` varchar(3) NOT NULL,
  `provincia` varchar(255) NOT NULL,
  `codistat_provincia` varchar(3) NOT NULL,
  `comune` varchar(255) default NULL,
  `codistat_comune` varchar(255) default NULL,
  `indirizzo` varchar(255) default NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `negozio_dt`
--

/*!40000 ALTER TABLE `negozio_dt` DISABLE KEYS */;
INSERT INTO `negozio_dt`
(`id`,`regione`,`codistat_regione`,`provincia`,`codistat_provincia`,`comune`,`codistat_co
```

```
mune`,`indirizzo`) VALUES
 (1,'Regione Liguria','07','Provincia di Genova','010','Comune di Genova','025','Via Mansueto 12'),
 (2,'Regione Liguria','07','Provincia di Genova','010','Comune di Genova','025','Via Jori 2'),
 (3,'Regione Liguria','07','Provincia di Genova','010','Comune di Genova','025','Via Zella 4'),
 (4,'Regione Liguria','07','Provincia di Genova','010','Comune di Busalla','003','Piazza Rossi 4'),
 (5,'Regione Liguria','07','Provincia di Savona','009','Comune di Savona','021','Piazza Verdi 1'),
 (6,'Regione Piemonte','06','Provincia di Asti','014','Comune di Asti','001','XX Settembre 3'),
 (7,'Regione Piemonte','06','Provincia di Alessandria','015','Comune di Alessandria','001','Via Torino 4'),
 (8,'Regione Piemonte','06','Provincia di Alessandria','015','Comune di Alessandria','001','Via Martiri della Liberta 3');
/*!40000 ALTER TABLE `negozio_dt` ENABLE KEYS */;


--
-- Definition of table `periodo_dt`
--

DROP TABLE IF EXISTS `periodo_dt`;
CREATE TABLE `periodo_dt` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `anno` int(11) NOT NULL,
  `mese` int(11) NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `periodo_dt`
--
/*!40000 ALTER TABLE `periodo_dt` DISABLE KEYS */;
INSERT INTO `periodo_dt` (`id`,`anno`,`mese`) VALUES
 (1,2009,1),
 (2,2009,2),
 (3,2009,3),
 (4,2009,4),
 (5,2009,5),
 (6,2009,6),
```

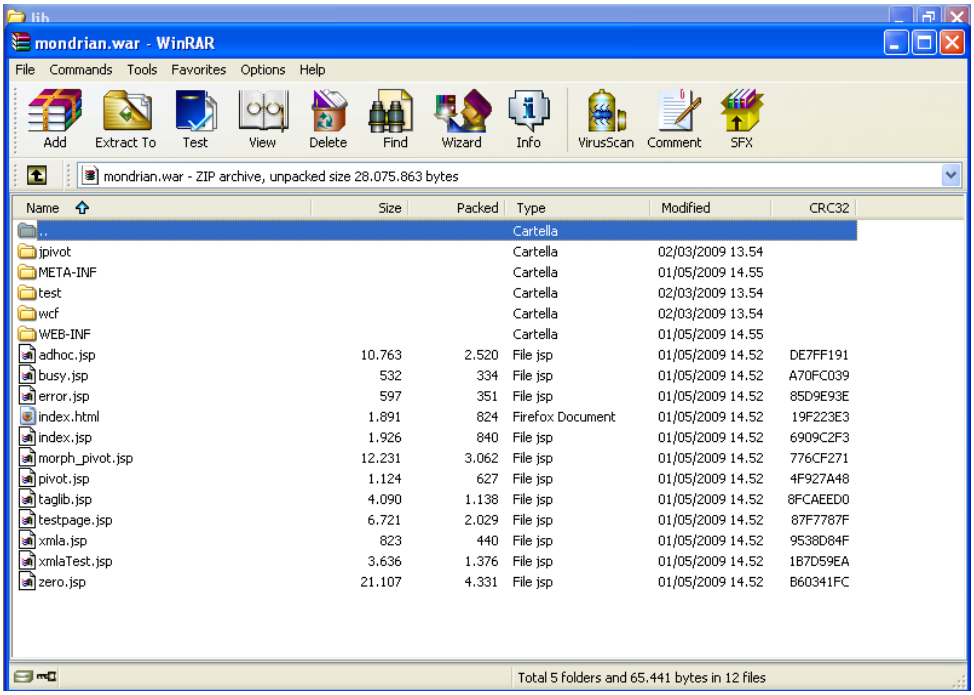
```
 (7,2009,7),
 (8,2009,8),
 (9,2009,9),
 (10,2009,10),
 (11,2009,11),
 (12,2009,12);
/*!40000 ALTER TABLE `periodo_dt` ENABLE KEYS */;

--
-- Definition of table `vendite_ft`
--

DROP TABLE IF EXISTS `vendite_ft`;
CREATE TABLE `vendite_ft` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `negozio_id` bigint(20) NOT NULL,
  `periodo_id` bigint(20) default NULL,
  `n_normali` bigint(20) default NULL,
  `n_notebook` bigint(20) default NULL,
  `n_netbook` bigint(20) default NULL,
  `n_mac` bigint(20) default NULL,
  `profitto_normali` bigint(20) default NULL,
  `profitto_notebook` bigint(20) default NULL,
  `profitto_netbook` bigint(20) default NULL,
  `profitto_mac` bigint(20) default NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `vendite_ft`
--
/*!40000 ALTER TABLE `vendite_ft` DISABLE KEYS */;
INSERT INTO `vendite_ft`
(`id`,`negozio_id`,`periodo_id`,`n_normali`,`n_notebook`,`n_netbook`,`n_mac`,`profitto
_normali`,`profitto_notebook`,`profitto_netbook`,`profitto_mac`) VALUES
 (1,1,1,20,4,10,0,2000,400,1000,0), (2,1,2,20,4,20,0,2000,400,2000,0),
 (3,1,3,20,10,2,4,2000,1000,200,400), (4,1,4,20,8,0,4,2000,800,0,400),
 (5,1,5,30,8,0,8,3000,0,0,800), (6,1,6,30,0,0,6,3000,0,0,600),
 (7,1,7,30,0,0,8,3000,0,0,800), (8,1,8,20,0,0,6,2000,0,0,600),
 (9,1,9,20,0,0,8,2000,0,0,800), (10,1,10,20,9,2,0,2000,200,1000,0),
 (11,1,11,10,8,4,0,1000,400,1000,0), (12,1,12,10,9,4,0,1000,400,1000,0),
 (13,2,1,20,4,10,0,2000,400,1000,0), (14,2,3,20,10,2,4,2000,1000,200,400),
```

```
 (15,2,4,20,8,0,4,2000,800,0,400), (16,2,5,30,8,0,8,3000,0,0,800),
 (17,2,6,30,0,0,6,3000,0,0,600), (18,2,7,30,0,0,8,3000,0,0,800),
 (19,2,8,20,0,0,6,2000,0,0,600), (20,2,9,20,0,0,8,2000,0,0,800),
 (21,2,10,20,9,2,0,2000,200,1000,0),
(22,2,11,10,8,4,0,1000,400,1000,0),
 (23,2,12,10,9,4,0,1000,400,1000,0),
(24,3,4,20,8,0,4,2000,800,0,400),
 (25,3,5,30,8,0,8,3000,0,0,800);
INSERT INTO `vendite_ft`
(`id`,`negozio_id`,`periodo_id`,`n_normali`,`n_notebook`,`n_netbook`,`n_mac`,`profitto
_normali`,`profitto_notebook`,`profitto_netbook`,`profitto_mac`) VALUES
 (26,3,6,30,0,0,6,3000,0,0,600), (27,3,7,30,0,0,8,3000,0,0,800),
 (28,3,8,20,0,0,6,2000,0,0,600), (29,3,9,20,0,0,8,2000,0,0,800),
 (30,3,10,20,9,2,0,2000,200,1000,0), (31,3,11,10,8,4,0,1000,400,1000,0),
 (32,3,12,10,9,4,0,1000,400,1000,0), (33,4,1,12,4,10,0,1200,400,1000,0),
 (34,4,2,10,4,11,0,1000,400,1100,0), (35,4,3,21,12,2,6,2100,1200,200,600),
 (36,4,4,15,7,0,3,1500,700,0,300), (37,4,5,23,4,0,8,2300,400,0,800),
 (38,4,6,12,0,0,9,1200,0,0,900),
(39,4,7,30,0,0,8,3000,0,0,800),
 (40,4,8,20,0,0,6,2000,0,0,600),
(41,4,9,29,0,0,9,2900,0,0,900),
 (42,4,10,24,6,1,0,2400,600,100,0),
(43,4,11,10,8,4,0,1000,400,1000,0),
 (44,4,12,10,9,4,0,1000,400,1000,0),
(45,5,1,40,8,20,0,4000,800,2000,0),
 (46,5,2,43,4,30,0,4300,400,3000,0),
 (47,5,3,20,60,2,8,2000,6000,200,800),
 (48,5,4,20,8,0,4,2000,800,0,400),
(49,5,5,30,8,0,8,3000,0,0,800);
INSERT INTO `vendite_ft`
(`id`,`negozio_id`,`periodo_id`,`n_normali`,`n_notebook`,`n_netbook`,`n_mac`,`profitto
_normali`,`profitto_notebook`,`profitto_netbook`,`profitto_mac`) VALUES
 (50,5,6,23,0,0,9,2300,0,0,900),
 (51,5,7,30,0,0,8,3000,0,0,800),
 (52,5,8,20,0,0,6,2000,0,0,600),
 (53,5,9,20,0,0,8,2000,0,0,800),
 (54,5,10,12,9,9,0,1200,200,900,0),
 (55,5,11,10,8,4,0,1000,400,1000,0),
 (56,5,12,10,9,4,0,1000,400,1000,0),
 (57,6,1,20,4,10,0,2000,400,1000,0),
 (58,6,2,20,4,20,0,2000,400,2000,0),
 (59,6,3,20,10,2,4,2000,1000,200,400),
 (60,6,4,20,8,0,4,2000,800,0,400),
```

```
 (61,6,5,30,8,0,8,3000,0,0,800),
 (62,6,6,30,0,0,6,3000,0,0,600),
 (63,6,7,30,0,0,8,3000,0,0,800),
 (64,6,8,20,0,0,6,2000,0,0,600),
 (65,6,9,20,0,0,8,2000,0,0,800),
 (66,6,10,20,9,2,0,2000,200,1000,0),
 (67,6,11,10,8,4,0,1000,400,1000,0),
 (68,6,12,10,9,4,0,1000,400,1000,0),
 (69,7,1,12,4,10,10,1200,400,1000,1000),
 (70,7,2,20,4,20,4,2000,400,2000,400),
 (71,7,3,10,10,2,4,1000,1000,200,400),
 (72,7,4,20,18,10,4,2000,1800,1000,400),
 (73,7,5,30,8,0,8,3000,0,0,800);
INSERT INTO `vendite_ft`
(`id`,`negozio_id`,`periodo_id`,`n_normali`,`n_notebook`,`n_netbook`,`n_mac`,`profitto
_normali`,`profitto_notebook`,`profitto_netbook`,`profitto_mac`) VALUES
 (74,7,6,30,0,8,6,3000,0,800,600),
 (75,7,7,30,0,5,8,3000,0,500,800),
 (76,7,8,20,10,10,6,2000,1000,1000,600),
 (77,7,9,20,0,0,8,2000,0,0,800),
 (78,7,10,20,9,2,0,2000,200,1000,0),
 (79,7,11,10,8,4,0,1000,400,1000,0),
 (80,7,12,10,9,4,0,1000,400,1000,0),
 (81,8,1,4,4,10,0,400,400,1000,0),
 (82,8,2,0,4,2,0,0,400,200,0),
 (83,8,3,3,10,2,4,300,1000,200,400),
 (84,8,4,10,0,0,4,1000,0,0,400),
 (85,8,5,3,8,0,8,300,0,0,800),
 (86,8,6,4,0,0,6,400,0,0,600),
 (87,8,7,0,0,0,8,0,0,0,800),
 (88,8,8,10,0,0,0,1000,0,0,0),
 (89,8,9,2,0,0,1,200,0,0,100);
/*!40000 ALTER TABLE `vendite_ft` ENABLE KEYS */;


/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

2.3. Configurazione di Mondrian

Una volta scaricato l'archivio di Mondrian ed averlo scompattato, all'interno troveremo una cartella di nome lib contenente il package mondrian.war.



Scompattiamolo e andiamo a copiarlo all'interno di TOMCAT_HOME/webapps/mondrian. Ora bisogna configurare Mondrian per poterlo utilizzare. All'interno della cartella TOMCAT_HOME/webapps/mondrian/WEB-INF/queries, vi è un file FoodMart.xml che viene utilizzato per la definizione del cubo di analisi. Creiamo un nuovo file computer.xml simile a FoodMart.xml.

```
<?xml version="1.0"?>
<Schema name="computer">
<!-- Shared dimensions -->

 <Dimension name="Negozio">
  <Hierarchy hasAll="true" allMemberName="Tutti i negozi"  primaryKey="id">
    <Table name="negozio_dt"/>
   <Level name="Regione" column="regione" uniqueMembers="true"/>
   <Level name="Provincia" column="provincia" uniqueMembers="false"/>
   <Level name="Comune" column="comune" uniqueMembers="false"/>
```

```
    <Level name="Indirizzo" column="indirizzo" uniqueMembers="true"/>
   </Hierarchy>
  </Dimension>

  <Dimension name="Periodo">
   <Hierarchy hasAll="true" allMemberName="Tutti i periodi" primaryKey="id">
    <Table name="periodo_dt" />
    <Level name="Anno" column="anno" type="Numeric" uniqueMembers="true"/>
    <Level name="Mese" column="mese" type="Numeric" uniqueMembers="false"/>
   </Hierarchy>
  </Dimension>

  <Cube name="Vendite">
   <Table name="vendite_ft"/>
   <DimensionUsage name="Negozio" source="Negozio" foreignKey="negozio_id"/>
   <DimensionUsage name="Periodo" source="Periodo" foreignKey="periodo_id"/>
   <Measure name="Qta Normali" column="n_normali" aggregator="sum"
formatString="#,###"/>
   <Measure name="Qta Notebook" column="n_notebook" aggregator="sum"
formatString="#,###"/>
   <Measure name="Qta NetBook" column="n_netbook" aggregator="sum"
formatString="#,###"/>
   <Measure name="Qta Apple MAC" column="n_mac" aggregator="sum"
formatString="#,###"/>
   <Measure name="Profitto Normali" column="profitto_normali" aggregator="sum"
formatString="#,###.##"/>
   <Measure name="Profitto Notebook" column="profitto_notebook"
aggregator="sum" formatString="#,###.##"/>
   <Measure name="Profitto Netbook" column="profitto_netbook" aggregator="sum"
formatString="#,###.##"/>
   <Measure name="Profitto Apple MAC" column="profitto_mac" aggregator="sum"
formatString="#,###.##"/>
  </Cube>
</Schema>
```

La definizione di uno Schema inizia con la definizione del suo nome, in questo caso computer. Come possiamo vedere dalla descrizione del file xml, le dimensioni vengono dichiarate al di fuori del Cubo e poi sono referenziate in esso. La nostra dimensione Negozio è associata alla tabella negozio_dt.

Successivamente questa dimensione è strutturata in maniera gerarchica in livelli. Si inizia dal concetto di Regione suddivisa in Province, suddivise a loro volta in Comuni e all'ultimo livello i negozi ubicati nei comuni.

La dimensione del tempo chiamata Periodo è associata alla tabella periodo_dt e suddivisa nei livelli Anno e Mese.

La tabella dei fatti è invece associata alla tabella vendite_ft. Al suo interno richiama le due dimensioni Negozio e Periodo precedentemente definite.

Dopo l'utilizzo delle dimensioni si definiscono le misure. Anche le misure sono associate ad una colonna e viene definita la funzione di aggregazione. Nel nostro caso è la somma (sum), ma possono essere usate la media, il count oppure possono essere definite, tramite un'opportuna interfaccia, le proprie funzioni di aggregazione.

La definizione del Cubo è la parte più importante di Mondrian. Questa può essere fatta anche attraverso un wizard grafico fornito da Pentaho (CubeDesigner), ma vista la facilità di lettura di questo XML si è preferito scriverla.

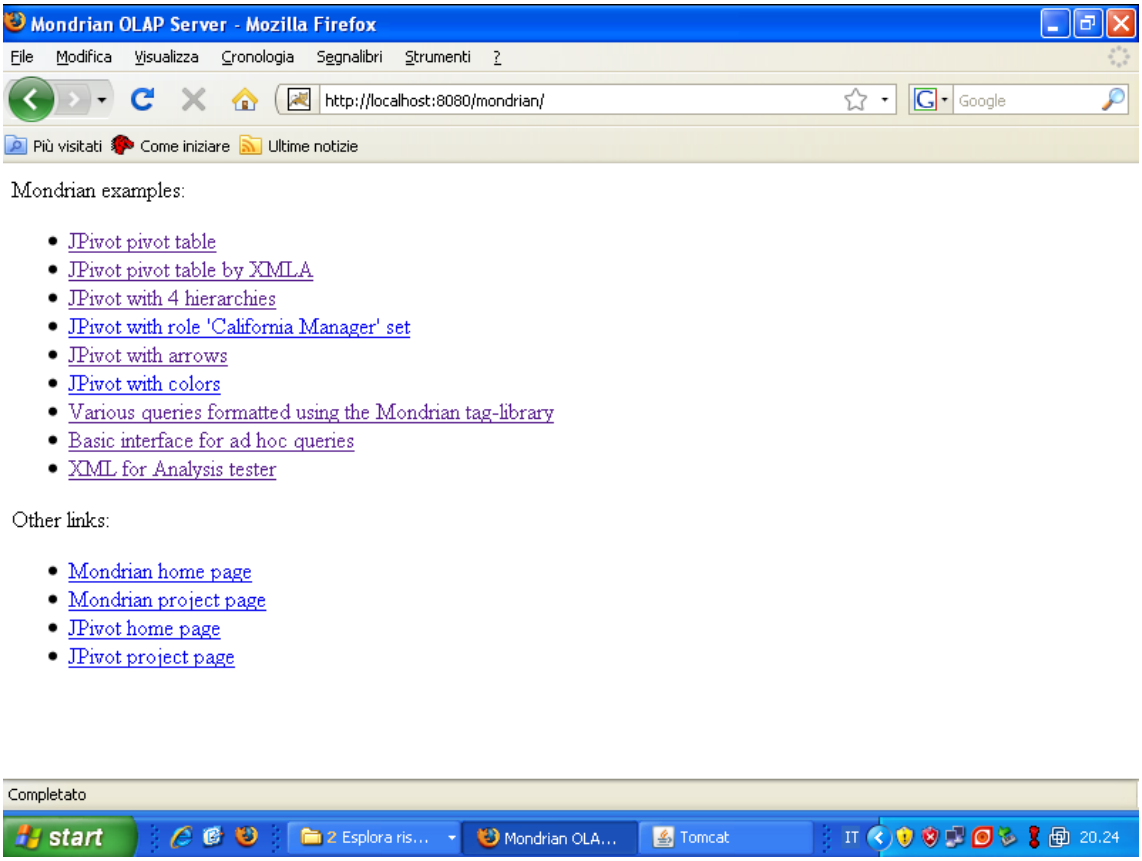
Definita la parte in Mondrian, passiamo ad utilizzare il cubo con JPivot. L'esempio fornito con Mondrian utilizza un file index.jsp che richiama testpage.jsp passandogli la query da utilizzare. Queste query sono nella directory TOMCAT_HOME/Mondrian/WEB-INF/queries. Modifichiamo WEB-INF/queries/mondrian.jsp per farla funzionare con il nostro esempio.

```
<%@ page session="true" contentType="text/html; charset=ISO-8859-1" %>
<%@ taglib uri="http://www.tonbeller.com/jpivot" prefix="jp" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<%-- Author: COSTANTE LUCA --%>

<jp:mondrianQuery id="query01"
jdbcDriver="com.mysql.jdbc.Driver"
jdbcUrl="jdbc:mysql://127.0.0.1:3306/computer"
jdbcUser="root" jdbcPassword="root"
catalogUri="/WEB-INF/queries/computer.xml">
```

```
select
  {[Measures].MEMBERS} on columns,
  {([Negozio].[Tutti i negozi],[Periodo].[Tutti i periodi])} ON rows
from vendite
</jp:mondrianQuery>
<c:set var="title01" scope="session">Vendita Computer</c:set>
```

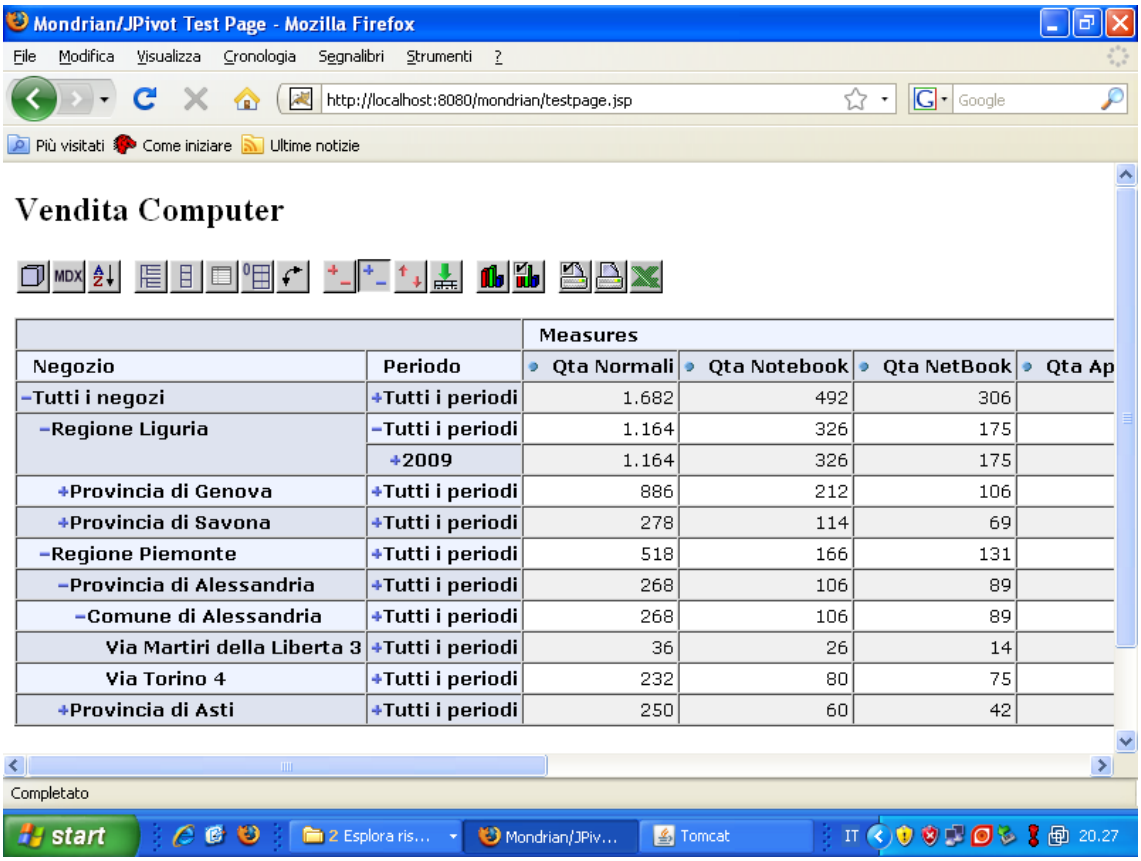
In questo esempio, poiché è stato utilizzato il database MySql, ho inserito i parametri relativi al tipo di driver JDBC che bisogna caricare, il nome dell'utente, la relativa password, la locazione del database (l'ho provato il locale quindi l'indirizzo è 127.0.0.1 sulla porta 3306 ma benissimamente può trovarsi in qualsiasi parte della terra, basta cambiare l'IP, il numero di porta, i relativi nome utente e password).

Per prima cosa andiamo a testare se mondrian è raggiungibile. Per farlo apriamo il nostro browser e digitiamo http://localhost:8080/mondrian.



A questo punto il gioco è fatto. Lanciare l'applicazione e cliccare sul primo link JPivot pivot table (quello che abbiamo modificato) della home page. Il risultato è una tabella pivot esplorabile sino alla grana più fine definita.

Notate anche la possibilità di fare grafici, di esportare in pdf e in excel. Tutto questo rende lo strumento di un'utilità impressionante.



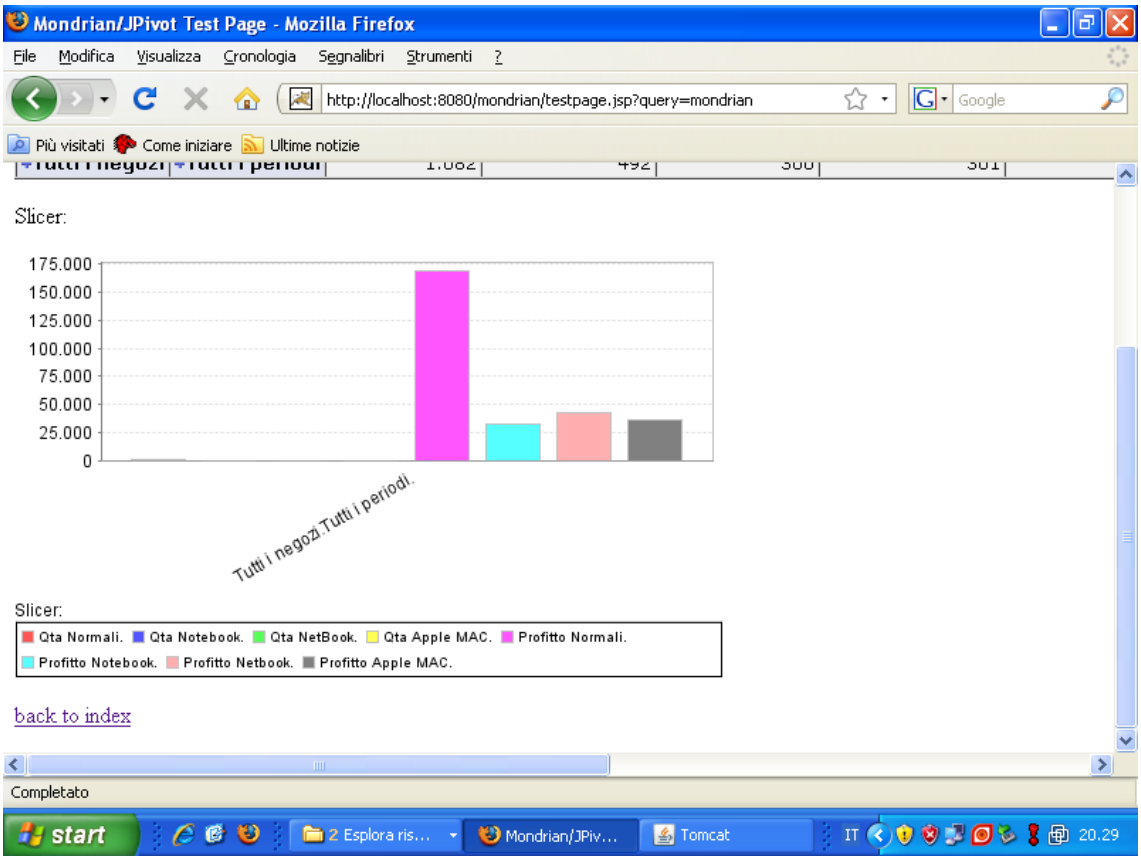
		Measures			
Negozio	Periodo	Qta Normali	Qta Notebook	Qta NetBook	Qta Ap
-Tutti i negozi	+Tutti i periodi	1.682	492	306	
-Regione Liguria	-Tutti i periodi	1.164	326	175	
	+2009	1.164	326	175	
+Provincia di Genova	+Tutti i periodi	886	212	106	
+Provincia di Savona	+Tutti i periodi	278	114	69	
-Regione Piemonte	+Tutti i periodi	518	166	131	
-Provincia di Alessandria	+Tutti i periodi	268	106	89	
-Comune di Alessandria	+Tutti i periodi	268	106	89	
Via Martiri della Liberta 3	+Tutti i periodi	36	26	14	
Via Torino 4	+Tutti i periodi	232	80	75	
+Provincia di Asti	+Tutti i periodi	250	60	42	

L'intelligenza sta nell'organizzare una grande quantità di dati (l'esempio è solo dimostrativo) e presentarli in maniera razionale ad un decisore (Decision Maker) per poter essere analizzati.

Tramite lo strumento offerto possiamo rispondere a domande quali:

- Quali sono i periodi dell'anno in cui vendiamo più notebook?
- Quali territori sono maggiormente interessati all'acquisto di netbook?
- Quali negozi hanno i maggiori profitti?

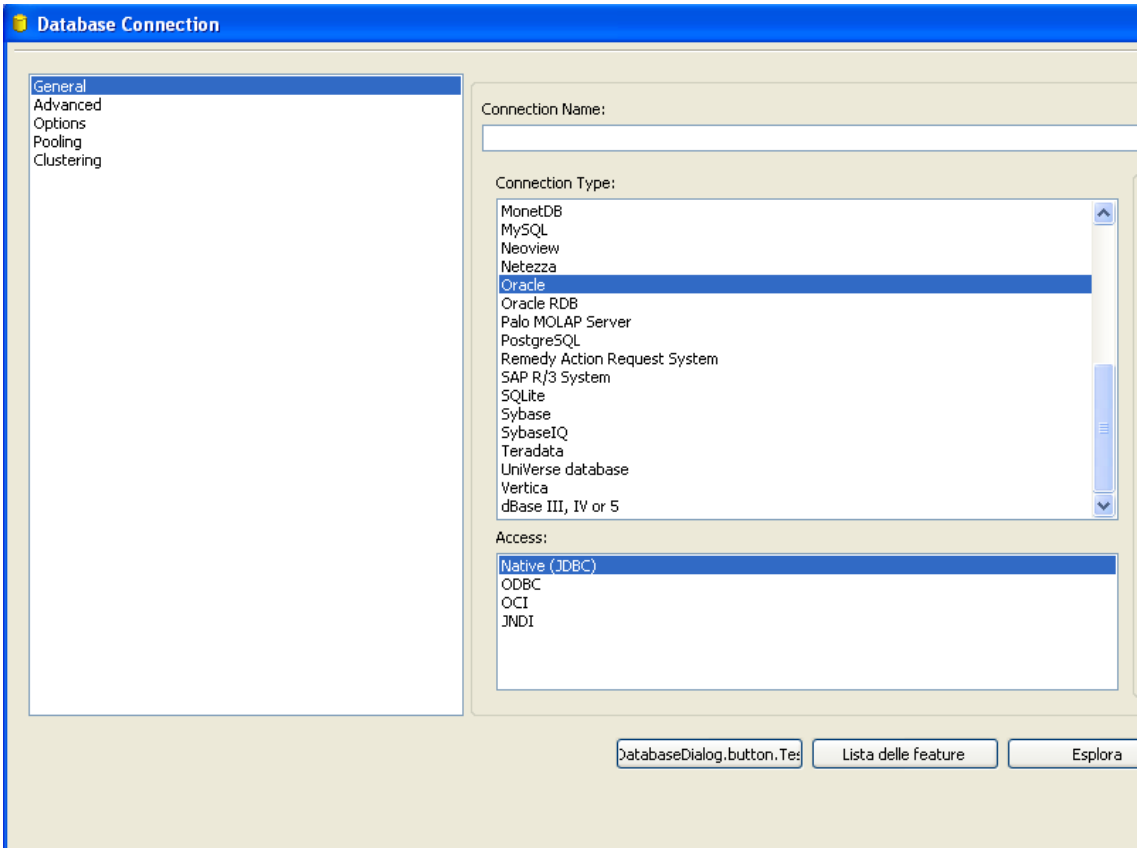
- L’apertura di un negozio a Genova ha portato ad una riduzione di vendite negli altri negozi?



2.4. Utilizzo di ETL

Effettuiamo l’installazione di Pentaho Data Integration (PDI) Enterprise Edition (trial 30 giorni).

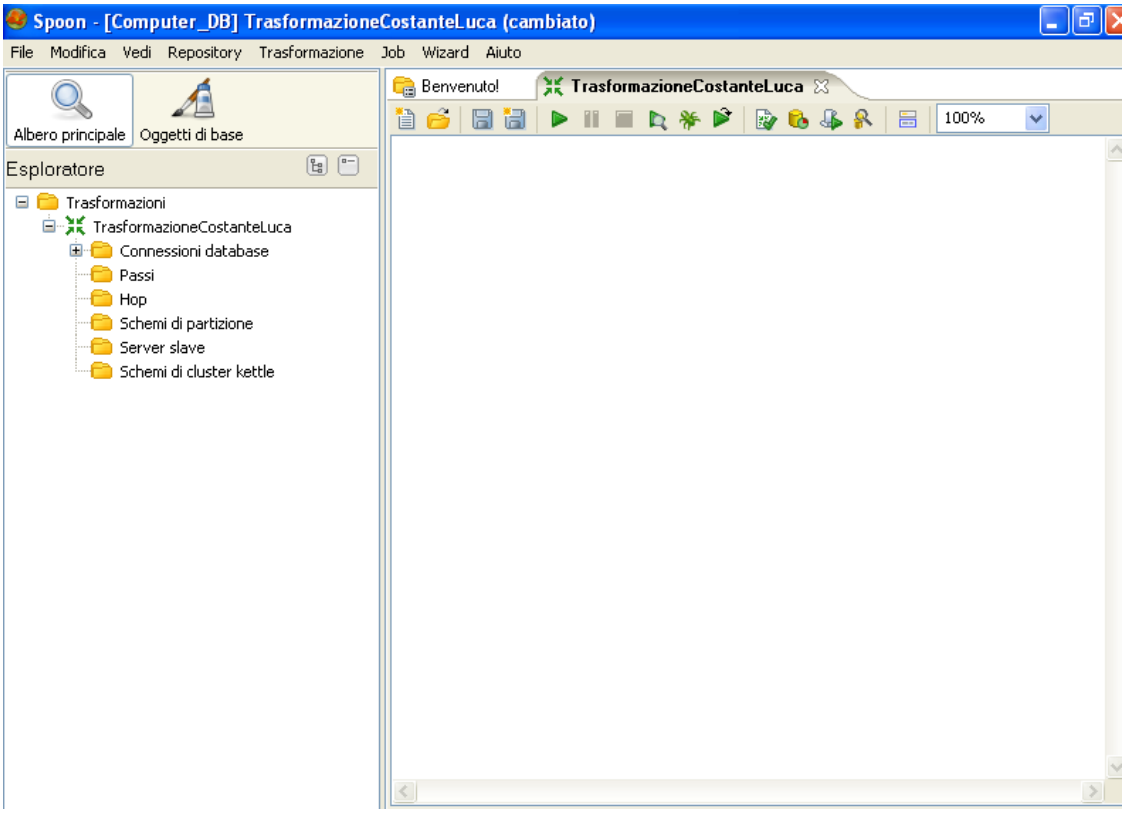
Avviando per la prima volta l’applicativo, si deve effettuare la fase di configurazione. Provvediamo quindi ad impostare la connessione al nostro database.



Per iniziare il mio primo test, ho creato una nuova Trasformazione che ho chiamato TrasformazioneCostanteLuca. All'interno di una Trasformazione possono essere inseriti diversi data source. Lo scopo del mio esempio è quello di caricare le informazioni dei dipendenti a partire da un file CSV così strutturato:

- ✓ id,cognome,nome
- ✓ 1,Costnate,Luca
- ✓ 2,Giardino,Daniele
- ✓ 3,Palo,Umberto
- ✓ 4,Schiavone,Dario
- ✓ 5,Caruso,Barbara
- ✓ 6,Silvestri,Daniele
- ✓ 7,Rossi,Vasco
- ✓ 8,De Filippi,Maria

PDI è molto semplice da usare. Ogni processo viene creato attraverso un tool grafico dove viene specificato cosa fare senza scrivere codice per indicare come farlo. PDI supporta una vasta tipologia di input ed output.



Immaginiamo di voler caricare, nel database precedentemente creato, le informazioni dei dipendenti a partire da un file dipendenti.csv.

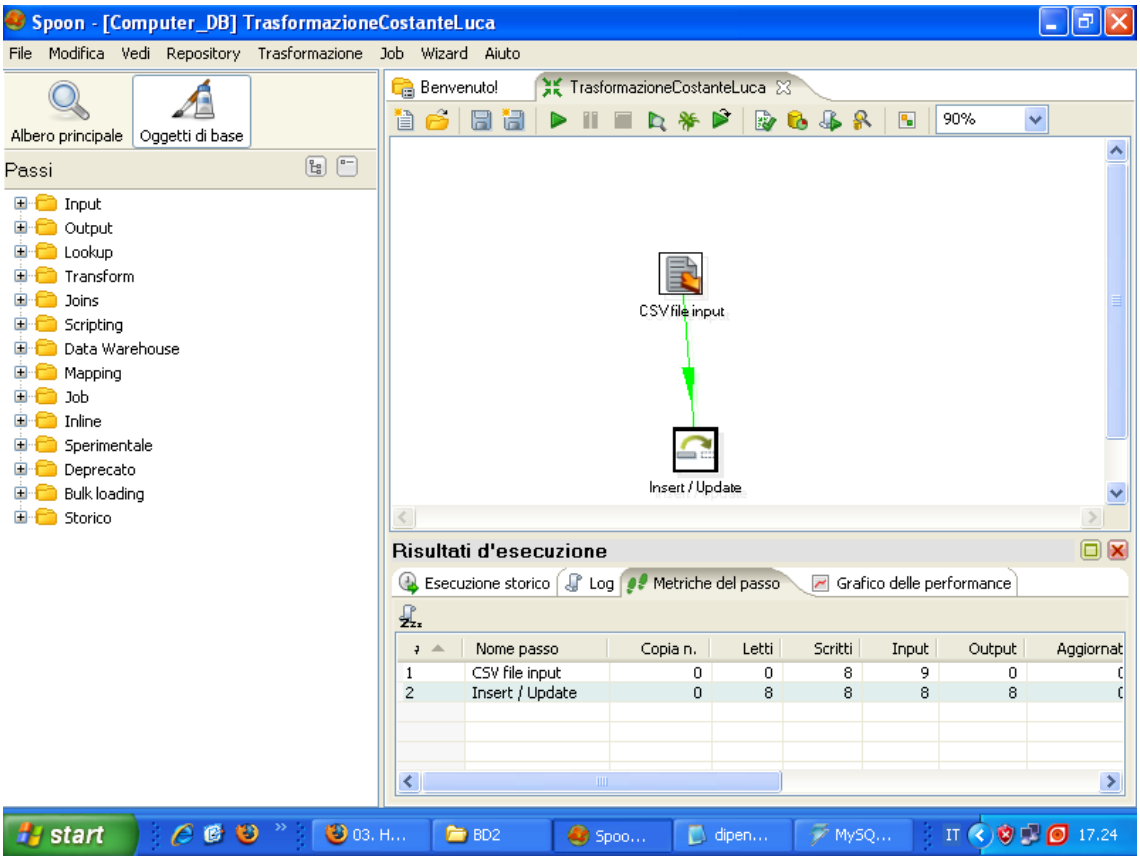
A partire dalla nostra trasformazione dobbiamo definire quelli che PDI chiama Steps, ossia l'unità più piccola all'interno delle Trasformazioni. Poiché PDI permette una vasta categoria di operazioni, queste sono state raggruppate in Input, Output, ecc. Un hop è una rappresentazione grafica per specificare l'evento tra 2 step. Un hop ha sempre una sola origine ed una sola destinazione.

La trasformazione che stiamo trattando è caratterizzata dalla seguenti operazioni:

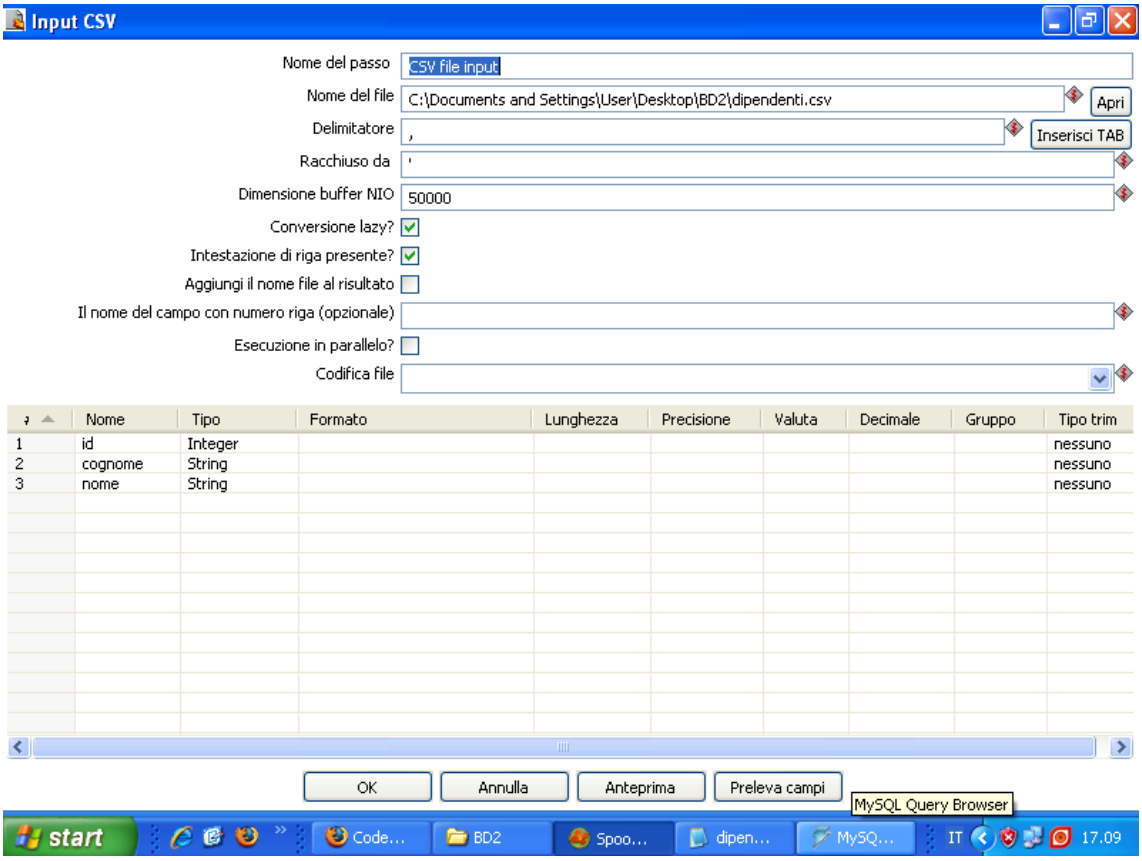
1. Lettura del file CSV
2. Salvare la trasformazione nel database

In PDI, quindi, effettuiamo le seguenti operazioni:

I. Nella parte sinistra del workspace, selezioniamo la categoria Input e spostiamo nella parte destra il file CSV.
II. Ora selezioniamo la categoria output e trasciniamo verso destra Insert/Update.
III. Per ogni elemento, ora dobbiamo creare il collegamento. Prendiamo il file CSV e tenendo premuto il tasto shift lo spostiamo verso il file di Insert/Update.



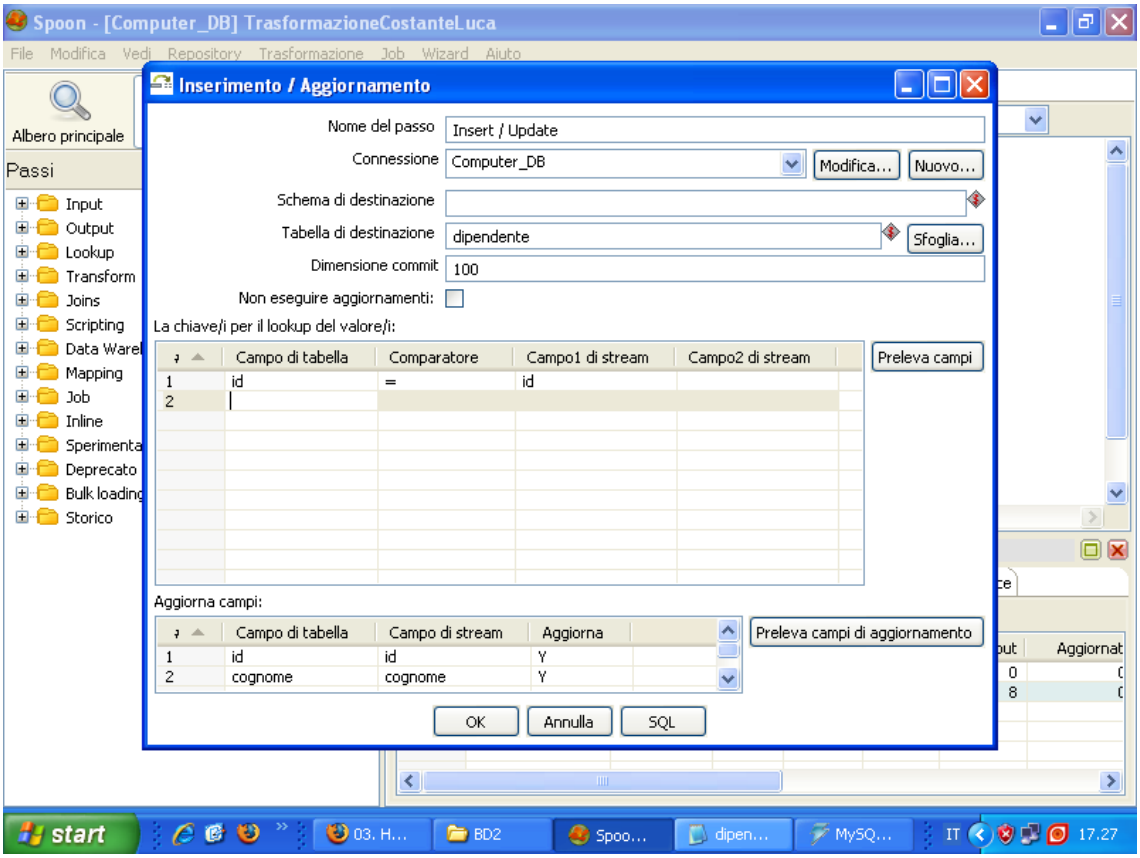
Ora bisogna configurare il file InputCSV, premendo doppio click sull'oggetto andiamo a specificare i vari parametri come da figura sotto.



Ora andiamo nel Database e dobbiamo solo creare la tabella coi rispettivi campi:

- id;
- cognome;
- nome.

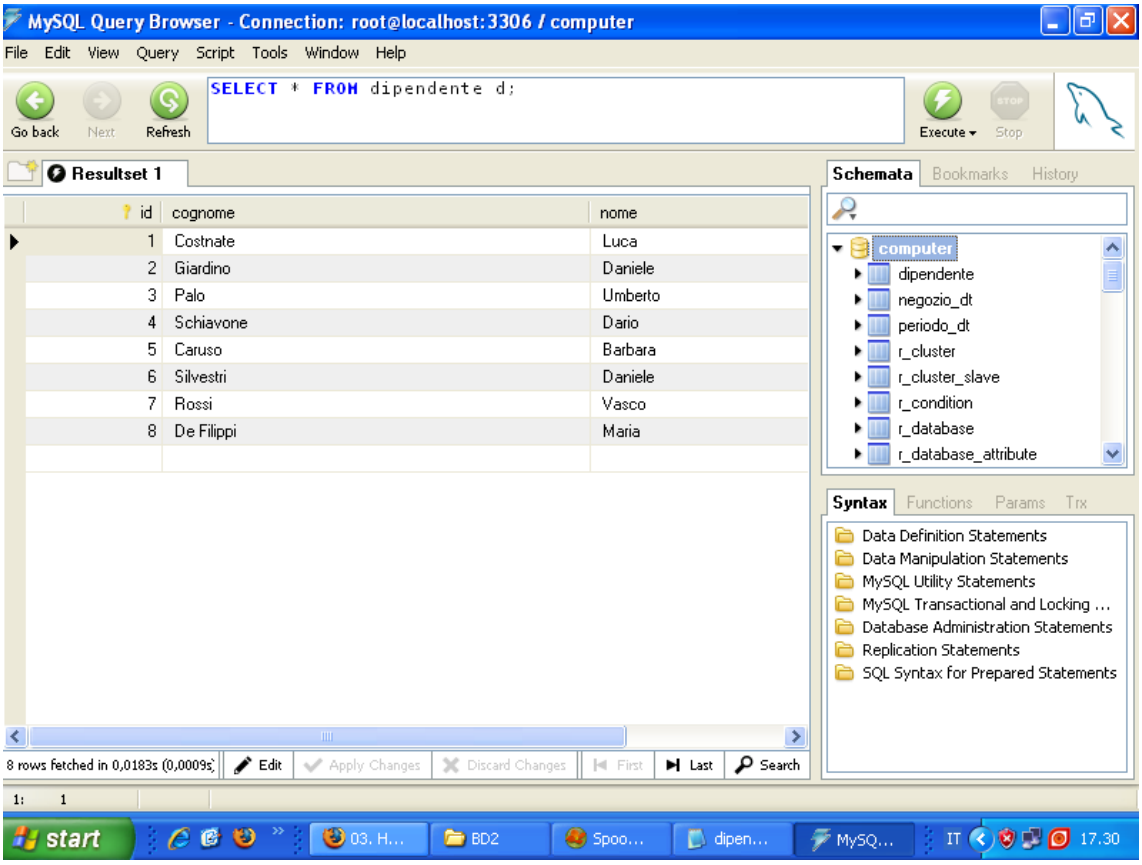
Infine dobbiamo specificare il comportamento dello step Insert/Update come da figura seguente.



Ora non ci resta che salvare la Trasformazione e premere su Avvia.

Nella parte sottostante della schermata, vediamo il log dell'operazione. Se l'hop non viene segnalato in rosso, tutto si è svolto correttamente.

Scrupolosamente, andiamo a controllare cosa è accaduto nel database. Oltre ad essere state inserite una serie di nuove tabelle da parte del tool, effettuando una query sulla tabella dipendente popolata dal tool in oggetto, vediamo che effettivamente l'operazione è andata a buon fine.



PDI si rivela quindi uno strumento molto utile, data anche la grande quantità di sorgenti (quasi tutti i DB che abbiano un driver jdbc, CSV, dati in excel, ecc.) e destinazioni (DB, fogli di testo, excel), permette una vasta gamma di operazioni per il caricamento e la conversione dei dati.