Architettura e Modello Dati

Della Piattaforma C&T di GIT

Marco Riccardini

[Architettura di sistema 1](#_Toc314064342)

[1.1. Architettura dati 1](#_Toc314064343)

[1.1.1. Architettura generale dei dati 1](#_Toc314064344)

[1.1.2. Flusso dei dati 2](#_Toc314064345)

[1.1.3. Staging Locale 4](#_Toc314064346)

[1.1.4. Repository forniture 4](#_Toc314064347)

[1.1.5. Staging Db 5](#_Toc314064348)

[1.1.6. Architettura del data warehouse di I Livello 5](#_Toc314064349)

[1.1.7. Architettura del data warehouse di II Livello 7](#_Toc314064350)

[1.2. Architettura Software 7](#_Toc314064351)

[1.2.1. Modello di architettura distribuita 7](#_Toc314064352)

[1.2.2. Scelte inerenti lo strato di accesso ai dati 8](#_Toc314064353)

[1.2.3. Sicurezza 9](#_Toc314064354)

[1.2.4. Pattern utilizzati 9](#_Toc314064355)

[1.2.5. Componenti software 9](#_Toc314064356)

[1.2.6. Schema architettura 10](#_Toc314064357)

[Modello dei Dati 12](#_Toc314064358)

[1.3. Schema Dati 14](#_Toc314064359)

[1.3.1. Schema Dati Staging DB 14](#_Toc314064360)

[1.3.2. Schema Dati DWH I livello 14](#_Toc314064361)

[1.3.3. Schema Dati DWH II livello 17](#_Toc314064362)

# Architettura di sistema

## Architettura dati

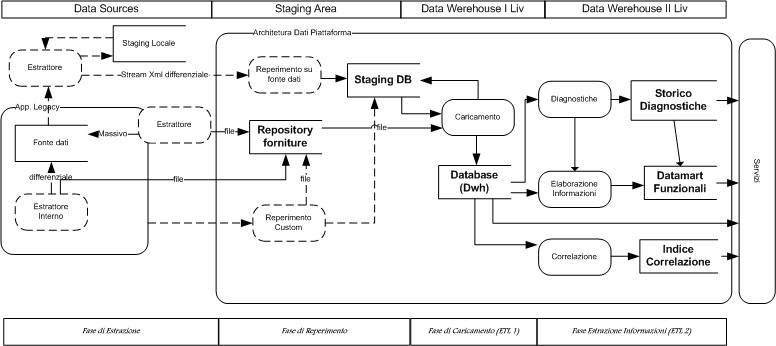
### Architettura generale dei dati

Lo schema seguente mostra i vari livelli di dati e la loro distribuzione rispetto ai livelli fisici della piattaforma.



### Flusso dei dati

La modalità standard di caricamento della piattaforma (modalità embedded) è quella che vede la fornitura estratta nel repository delle forniture in formato txt con tipi record (fatte salve le forniture istituzionalizzate) .



La piattaforma può accedere alle forniture dati in differenti modalità, configurabili o programmabili a seconda del grado di complessità della fonte dati da importare.

Particolare attenzione viene posta alle problematiche di avvio e di gestione del patrimonio dati della piattaforma.

La fase di estrazione della fornitura dati non è una funzione appartenente al dominio applicativo della piattaforma, tale funzione, eventualmente richiesta , potrà essere svolta dai moduli di reperimento e da implementazione e configurazioni custom.

I processi o le funzioni di reperimento si occupano di reperire la fornitura dati indipendentemente dal suo contenuto applicativo.

I processi di caricamento dovranno invece conoscere che tipologia di fornitura possiedono in input in base alla caratteristiche che possiamo denominare consistenza dell’aggiornamento.

La fornitura dati può essere sempre completa (contenente tutta la banca dati di interesse) oppure essere completa soltanto in fase di primo caricamento e poi essere costituita dai soli aggiornamenti occorsi.

Ne caso di fornitura completa la piattaforma è munita di meccanismi che evitano la duplicazione delle informazioni da uno scarico all’altro e che permettono una storicizzazione del dato solo per quelle informazioni che hanno subito un aggiornamento: la procedura di caricamento non risulterà efficiente in termini di prestazioni come nel caso di fornitura di un delta ma tale meccanismo, basato sul concetto di hash del dato, permette di realizzare una storicizzazione di sistema al dato fornito.

L’estrazione delle fonti dati dovrà essere in formato txt con tipi record, al netto delle fonti dati delle agenzie nazionali.

* Fase di Estrazione
  + Le modalità operative della fase devono essere concordate di volta in volta con i fornitori dei sistemi gestionali
  + Il risultato dell’estrazione è un file di formato concordato registrato nel repository delle forniture
  + Se l’estrazione concordata non rispetta il formato standard di piattaforma si è in presenza di una fornitura custom che necessita di configurazione dell’ambiente di acquisizione
* Fase di Reperimento
  + La piattaforma è in grado di reperire (realizzando autonomamente la fase di estrazione) i dati direttamente presso la fonte (gestionale legacy o esportazione locale del dominio dati interessato) attraverso un modulo di estrazione configurabile. I dati estratti possono essere di tipo massivo o differenziale, dipendente dalla qualità del dato (presenza o meno di chiavi primarie identificative) e dalla complessità della fonte.
  + Processi di reperimento custom possono essere implementati attraverso le interfacce API della piattaforma e istallati nel modulo di piattaforma adibito all’orchestrazione dei processi.
  + Il risultato della fase di reperimento è sempre una o più tabelle nello Staging DB.
* Fase di Caricamento
  + Nel caso in cui a) il reperimento venga svolto in modalità custom b) il file txt estratto sia di tipo non standard : il caricamento della fornitura prevede una attività preliminare unatantum di configurazione o programmazione.
  + In presenza di forniture standard la fase di caricamento registra nel datawarehouse di I livello la fornitura e la rende disponibile per i processi di II livello

La tipologia di flusso dati in fase di caricamento dipende strettamente dal formato della fornitura , dalla modalità di estrazione e di reperimento e dalla configurazione scelta in fase di impianto della piattaforma:

 caricamento nel Dwh direttamente dal repository delle forniture (per fonti dati con forniture già strutturate in semi-relazionale). Alcune forniture esterne (per es. successioni e compravendite) , possiedono una struttura xml semi-relazionale in virtù della quale possono saltare la fase di staging ed essere introdotte con un processo transazionale direttamente nel Dwh diI livello.

 caricamento attraverso l’utilizzo dello staging db da parte del processo di caricamento. Deposito di fornitura “grezza” e successivo trattamento per l’inserimento in Dwh.

 Caricamento dal database di staging valorizzato dai processi di reperimento.

* Fase di Estrazione Informazioni
  + Contiene l’insieme dei processi di elaborazione che a partire dal data warehouse costruiscono il set di informazioni utili alle esigenze operazionali degli uffici
  + La modalità di utilizzo delle informazioni di II livello potrà essere
    - Accesso web a una specifica informazione, diagnostica o datamart funzionale
    - Esportazione file
    - Reportistica
    - Navigazione integrata attraverso indice di correlazione
    - Le informazioni sono anche disponibili ai servizi di front-end del progetto

### Staging Locale

E’ costituito da un database opensource la cui applicazione è stata appositamente pensata per il recupero delle informazioni direttamente presso la fonte dati, eliminando in questo modo la necessità di estrattori sugli operazionali dell’ente. Attivando il db di staging locale l’ente realizza un estrattore ad hoc i cui dati andranno censiti e configurati nella piattaforma nella fase di caricamento.

### Repository forniture

E’ rappresentato da un file system accessibile dai componenti della piattaforma.

Una struttura tipo di repository , in una istallazione di piattaforma a servizio di più organizzazioni è del seguente tipo:



Ogni fornitura ha un formato specifico che in alcuni casi a definito a livello nazionale, per le forniture interne all’ente è stabilito un formato standard di piattaforma.

### Staging Db

E’ costituito da un RDBMS relazionale contenente delle flat table con struttura identica alle forniture originali, al netto di alcuni attributi di sistema utili all’identificazione del processo ETL. Lo staging sb contiene per ogni data source, nel tempo, tutte le forniture ingessate in piattaforma con l’indicazione di un flag “caricato s/n”.

Il numero di forniture ingessate nel database di staging e presenti nel Dwh dovrebbe essere sempre lo stesso in un sistema attivo ed allineato.

Database DWH I Liv, Storico Diagnostiche, Datamart, Indice di Correlazione: Database relazionale

### Architettura del data warehouse di I Livello

L’architettura dati della piattaforma è quella di un database relazionale disegnato secondo i principi del data warehousing.

Il database è disegnato secondo i principi del snowflake schema , strettamente correlato allo star schema questo tipo di disegno permette la normalizzazione delle dimensioni di data warehousing in tabelle multiple correlate.

In pratica la banca dati risulta una banca dati relazionale nella quale le relazioni rappresenteranno le dimensioni di una entità fatto centrale rispetto allo schema.

Con questo tipo di disegno relazionale, semi normalizzato , ogni dimensione potrà essere considerata un fatto a seconda delle domande che si porranno alla banca dati.

Di seguito un confronto con il più famoso “star schema”:

FATTO

Dimensione

Dimensione

Dimensione

Dimensione

Star Schema

FATTO

Dimensione

Dimensione

Dimensione

FATTO

Dimensione

Dimensione

Snowflake Schema

Dimensione

FATTO

Ogni tabella ha una dimensione temporale fornita dall’uso di attributi

|  |  |  |
| --- | --- | --- |
| DATA\_INIZIO\_VAL | Data inizio validità del dato | * Se presente l DT\_INI\_VAL\_DATO assume tale valore * Assume il valore DT\_ELAB in caso contrario |
| DATA\_FINE\_VAL | Datafinale di validità del dato | Valorizzata in fase di storicizzazione di una stessa istanza di dato pervenuta in fase di aggiornamento della fonte – assume un valore applicativo DT\_FINE\_VAL\_DATO se presente alla fonte , altrimenti assume il valore della data di sistema |
| DT\_INI\_VAL\_DATO | Data inizio validità nativa dell’informazione presso la fonte | passata al processo ETL di I livello dalla fonte dati (opzionale) |
| DT\_FINE\_VAL\_DATO | Data fine validità nativa dell’informazione presso la fonte |
| DT\_ELAB | Data elaborazione | Data di esecuzione del processo ETL |
| PROCESSID | Identificativo del processo ETL | Tramite questo identificativo è possibile reperire tutte le informazioni pervenute durante un singolo processo ETL |

La dimensione temporale delle entità risulterà utile al fine di ricostruire la storia di una singola informazione.

Alternativa teoriche allo snowflake schema possono essere:

* L’uso di uno schema stella
* Il disegno relazionale della banca dati delle fonti
* L’import delle fonti dati “as is” (flat schema)

In realtà lo schema scelto è un buon compromesso, riesce ad affrontare in modo corretto le problematiche di : dimensione eccessiva della banca dati (schema a stella), relazioni non generabili a causa della qualità dei dati(db relazionale), difficoltosa interpretazione dei dati (flat schema).

### Architettura del data warehouse di II Livello

Il secondo livello Dwh è costituito da un set di tabelle disegnate con differenti modelli di banca dati a seconda delle tipologie di risposte che la piattaforma deve fornire.

Uno di questi modelli è lo star schema ma sono presenti anche modelli relazionali semplici con le tabelle in seconda e terza forma normale.

Questo livello di informazioni, preso atto della tipologia di dati ingessati e della eterogeneità delle codifiche e degli attributi anche riferiti ad una stessa classe di informazioni, deve essere in grado di fornire all’utente gli strumenti necessari per il proprio lavoro , non introducendo d’altro canto ulteriori effort e dispendio di risorse per la gestione del dato (si pensi a piccole realtà comunali) ma anzi, agevolando supportando il lavoro quotidiano d’ufficio con informazioni aggiuntive e aggregate non reperibili e difficilmente organizzabili senza l’utilizzo della piattaforma. A tale scopo le strutture dati fornite permetteranno di:

* Controllare la qualità dei dati i n proprio possesso
* Svolgere funzioni di controllo amministrativo e avere un supporto alla gestione delle pratiche d’ufficio
* Adempiere ad obblighi legislativi
* Ricercare informazioni in modo trasversale all’interno del patrimonio informativo dell’Ente
* Gestire pratiche di accertamento

## Architettura Software

### Modello di architettura distribuita

L’architettura software del sistema è di tipo multilivello distribuita. La piattaforma può essere istallata in modo distribuito, diversi host per moduli differenti. Ogni modulo di piattaforma è costituito da un applicativo enterprise multilivello.

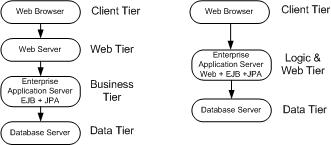
Il linguaggio di implementazione principale dei componenti di sistema è Java su piattaforma J2EE.

J2EE è l’acronimo di Java 2 Platform Enterprise Edition. Ambiente server applicativo di Sun Microsystems per lo sviluppo di applicazioni distribuite. La Piattafaroma J2EE consiste di un insieme di servizi , Api e protocolli, che forniscono funzionalità per lo sviluppo di applicazioni Web multilivello.

Le tecnologie scelte per i vari layer applicativi sono: Jsf per lo strato Web , EJB per lo la logica di business, JPA per il persistence layer, RBBMS relazionale per il database.

I sistemi distribuito offrono una maggiore accessibilità ai servizi offerti e permettono di avere oggetti che forniscono servizi ubicati su una macchina che vengono resi disponibili ad altri oggetti ubicati su macchine diverse.

La modularità della piattaforma permette l’istallazione su sistemi 3-tuier o 4-tier come mostrato in figura.



O addirittura a 5 livelli nel caso il persistence layer venga istallato come modulo a se stante.

A livello client è utilizzato un web browser che renderizza pagine di tipo Java Server Faces generate dal livello applicativo web gestito da un web container.

La natura enterprise dei moduli software, implementati su tecnologia EJB, ne permette diversi schemi di deploy e consente una elevata adattabilità alle risorse hardware disponibili.

Per evidenziare quale è stato il percorso progettuale che ha portato alla definizione dell’architettura software si riportano nei paragrafi alcune decisioni di design.

### Scelte inerenti lo strato di accesso ai dati

**Tipologie di accesso ai dati**

**Accesso alle fonti dati (driver di estrazione)**

I driver di estrazione delle fonti dati (utilizzati per situazioni custom o in assenza di estrazioni complaint degli applicativi legacy) sono implementati come componenti Java standard accessibili tramite Servlet/Jsp o da scheduler di back-end. Non necessitano dunque di application server Enterprise.

**Caricamento dati nel data warehouse I liv**

Questa operazione impone un numero elevato di accessi in lettura e scrittura e logiche di trattamento delle informazioni (utilizzo di algoritmi di hash, scritture massive di record ) che impongono l’utilizzo di connessioni dirette jdbc alla banca dati senza strato ORM. Lo strato ORM è assente da questo strato soprattutto in virtù della natura dinamica delle fonti dati da caricare che non permette l’utilizzo dichiarativo di librerie ORM.

**Accesso al dato da verticale di servizio**

Si è scelto di utilizzare JPA, nell’implementazione fornita dalla librerie Hibernate, come strato di persistenza utilizzato dai verticali web di servizio per accedere ai dati del data warehouse (fonti a sistema).

**Creazione DWH II liv**

I vari componenti sono stati implementati come librerie a plug-in , ogni plug-in definisce al proprio interno la tecnologia per la persistenza delle informazioni.

**Gestione delle tabelle di sistema**

Il modello di accesso alle tabelle di sistema della piattaforma è dababase-indipendent basato su JPA.

### Sicurezza

La sicurezza della piattaforma si basa sui concetti di autenticazione e autorizzazione. Entrambe le funzioni vengono svolte da uno stesso componente software che accede ad un database RBAC (basato sui ruoli).

Il componente centralizzato fornisce le interfacce per la sicurezza sia per i servii di front-end che di back-end.

### Pattern utilizzati

Per lo sviluppo software sono utilizzati i seguenti Design Pattern

**MVC:** Per implementare questo pattern viene utilizzato il framework JSF. “JavaServer Faces (JSF) è una tecnologia Java basata sul design pattern architetturale Model-View-Controller (MVC). Il suo scopo è di semplificare lo sviluppo dell' interfaccia utente (UI) di una applicazione Web; può quindi essere considerata un framework per componenti lato server di interfaccia utente”

**SESSION FACADE:** In particolare viene implementato un EJB session Facade per includere la logica degli Entity all'interno di uno o più Session Beans. In questo modo si ha il vantaggio che il Client è indipendente dagli Entity e inoltre può essere garantita l’integrità delle transazioni per le operazioni che richiedono diversi accessi al database.

**DATA TRANSFER OBJECT:** Per limitare il numero di chiamate sulla rete sono sviluppati degli oggetti che contengono tutte le informazioni necessarie ad un particolare utilizzatore. Ad esempio una pagina Web anziché avere più chiamate per ottenere diversi Entity possiede un'unica chiamata che ritorna un Java Bean che contiene ed incapsula i dati in un unico oggetto (DTO) “trasportabile” attraverso la rete.

**IOC (Inversionon Of Control):** E’ il pattern implementato dalla libreria Spring per l’iniezione di componenti all’interno di altre classi.

### Componenti software

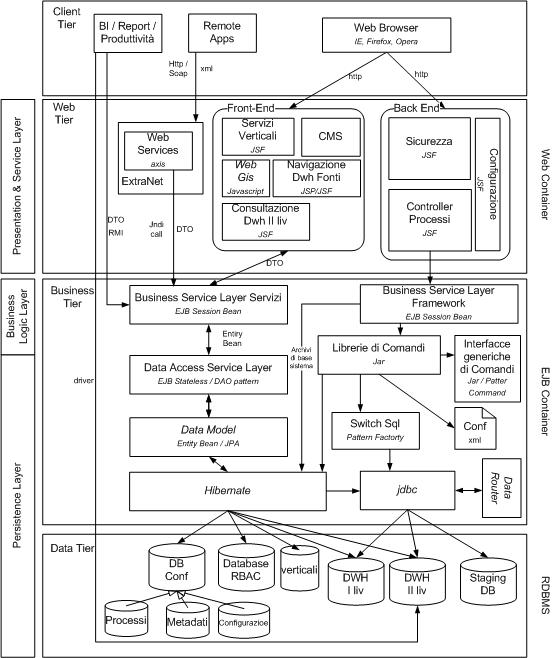
I componenti di business della piattaforma si dividono in differenti tipologie di componenti.

In un’ottica di istallazione distribuita ogni componente realizza un pattern architetturale e adotta determinate tecnologie in base alla propria funzione.

|  |  |
| --- | --- |
| Componenti per il trattamento delle fonti e il caricamento dei dati | pattern Command - libreria |
| Componenti per la gestione della sicurezza | Database di tipo RBAC – Enterprise application |
| Componenti per la gestione della configurazione e il reperimento di metadati | Session Facade – libreria EJB |
| Componenti per l’orchestrazione dei processi | Session Facade - Command – Enterprise Application |
| Componenti di accesso al data warehouse e servizi verticali | MVC - Enterprise Application |

### Schema architettura

Sulla base del modello architettura a 4-tier si riporta nello schema seguente l’architettura software della piattaforma.



Il **Client Tier** è costituito da tutti quegli strumenti e applicazioni che consentono di accedere ai servizi esposti dalla piattaforma nei diversi protocolli di comunicazione utilizzati. La natura standard ei protocolli non impone una architettura hardware o software specifica a questo livello.

http : vengono fornite le pagine web degli applicativi rivolti all’utente, mediante lo stesso protocollo transitano inoltre i messaggi Soap dei web services esposti (http/Soap)

RMI: Gli EJB si appogiano a RMI per esporre i propri servizi, aggiungendo però una gestione automatica da parte del container di transazioni, scalabilità e sicurezza. Attraverso il protocollo RMI, evoluto recentemente nell’RMI-IIOP, Sun ha introdotto una compatibilità con il modello Corba utilzzato a piattaforme Enterprise di altri vendors. Non è previsto dalla piattaforma esporre servizi di tipo RMI-IIOP esternamente alla Lan di impianto, questo perché si restringerebbero le possibilità di cooperazione a piattaforma Enterprise, ma nulla vieta comunque che soluzioni personalizzate di accesso diretto al livello di business possano essere implementate.

Database Driver: un driver è uno strato software che funge da connettore e che permette ad un client di accedere ad una banca dati. Molti strumenti di produttività desktop , reportistica e business inteligence utilizzano driver per accedere alla base di dati di interesse.

Nel **Web Tier** risiedono le interfacce web e i servizi esposti al Client Tier attraverso i protocolli supportati.

Con interfacce utente di front-end si intendono tutti quei servizi web (web application) che svolgono compiti funzionali e legati allan avigazione, interpretazione, lettura, esportazione dei dati del data warehouse, in generale dunque alla fruizione di tali dati.

Per interfacce di back-end si intendono tutti quegli strumenti web adibiti alla gestione della piattaforma in termini di popolamento, configurazione , profilazione.

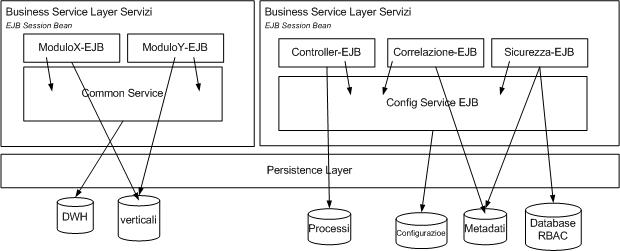
Anche i web services (realizzati attraverso la libreria Apache Axis) fanno parte del Web Tier anche se si può ipotizzare di impiantare determinati servizi in extraNet per renderli fruibili a soggetti esterni all’Ente (per esempio livello regionale che richiede info su residenti al livello comunale).

A questo stato dell’architettura è presente un web container (che può anche coincidere fisicamente con l’EJB Container realizzando una architettura a tre strati) che fornisce i servizi web al primo livello.

Il **Business Tier** è il livello maggiormente ingegnerizzato della piattaforma e con la maggior presenza di pattern e frame work. Al fine di ottimizzare ogni singola funzionalità di piattaforma il livello business non è costituito da un monoblocco Ejb ma da differenti strati di servizi:

* Il Business Service Layer Servizi è composto da un layer di servizi di tipo Session Bean Stateless e da n layer di servizi relativi ognuno ad un verticale di front-end
* Il Business Service Layer Framework similmente è costituito da un modulo Ejb di accesso alla configurazione utilizzato da tutti gli applicativi di back end e da più moduli verticali , uno per ogni macro-funzione di back-end

Nella figura seguente si evidenzia, considerando trasparante il layer di persistenza, il legame fra moduli di business e banche dati.



L’architettura modulare progettata a livello dello strato di business permette alla piattaforma di identificare i componenti applicativi di ogni servizio, tenere sotto controllo le dipendenze dei moduli ed esporre dunque funzionalità con una granularità fine.

Una considerazione la merita inoltre il componente software delle Librerie di Comandi che attraverso l’implementazione del potentissimo pattern Command permette all’utente amministratore di implementare classi di comandi attraverso due semplici step:

* Implementazione di una interfaccia di tipo Command
* Definizione del comando su file properties e su database relazionale
* Realizzazione dello script di esecuzione del comando con linguaggio Xml-like Jelly

il componente software citato rappresenta il fondamentale strumento per il mantenimento dell’integrità e della consistenza dei dati della piattaforma. L’interfaccia di tipo Command infatti permette una certa libertà implementativa ( a valle del Command sono molto spesso presenti classi che utilizzano altri pattern o strumenti) a supporto della scalabilità e della manutenzione della soluzione.

La piattaforma è di tipo multiente, nel senso che una stessa istallazione può servire n clienti e dunque è adatta ad essere impiantata in una realtà di polo catastale o aggregazione di comuni. Questa caratteristica è assicurata dalla libreria Data Router che permette di gestire un numero elevato di data source e permette ad hibernate e allo strato jdbc in generale di essere “instradato” verso la connessione corretta. Il modulo Data Router utilizza la libreria Spring Bean per realizzare il pattern IoC (Inversion of Control).

 Data Router

Il **Data Tier** è composto dal DBMS relazionale con i seguenti requisiti minimi:

* Indicizzazione su campi composti e su funzioni
* Viste o tabelle di metadati accessibili con Sql
* Attributi indicizzazione di dati spaziali
* Creazione di sinonimi
* Creazione di viste
* Operazioni transazionali e gestione rollback
* Presenza di driver jdbc di tipo 4 (indipendente dalla piattaforma)

# Modello dei Dati

Considerando le evoluzioni in materia di ruolo e compiti nella gestione riconosciuti ai Comuni, attraverso un insieme di norme che a partire dagli anni ‘90 (anche se esistono provvedimenti di molto antecedenti, ma poco o nulla interpretati) hanno modificato il rapporto amministrativo tra Ente locale e Comunità amministrata, si sono un insieme di contenuti informativi di interesse.

In tale verifica sono state tenute in conto le norme salienti dei rapporti del Comune con l’Agenzia del Territorio (es. norme L.662/96, L.311/04, L.80/06, ec…) e dello stesso con L’Agenzia delle Entrate (partendo dai disposizioni delle finanziarie 2005 -….- 2011, che rivedono e perfezionano quanto in parte già stabilito alla fine degli anni 70’ dai decreti Stammati), oltre a considerare le nuove necessarie forme di organizzazione del lavoro interne all’Ente che richiedono l’intersettorialità tra uffici.

Alla luce di queste considerazione è stata prevista la presenza, come fonti informative del DWH, delle seguenti banche dati:

*Sistema Informativo comunale*

* Anagrafe della popolazione
* Tributi ICI, TARSU e COSAP
* Licenze di commercio
* Pratiche edilizia privata
* Toponomastica e censimento civici
* Multe e Verbali
* Servizi comunali alla persona

*Agenzia del Territorio SISTER*

* Catasto fabbricati e terreni impianto (consigliato dal 01/01/1995) e aggiornamenti ad oggi
* Carta CXF attuale
* file comunicazioni ai comuni atti compravendite dal 2006
* file comunicazioni docfa presentati dai professionisti dal 2006 o ultimo anno (gli anni precedenti sono ritirati dal sito se il Comune non li ha scaricati). Per questa fattispecie occorre fare una verifica per Comune in relazione allo stato degli scarichi ed alla presenza del CD gennaio – ottobre 2006
* dati misurazioni superici per tipologia di vano c.340, scarico ai fini TARSU (c.340 L.311/04)
* Tabella OMI valore immobili per microzona, da Osservatorio, ai fini applicazione c.336 L.311/04. Contiene le tariffe di estimo per le categoria abitative e le classi minime per le categorie C.
* Planimetrie se richieste a parte dai Comuni, in quanto non previste nello scarico standard da portale (escludendo chiaramente quelle provenienti con i docfa)

*Agenzia delle Entrate SIATEL*

* Redditi dalla data di disponibilità sul Sito
* Successioni
* Localizioni
* Forniture contratti ENEL elettrico da data di disponibilità sul Sito
* Forniture contratti ENEL gas da data di disponibilità sul Sito

*Gestore della riscossione*

* Scarico flusso della riscossione a Ruolo, secondo modello CNC attuale: flusso di richiesta di messa a ruolo da parte del Comune (tracciato 290), trattamento del riscossore sul conferimento a ruolo della riscossione (tracciato 750 o 290 arricchito), stato della riscossione.

Queste fonti informative costituiscono il DWH di primo livello e il conseguente modello dell’architettura dati.

Nei paragrafi seguenti viene fornita una descrizione del modello concettuale dei dati come è stato articolato per fonte, la metodologia adottata per la realizzazione del DB.

## Schema Dati

### Schema Dati Staging DB

Lo schema dati del database di staging rispecchia fedelmente la struttura delle forniture dati , sia esterne (definite a livello sovra-comunale ) che interne (definite nell’ambito del progetto).

### Schema Dati DWH I livello

Sulla base delle fonti sopra descritte, il modello dati del DWH ha tenuto conto di una serie di aspetti derivati dai contenuti presenti nei flussi:

* Ogni fonte ha un flusso di informazioni che descrive il territorio sulla base del compito svolto dall’operazionale di riferimento;
* La descrizione del territorio insiste sui paradigmi di “Soggetto”, “Oggetto”, “Toponomastica” che in tutto o in parte sono ritrovati in tutte le fonti;
* La presenza di uno o più paradigmi in ogni fonte rende l’informazione quasi sempre ridondante tra le fonti e quindi nel DWH che per tale motivo NON può e NON deve essere pensato in terza forma normale;
* L’interesse dal DWH è quello di conservare ed organizzare l’informazione presente nelle fonti, gestendo la ridondanza e etichettandola con la fonte di appartenenza;
* La ridondanza dell’informazione deve essere gestita attraverso il concetto di storicizzazione dei dati delle fonti che arrivano periodicamente, assicurando l’aggiornamento del DWH al passare del tempo ed all’incedere degli eventi;
* La presenza di più tabelle del DWH per la stessa informazioni di fonti diverse deve essere organizzata dal Progetto in modo da esaltare il singolo significato e la differente valenza informativa che ogni fonte dà al proprio dato, anche in situazione di valenza informativa similare (es. titolare a catasto di un immobile e dichiarante ICI per lo stesso immobile nel Comune);
* L’organizzazione del dato deve tenere conto dei requisiti di risposta previsti per questo tipo di piattaforma, dove è ESSENZIALE la distinzione tra i contenuti delle fonti e la velocità di reperimento confronto e esposizione dell’informazione. Questo prevede da un lato una determinata modalità di progettazione , dall’altro la predisposizione di una architettura che consenta di creare a livello più alto le informazioni contenute del DWH e provenienti dalle diverse fonti.

A seguire viene affrontato il tema dei contenuti informativi che popolano il DWH secondo modelli ER. Nella rappresentazione si sono evidenziate le entità in cui sono strutturate le informazioni delle fonti, che organizzano il modello dell’informazione presente nel DWH e contestualmente esplicitano le peculiarità di questo progetto derivanti dalle considerazione sopra espresse.

Nella descrizione ogni fonte è stata organizzata raccogliendo le informazioni in una o più entità, classificata/e nel contesto delle TRE chiavi attraverso cui si è deciso di identificare quella informazione: *Soggetto*, *Oggetto/fabbricato*, *Indirizzo*.

Il risultato complessivo per ogni fonte è il seguente:

* Ogni informazione (campo / contenuto informativo elementare) dell’unità informativa del flusso (record / descrittore consistente del costrutto informativo gestito) è collocata all’interno di un contenitore (entità) etichettato a livello architetturale come riferito ad una delle tre chiavi primarie del sistema. Questo pone in diretta relazione i campi del record inseriti nella Entità con l’Indice di correlazione (livello superiore del DWH di raccordo dell’informazione, dove trova spazio un modello basato su schemi a stella per le informazioni e costrutti Entity Relationship per i legami di servizi tra i dati);
* Ogni fonte, attraverso questo modello può avere raggruppate le sue informazioni all’interno di Entità che sottostanno a una o più chiavi dette di “*correlazione*”. Attraverso questo approccio l’accesso all’informazione avviene partendo da una chiave ottenuta dalla combinazione dei campi presenti nell’entità della fonte ( in questo contesto le chiavi possono essere semplici o complesse, definite come aggregazione di informazioni che identificano in modo univoco una di esse. Ad esempio la chiave soggetto può avere come identificativo banalmente il codice fiscale, presente nella entità “Persona” della fonte Anagrafe, ma anche i dati di nascita, più semplicemente quelli di residenza, in taluni casi potrebbe essere preso in considerazione il nome e il codice famiglia, ecc…);
* La combinazione dei campi dei campi può consentire una definizione univoca dell’elemento cercato o non univoca, come nel caso di fonti con informazioni incomplete o parziali. La discriminante in questo caso è l’utilizzo dell’informazione da parte dell’utente, al Sistema spetta il compito di SEGNALARE il livello di consistenza e di unicità dell’informazione, all’utente quello di decidere l’uso;
* Il risultato complessivo di gestione del dato è di estrema flessibilità, di facilità di gestione della storicizzazione, di collegamento immediato con la fonte originaria salvaguardando integralmente il collegamento tra fonte e informazione trasferita;
* Alta flessibilità nel trattamento delle informazioni in fase di consultazione, ma anche nel supporto, controllo e monitoraggio nel contesto di operazioni di bonifica;
* Estrema facilitazione nelle operazioni di aggiunta di una nuova banca dati nel Sistema. Basta analizzarla costruendo il diagramma ER raggruppando i campi all’interno di Entità “incasellate” nei tematismi di correlazione e parametrizzare l’indice di correlazione per “mappare” le chiavi per i singoli tematismi.

Di seguito la schematizzazione delle fonti sopra indicate.

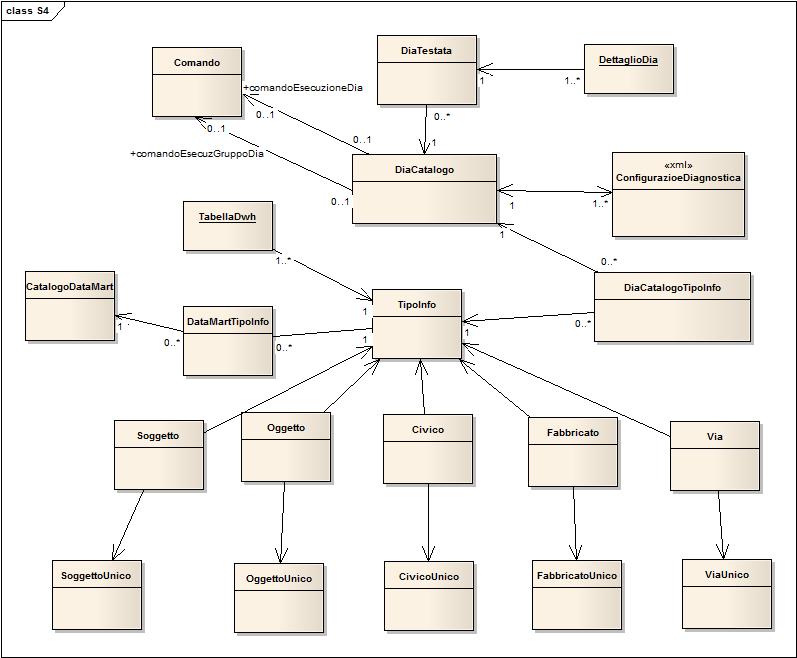
Nella sezione successiva viene descritta la metodologia di progettazione utilizzata e il modello dell’architettura dati nel suo complesso. La descrizione funzionale della piattaforma e quindi dei componenti applicativi di gestione di queste architetture è stata peraltro già data nella sezione dedicata al modello logico del Sistema.



### Schema Dati DWH II livello

Il database di II livello è strutturato principalmente in tre componenti: diagnostiche, correlazione, datamart tematici.

Coerentemente con la metodologia di progettazione adottata, dal modello concettuale dell’ambiente delle diagnostiche si è generato un class diagram rappresentante il modello dati ad alto livello comprendente le tre componenti architetturali.



Il framework architetturale proposto è predisposto attraverso interfacce di configurazione e programmatiche ad ospitare viste, elaborazioni, statistiche attraverso un sistema di tabelle di metadati che permettono di censire, istanziare e tipizzare ogni richiesta dell’utente verso il patrimonio informativo della piattaforma.

|  |  |
| --- | --- |
| **Dizionario dati diagnostiche** | |
| **DiaCatalogo** | Contiene l’anagrafe delle diagnostiche e rappresenta la tabella dei metadati contenente le definizioni di ogni diagnostica in termini di:   * Specifica della categoria di diagnostiche standard alla quale appartiene la singola diagnostica (configurazione di una nuova diagnostica standard) * Componenti sw custom o complessi che realizzano la funzione diagnostica evoluta * Modalità di esecuzione della diagnostica (singola / gruppo di diagnostiche) * Abilitazione / disablitazione diagnostica |
| **DettaglioDia** | Il dettaglio di ogni diagnostica ha una struttura propria , nello schema si è scelto di riportare genericamente la dicitura DettaglioDia vista l’impossibilità di prevedere la natura delle diagnostiche che si andranno a generare in fase di analisi di dettaglio |
| **ConfigurazioneDiagnostca** | Rappresenta il repository xml di definizione dei componenti sw che realizzano la diagnostica e di censimento delle tabelle risultato diagnostica generate. |
| **DiaCatalogoTipoInfo** | Contiene la tipizzazione degli attributi che costituiscono la relativa tabella dei risultati diagnostici (DettaglioDia).  Per ogni diagnostica è possibile specificare la tipologia di attributi che contiene. Per esempio è possibile specificare quale attributo rappresenta una chiave di una fonte dati X così da dinamicizzare e permettere l’accesso al dato diagnostico durante l’interrogazione della fonte dati X. La tabella permette di creare n entrypoint sul dato diagnostico e dunque di definire relazioni dinamiche fra i dati. |
| **TestataDia** | Contiene l’elenco storicizzato di tutte le esecuzioni diagnostiche. |

|  |  |
| --- | --- |
| **Dizionario dati di frame work**  Nello schema sono riportate alcune tabelle di frame work , quelle che concorrono alla realizzazione dello strato del database di II livello | |
| **Comando** | Rappresenta l’anagrafe dei Comandi di esecuzione della piattaforma attraverso la quale vengono avviati tutti i processi o librerie di processi. |
| **TipoInfo** | Anagrafe delle classificazioni informative suddivisa per fonte dati di appartenenza. Un esempio di classificazione:  Fonte Dati : Anagrafe  TipoInfo: Persona, Civico  Fonte Dati: Concessioni Edilizie  TipoInfo: Pratica, Richiedente, Impresa, Progettiste  Le classi di informazione censite vengono mappate in piattaforma alle tipologie di dati Soggetto, Oggetto, Fabbricato, Via, Civico |
| **TabellaDwh** | E’ l’istanza di una tabella standard del data warehouse contenete tutti gli atributi per l’identificazione della classe di informazione e per la gestione della storicizzazione dell’informazione |

|  |  |
| --- | --- |
| **Datamart Tematici** | |
| **CatalogoDataMart** | Contiene l’anagrafe dei DatMart come estensione dell’anagrafe delle diagnostiche.  Sono specificate le caratteristiche di ogni Data Mart in termini di **fatti, dimensioni, misure.**  Vista la configurabilità richiesta e la possibilità di generare data mart “a runtime” la soluzione impone che la tabella contenga i **metadati** di ogni data mart.  A differenza delle diagnostiche che possiedono una logica di tipo push e la cui esecuzione è dunque legata ad eventi che rigurdano la fonte dati , i data mart possono essere anche di tipo pull allorchè costituiti da elaborazioni poco dispendiose o addirittura da viste.  Per i data mart di tipo push vengono specificate le regole di aggiornamento (tempistica, eventi ecc.) |
| **DataMartTipoInfo** | In modo simile all’ambiente delle diagnostiche anche i dati contenuti nei data mart sono correlati all piattaforma attraverso l’indicazione delle caratteristiche o attributi di correlazione facenti riferimento alle classi informative censite nel framework. |

|  |  |  |
| --- | --- | --- |
| **Indice di Correlazione** | | |
| **Soggetto** | Contiene il riferimento alle righe delle tabelle del data wearehouse che posiedono una informazione di tipo “soggetto”. | Ogni riga possiede un riferimento alla classe di informazione trattata. Per esempio MARIO ROSSI potrebbe essere presente n volte all’interno della tabella **Soggetto** ma con ruoli differenti (Residente, Progettista, Locatario ecc.) |
| **Oggetto** | Contiene il riferimento alle righe delle tabelle del data wearehouse che posiedono una informazione di tipo “soggetto”. |
| **Fabbricato** | Contiene il riferimento alle righe delle tabelle del data wearehouse che posiedono una informazione di tipo “fabbricato”. |
| **Via** | Contiene il riferimento alle righe delle tabelle del data wearehouse che posiedono una informazione di tipo “via”. |
| **Civico** | Contiene il riferimento alle righe delle tabelle del data wearehouse che posiedono una informazione di tipo “civico”. |
| **SoggettoUnico** | Contiene la lista dei soggetti presenti all’interno delle fonti dati in modo univoco. La tabella è il risultato del processo di aggregazione delle fonti basato su criteri multipli di aggregazione | I record contenuti nelle tabelle provengono dai record delle tabelle sopra elencate, attraverso un ordine di inserimento che si basa sul concetto di attendibilità del dato (impostato in fase di impianto del sistema e valutato dal processo di aggregazione a run-time) |
| **OggettoUnico** | Idem per ogetto (unità immobiliare) |
| **FabbricatoUnico** | Idem per fabbricato (edificio) |
| **ViaUnico** | Idem per via o piazza |
| **CivicoUnico** | Idem per Civico |