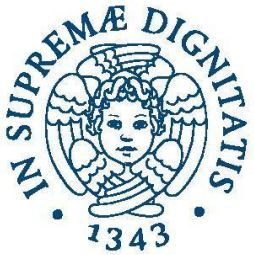


UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA
Corso di Laurea Magistrale in
Informatica per l'Economia e per l'Azienda
(Business Informatics)

Tesi di Laurea Magistrale

PROGETTAZIONE DI UN DATA WAREHOUSE PER IL SUPPORTO AL CONTROLLO QUALITATIVO DELLA PRODUZIONE DI INSULINA

Relatore:
Franco Turini

Candidato:
Alessandro Croce

Tutor aziendale:
Marco Luchini

Matricola:
444474

Anno Accademico 2015-2016

Riassunto

L'obiettivo di questo lavoro di tesi è quello di realizzare uno strumento a supporto del processo di controllo della qualità per quanto riguarda la produzione di insulina all'interno di un'azienda multinazionale dell'industria farmaceutica.

Lo scopo finale è quello di fornire agli utenti dei report uno strumento che certifichi la bontà del farmaco prodotto al fine di decidere la sua immissione sul mercato. Nel particolare l'attenzione di questo lavoro verte sui report che trattano sull'esecuzione dei cicli di pulizia e sterilizzazione che riguardano i macchinari coinvolti nella fase di Formulazione, la fase iniziale del processo produttivo di insulina. Il lavoro svolto prevede un'introduzione sul caso di studio, le attività di progettazione e implementazione del *data warehouse*, una trattazione sugli strumenti utilizzati, fino ad arrivare al processo di elaborazione dei dati in ingresso e alla rappresentazione grafica delle informazioni agli utenti.

Indice

Elenco delle Figure

Elenco delle Tabelle

1 INTRODUZIONE

1.1 Presentazione del Problema

La qualità dei prodotti destinati alla distribuzione al pubblico, al giorno d'oggi, rappresenta il principale fattore di successo di un'azienda. Per le aziende che lavorano nel settore farmaceutico questo fattore diventa fondamentale in quanto dalla qualità del prodotto dipende anche la salute dei loro clienti, tanto da avere una specifica area dedicata allo svolgimento di questo compito, chiamata appunto Controllo Qualità (abbreviato come QC). Ciò che viene prodotto, infatti, deve seguire delle linee-guide obbligatorie con requisiti minimi per i produttori. Il cliente, paziente in questo caso, non è un vero e proprio consumatore in quanto spesso è costretto ad utilizzare uno specifico farmaco per raggiungere la guarigione. È quindi molto importante garantire al cliente un prodotto che sia di alta qualità. Generalmente per il cliente i difetti di un prodotto farmaceutico non sono di facile individuazione, e le conseguenze di un prodotto non conforme a ciò che ci si aspetta potrebbero essere molto gravi.

Per raggiungere questi obiettivi, riducendo al minimo il margine di errore, acquistano grande importanza le informazioni derivanti dall'acquisizione di dati in formato elettronico. La raccolta e la corretta interpretazione dei dati ha permesso alle aziende di avvantaggiarsi sul mercato e di garantire ai propri clienti un prodotto sicuro e di alta qualità. Queste azioni si traducono con il monitorare e l'analizzare i dati informatici che provengono dall'intero processo produttivo aziendale. Acquistano fondamentale importanza l'integrazione e la veridicità dei dati raccolti oltre alla possibilità di analizzarli in tempi brevi. Una volta pervenuti e preparati i dati, è importante interpretare correttamente gli stessi. Nel caso di aziende multinazionali i dati infatti provengono verosimilmente da più stabilimenti dislocati per il globo, e quindi sono spesso diversi tra di loro per il modo in cui vengono raccolti pur riflettendo le stesse informazioni, per questo motivo è importante che siano uniformati e ben interpretati. In questo contesto la *Business Intelligence* diventa necessaria sia per una corretta interpretazione dei dati sia per un rilascio in tempi brevi di report da consegnare all'utente finale.

Le attività analitiche per comprovare la qualità del prodotto sono condotte sulla base di specifiche e metodi approvati. Ogni attività e ogni dato ottenuto deve essere registrato in tempo reale per una completa tracciabilità delle operazioni svolte.

1.2 Rassegna della Letteratura

Nella stesura del presente lavoro di tesi sono state consultate numerose fonti. Una parte sostanziosa del lavoro è basata sul contenuto di *Basi di Dati di Supporto alle Decisioni* di Antonio Albano [Albano 12], particolarmente adatto come guida alla realizzazione di progetti di *data warehousing* in quanto sono presenti concetti teorici e pratici

sull'argomento trattato. Le informazioni relative alle società sviluppatrice nel progetto sono state estratte dal sito web ufficiale [SDG Group 16].

Le nozioni sull'industria farmaceutica sono state estratte dal lavoro di Sarah Holland e Bernardo Batiz-Lazo *The Global Pharmaceutical Industry* [Holland 04]. Per le caratteristiche dell'ambiente di sviluppo e dei *tool* utilizzati, per le procedure di estrazione, trasformazione e caricamento e per la reportistica viene fatto riferimento rispettivamente alla documentazione ufficiale Informatica PowerCenter [Informatica 13], e SAP Business Object [SAP 12]. Ciò che riguarda i concetti più teorici inerenti la fase di ETL si basa sul lavoro di Ralph Kimball e Joe Caserta *Practical Techniques for Extracting, Cleaning and Delivering Data* [Kimball 04]. Tutte le informazioni inerenti l'azienda cliente, a causa di motivi legati alla violazione della *privacy*, non sono referenziate. Si tratta di informazioni relative alla struttura del *data warehouse*, dei sistemi sorgente e delle logiche produttive che erano reperibili unicamente in loco da documenti interni di cui l'azienda è proprietaria.

1.3 Contenuto della Tesi

Il presente lavoro di tesi ha l'obiettivo di illustrare il progetto di *data warehousing* finalizzato a supportare il controllo della qualità tramite la realizzazione di uno strumento utilizzato per monitorare la produzione di insulina all'interno di una multinazionale farmaceutica di origine americana, le cui aree aziendali interessate sono state la manifattura e il controllo qualità eseguito nell'area di *Information Technology*.

Nella realizzazione di questo lavoro, il sottoscritto è stato conivolto in prima persona all'interno di un team con compiti ben definiti ma comunque molto coeso e in stretto contatto in tutti i suoi membri per la realizzazione di tutte le fasi del progetto. In particolare il lavoro del sottoscritto è incentrato sulle fasi preliminari alla realizzazione della soluzione finale, e cioè le fasi di ETL e di *back-end* in generale. La realizzazione dei report della soluzione non è stata realizzata in prima persona dal sottoscritto ma vi è stata comunque una stretta collaborazione e attività di supporto, che ha previsto costanti modifiche al *data warehouse* al fine di una corretta e conforme rappresentazione dei report.

Il Capitolo 2 è dedicato alla presentazione del caso di studio e si pone come obiettivo principale quello di introdurre ambito, soggetti ed altri aspetti relativi al progetto realizzato. Si definiscono pertanto inizialmente l'azienda sviluppatrice, l'azienda farmaceutica in generale e quella committente nel dettaglio. In seguito sono presentati alcuni concetti chiave relativi alla tematica centrale del lavoro, cioè la produzione dei flaconi di insulina e il controllo che deve essere effettuato durante la produzione. In questo capitolo sono descritte le fasi del processo produttivo per una maggiore comprensione delle tematiche affrontate nei capitoli successivi. Viene infine introdotto il progetto oggetto della tesi.

Nel Capitolo 3, dopo una presentazione di ciò che è il *data warehousing*, viene presentata l'architettura di *data warehouse* adottata dall'azienda cliente, dal livello base dei sistemi sorgente fino al livello finale rappresentato dall'area delle *extension*.

Nel Capitolo 4 è descritta la progettazione del sistema di *Business Intelligence*, che include la fase di analisi dei requisiti, di progettazione del *data mart* e la descrizione del progetto logico, corrispondente a uno schema relazionale.

Il Capitolo 5 è dedicato all'introduzione dei principali strumenti usati per lo sviluppo del sistema. Si descrivono quindi la piattaforma sulla quale vengono modellati i dati dai sistemi sorgente (Oracle) con relativo strumento di sviluppo, la piattaforma in cui viene sviluppato il *data warehouse* (Netezza-IBM) con relativo strumento di sviluppo, lo strumento usato per le operazioni di estrazione, trasformazione e caricamento (Informatica PowerCenter) e la piattaforma dedicata alla realizzazione di report grafici (SAP Business Object). Dei vari strumenti si descrivono le funzionalità più importanti che sono state utilizzate per la realizzaione di questo progetto.

Nel Capitolo 6 si presenta il processo di ETL, volto a combinare i dati provenienti dalle varie fonti a disposizione per ottenere le informazioni finali da caricare nelle strutture definite in fase di progettazione.

Il Capitolo 7 è dedicato alla descrizione dell'interfaccia del sistema rivolta all'utente finale. Si descrive inizialmente in generale ciò che riguarda la creazione di uno strato semantico dei dati tra il *data warehouse* e il *tool* di reportistica. Successivamente, viene mostrato graficamente l'aspetto del prospetto realizzato e il significato delle sezioni di un report di esempio.

Il Capitolo 8 infine conclude la tesi, e in particolare vengono fatte varie considerazioni sul lavoro affrontato e descritto in questa tesi, indicando gli sviluppi futuri che possono essere portati avanti.

2 CASO DI STUDIO

Il presente lavoro descrive un progetto, svolto presso la società di consulenza gestionale SDG Group, che prevede la realizzazione di un *data warehouse* per il monitoraggio delle informazioni rilevate durante la produzione di cartucce di insulina.

Il sistema richiesto dovrà essere regolamentato dalle norme di buona fabbricazione (GMP, *Good Manufacturing Practices*) e ha l'obiettivo di fornire agli utenti del reparto qualità, una serie di report per effettuare analisi al fine di rilasciare il prodotto finale conforme alle normative vigenti. Lo scopo principale delle norme relative alla produzione, alla distribuzione e all'uso dei medicinali deve essere quello di assicurare la tutela della sanità pubblica.

Nel presente capitolo, dopo aver descritto brevemente l'azienda sviluppatrice e il mondo dell'industria farmaceutica, sono presentate alcune informazioni generali relative alla produzione dell'insulina inquadrata nell'ambito dell'azienda cliente. Informazioni dettagliate sull'azienda committente non possono essere descritte in questo lavoro di tesi in quanto è stato richiesto di mantenere l'anonimato su di essa.

2.1 L'Azienda Sviluppatrice: SDG Group

SDG Group è una società di consulenza gestionale leader del settore, caratterizzata da una speciale visione delle pratiche di *Business Intelligence*, *Corporate Performance Management* e *Collaborative Business Analytics* nello sviluppo di soluzioni manageriali e analitiche. SDG Group nasce nel 1991. Attualmente è presente a Milano, Roma, Verona, Firenze, Londra, Barcellona, Madrid, Lisbona, Amburgo, Monaco, Il Cairo, Algeri, Parigi, Londra, York, Lima e Bogotà.

Nel 1997 è stata delineata una nuova strategia di crescita con un chiaro focus sulle soluzioni di *Business Intelligence*. L'azienda sceglie di dedicarsi allo sviluppo di innovativi servizi di consulenza di *management intelligence* e *knowledge management*. I servizi e le soluzioni offerte da SDG si sono progressivamente arricchiti negli anni con l'aggiunta di *social intelligence practices*, *big data architecture*, *business analytics* d'avanguardia e *digital transformation roadmaps*.

Figura 1: Logo di SDG Group

In quest'ottica SDG Group si impegna a diventare leader nell'offerta di metodi e strumenti di *Business Intelligence* grazie all'unione di specifiche capacità di *information technology* e consulenza gestionale, capace di soddisfare le necessità di ogni specifico settore industriale, aprendo la strada a nuove soluzioni per una migliore analisi dei dati. [SDG Group 16]

2.2 Industria Farmaceutica

L'industria farmaceutica si distingue per un elevato grado di complessità ed articolazione, per i considerevoli rischi d'impresa a cui è soggetta e per gli elevati ricavi di cui godono soprattutto le grandi aziende farmaceutiche, le cosiddette *Big Pharma*, caratterizzate da ricavi superiori ai 3 miliardi di dollari e con almeno 500 milioni di dollari investiti nel settore di ricerca e sviluppo.

Le origini dell'industria moderna vengono datate tra la fine del diciannovesimo e l'inizio del ventesimo secolo. È possibile evidenziare tre macro aree in cui si articola l'industria farmaceutica: *Pharmaceutical*, *Biotechnology* e *Life Science Medical Device*. Il settore *Pharmaceutical* è dominato dai cosiddetti *ethical drug*, ovvero i farmaci convenzionali o gli agenti più complessi, tra cui i vaccini, per i quali è necessaria la prescrizione medica. I farmaci etici costituiscono l'industria farmaceutica in senso stretto in cui operano le *Big Pharma*. A questo settore possono tuttavia essere ricondotti anche i farmaci *over the counter* (OTC *Pharmaceutical*), che prendono il loro nome dalla possibilità di essere acquistati senza prescrizione direttamente dal consumatore finale. Il settore *Biotechnology* comprende i farmaci prodotti attraverso molecole naturali più complesse che molto spesso sono create a partire da cellule viventi, mentre quello dei *Life Science Medical Device* è costituito dalle strumentazioni e dalle attrezzature al servizio dell'industria farmaceutica e di quella biotech. [Holland 04]

2.3 L'Azienda Committente

L'azienda committente è una multinazionale farmaceutica di origine statunitense, fondata sul finire dell'800. Si tratta di una delle cosiddette *Big Pharma* e si trova stabilmente tra le prime 15 aziende farmaceutiche a livello globale per quanto riguarda i ricavi. La società ha stabilimenti produttivi in 13 paesi dislocati tra Nord America, Sud America, Europa, Africa e Asia, e conta su più di 50.000 dipendenti. Dal 2008 l'azienda sviluppatrice, SDG Group, collabora con l'azienda committente ed è stata impegnata con dei progetti che hanno coinvolto 9 dei 13 stabilimenti, tra cui lo stabilimento italiano che ha sede a Sesto Fiorentino. L'attività dello stabilimento italiano è iniziata circa 50 anni fa, e da pochi anni si è spacializzata unicamente nella produzione di insulina a partire da DNA ricombinante destinata ai paesi europei ed extraeuropei. Al giorno d'oggi oltre 10 milioni di persone nel mondo utilizzano quotidianamente insulina prodotta dall'azienda committente.

Inizialmente lo stabilimento di produzione aveva il compito di manifatturare molti prodotti diversi in varie formulazioni e forme farmaceutiche, iniettabili e orali, sia per uso umano che veterinario, con un'alta complessità in termini di tecnologia farmaceutica per il mercato italiano.

Nel 2003, per far fronte alla domanda in continua crescita di insulina per il trattamento del diabete e alle esigenze del contesto italiano, viene deciso di specializzare lo stabilimento di Sesto Fiorentino nella produzione di insulina ed analoghi da biotecnologia in cartucce e dispositivi di somministrazione pre-riempiti. Il nuovo stabilimento, progettato

sul principio dell'eccellenza, ha impianti all'avanguardia per prodotti sterili da biotecnologia e presenta un'architettura dei sistemi informatici integrata con l'automazione di processo e istruzioni di lavoro totalmente elettroniche.

L'azienda è orientata all'eccellenza per rendere la vita migliore alle persone di tutto il mondo. Per questo investe circa il 20% delle vendite nel settore di ricerca e sviluppo per medicine innovative di alta qualità che aiutino le persone a vivere meglio e più a lungo.

2.4 Produzione dell'Insulina

Il diabete è una malattia cronica caratterizzata dalla presenza di elevati livelli di glucosio nel sangue (iperglicemia) ed è dovuta a un'alterata quantità o funzione dell'insulina. L'insulina è un ormone prodotto dal pancreas, che consente al glucosio l'ingresso nelle cellule e il suo conseguente utilizzo come fonte energetica. Quando questo meccanismo è alterato, il glucosio si accumula nel circolo sanguigno.

Anticamente i preparati insulinici industriali venivano prodotti a partire dal pancreas di bovini e suini, in quanto si tratta di un organo chimicamente simile a quello umano, ma il processo di estrazione era molto complesso. La quantità di insulina presente è infatti molto scarsa e per poter produrre un flaconcino erano necessari circa sei mesi. Inoltre, le insuline di origine animale contengono impurità, che provocano intolleranze e reazioni allergiche. [Gebel 03]

Attualmente, la maggior parte dei preparati insulinici, sono preparati tramite tecnologia genica. Per la sintesi di insulina viene utilizzato il microrganismo *Escherichia Coli.* Piccoli tubi che contengono questi batteri vengono conservati in freezer a una temperatura di -70° C per decenni.

Nel 1980, l'azienda committente, ha prodotto un lotto padre, chiamato *master cell bank*, e da allora continua a coltivare un lotto di *Humulin*, insulina umana biosintetica che viene prodotta tramite la tecnica del DNA ricombinante utilizzando il batterio *Escherichia Coli*. Questo preparato, prima di entrare in azione, necessita di almeno un'ora e la sua azione si esaurisce lentamente. Ogni volta che si ha la necessità di una nuova scorta di *Humulin* viene estratto un tubo dal *master cell bank*, fatto scongelare, e i batteri vengono stimolati alla crescita. Partendo da mezzo grammo di batteri questi cominciano a replicarsi esponenzialmente, raddoppiando il loro numero ogni venti minuti circa. Quando il tubo che li contiene diventa troppo affollato i batteri vengono spostati in recipienti sempre più capienti. I microrganismi si sviluppano in un brodo batterico composto da acqua, zucchero, sale, azoto e un additivo per contrastare ogni forma di contaminazione. Dopo diversi giorni i batteri sono pronti per la produzione di insulina, e possono essere separati dal brodo in cui sono contenuti. Questo viene fatto grazie a una centrifuga che spinge i batteri verso il fondo della macchina. Il brodo viene rimosso e sostituito con un liquido contenente una sostanza in grado di rompere le membrane cellulari, per fare in modo che i batteri vengano separati dall'insulina prodotta al loro interno. Il passo successivo è la purificazione dell'insulina e questo viene fatto sfruttando la diversità in termini di carica elettrica, acidità e dimensione dell'insulina rispetto alle altre molecole.

Più recentemente la tecnica del DNA ricombinante ha permesso di sviluppare un nuovo tipo di insulina chiamata *Humalog* (conosciuta anche come LISPRO). Si tratta, anche in questo caso, di insulina umana in cui viene modificata la sequenza di due amminoacidi,

Prolina e Lisina. Questo tipo di insulina si comporta in maniera analoga all'insulina presente su un soggetto non diabetico, e quindi entra in azione immediatamente dopo l'assunzione di un pasto e ha una durata più corta nel tempo.

Lo step finale da compiere, prima che l'insulina sia pronta per il *packaging*, è la cristallizzazione. È un'operazione molto delicata, e viene ottenuta tenendo a riposo l'insulina prodotta per non meno di 24 h con temperatura compresa tra i 14 e i 16 °C e PH neutro. L'insulina viene mischiata con zinco, che aiuta a formare cristalli stabili, e disidratata fino ad assumere la forma di una polvere di cristalli. A tempo debito, i cristalli vengono re-idratati in soluzione e versati in flaconcini, penne e cartucce, pronte per la spedizione.

L'insulina prodotta viene commercializzata in tre diverse tipologie che differiscono per entrata in azione e durata, e sono nello specifico: soluzione, sospensione e miscela. Le soluzioni si presentano vivamente limpide, entrano subito in azione e hanno una breve durata. Le sospensioni contengono cristalli di insulina e si presentano torbide, la loro entrata in azione è più lenta e hanno una maggiore durata nel tempo. Infine le miscele, come si può intuire dal nome, hanno caratteristiche intermedie tra le due tipologie precedentemente descritte, si presentano torbide, agiscono rapidamente e hanno un'azione duratura nel tempo.

2.5 Logiche del Ciclo Produttivo

Il processo manifatturiero dell'azienda committente è composto da 5 fasi: *Componets*, *Formulation*, *Filling*, *Sorting* e *Packaging*. Solamente le fasi di *Formulation*, *Filling* e *Sorting* sono però coinvolte nel processo produttivo delle cartucce di insulina. Inoltre abbiamo la fase di *Campaign* che corrisponde ad un insieme di fasi di *Filling*. Quindi oltre al processo orizzontale esiste questa macro-fase verticale.

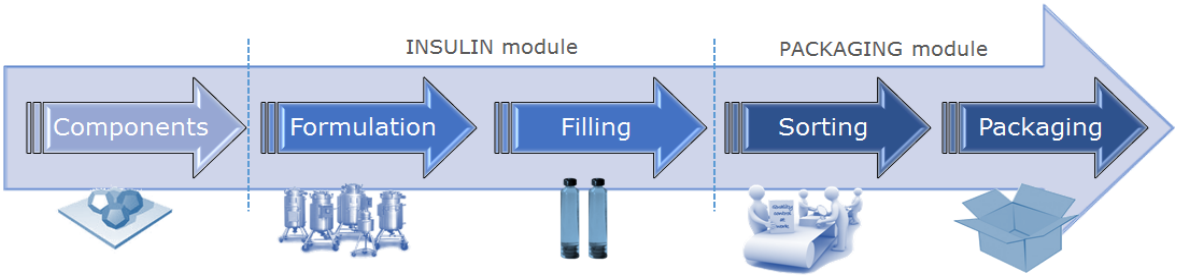


Figura 2: Fasi del processo produttivo

La prima fase è quella di *Componets*, in cui si lavora direttamente sulle materie prime, e precede la fase di produzione delle cartucce di insulina che verranno immesse sul mercato. Le fasi successive che sono coinvolte direttamente nella produzione dell'insulina e nel suo incapsulamento sono descritte di seguito più nel dettaglio. Le cartucce che superano con successo il controllo della fase di Sorting sono pronte per passare alla successiva fase, il *Packaging*, che si occupa di assemblare e confezionare il prodotto finale.

2.5.1 Formulation

La fase di *Formulation* è la prima fase in cui si inizia la produzione dell'insulina, è il processo in cui le diverse sostanze chimiche, tra cui il principio attivo dell'insulina, sono combinate per produrre la preparazione liquida. Questa fase ha una durata di 1 giorno se si tratta di insulina sotto forma di soluzione, oppure di 2/3 giorni se si tratta di insulina sotto forma di sospensione.

Inizialmente i vari componenti utili alla produzione dell'insulina vengono inseriti in un *Charge Tank* in cui vengono miscelati tra loro. Tra i componenti aggiunti abbiamo:

- Principio attivo dell'Insulina;
- Glicerina (Isotonizzate): porta l'insulina alla stessa pressione del sangue;
- Protamina (in sospensione): forma il cristallo permettendo la cristallizzazione;
- Ossido di Zinco: agente stabilizzante utile alla cristallizzazione;
- Fenolo (in sospensione) e Metacresolo (Conservanti): prevengono la contaminazione durante l'uso del prodotto;
- Fosfato di Sodio: tampone per stabilizzare il PH;
- Acido Cloridrico e Idrossido di Sodio: agenti correttivi del PH;
- WFI (*Water for Injection*, Acqua per Iniezione).

Una volta miscelati, i componenti vengono riversati, attraverso un componente chiamato Pompa Peristaltica, nel *Formulation Tank* in cui viene anche aggiunta WFI per portarli a peso. A questo punto la miscela passa attraverso dei filtri sterilizzatori, detti *Sterile Filters,* che trattengono i microorganismi e poi attraverso uno *Sterile Transfer Panel,* che funge da tramite tra i *tank* non sterili e quello sterile. Infine si arriva nello *Sterile Tank*, che corrisponde all'ultimo step della fase di *Formulation*, in cui si ultima la produzione dell'insulina.

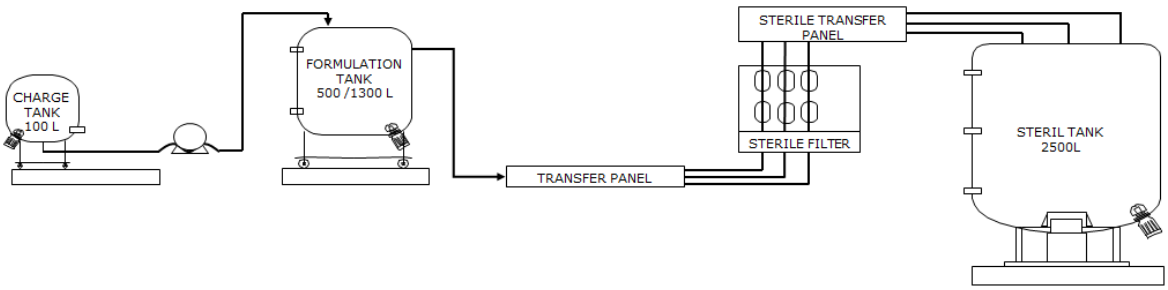


Figura 3: Macchinari coinvolti nella *Formulation*

In questa fase i macchinari coinvolti sono caratterizzati anche da cicli volti alla pulizia e sterilizzazione degli stessi, nello specifico tali cicli sono chiamati CIP (*Cleaning in Place*), SIP (*Sterilization in Place*) e RINSE (una versione più rapida dei cicli CIP). Questi cicli di lavaggio sono fatti prima con acqua e acqua ossigenata (4%) e poi con vapore a 125-135° e hanno una durata di 40 minuti.

2.5.2 Filling

La fase di *Filling* è la fase di riempimento delle cartucce con il preparato realizzato nel processo precedente. Questa fase, che ha una durata di 2 giorni, è leggermente differente tra i siti americano e italiano (*Filling* RABS) e il sito francese (*Filling* ISO). Per questo motivo la soluzione finale presenterà due report per la *Filling*, così come per la *Campaign*.

Il tipo di linea di *Filling* ISO è caratterizzata da un tipo di struttura chiamata *Isolator*. Si tratta di un'unità completamente sigillata in cui l'aria al suo interno è protetta tramite differenziale di pressione.

La linea di *Filling* RABS (*Restricted Access Barrier System*) invece è caratterizzata da un tipo di struttura aperta in cui l'aria fluisce liberamente tra il RABS e l'ambiente circostante senza la presenza di sistemi di filtrazione.

La seconda differenza tra i due tipi di linea è che *Isolator* utilizza un sistema di bio-decontaminazione automatizzato che utilizza perossido di idrogeno vaporizzato, RABS invece prevede una pulizia di tipo manuale. Gli *Isolator* sono maggiormente acclamati dall'ambiente scientifico di riferimento. I vantaggi di *Isolator* sono rappresentati da:

- alta protezione del prodotto;
- si possono controllare il grado di temperatura e umidità ;
- gli operatori sono protetti totalmente, soprattutto in caso di prodotti tossici;
- cicli di pulizia più accurati e ridotti nel tempo.

L'aspetto negativo è dato dagli elevati costi d'acquisto. Le cartucce in questa fase assumono un ruolo da protagonista, esse sono formate da quattro elementi:

- ***Disc Seal***, la guarnizione di chiusura per garantire l'integrità e la protezione dell'insulina all'interno della cartuccia;
- **Cartuccia**, contenitore in vetro pronto per essere riempito e inserito in dispositivi di iniezione;
- ***Glass Bead***, perline di vetro pure per garantire la miscelazione, in caso si tratti di insulina in sospensione;
- ***Plunger***, pistone per erogare l'insulina all'interno della cartuccia.

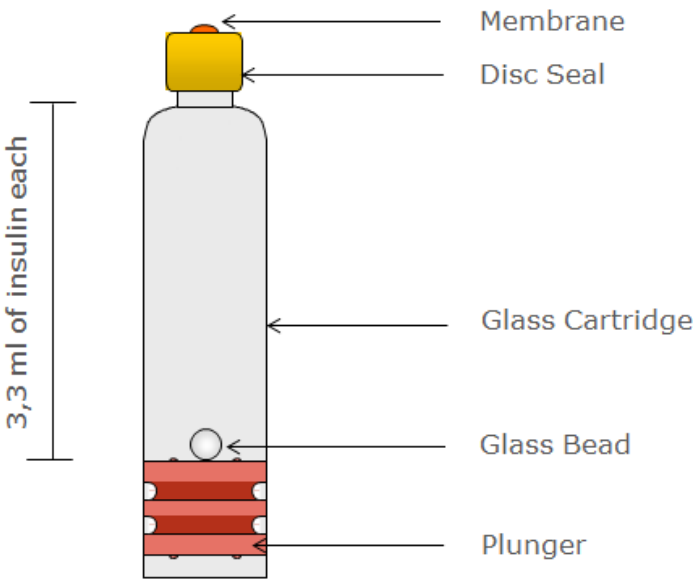


Figura 4: Cartuccia di insulina

Le cartucce prodotte, a loro volta, possono essere utilizzate in specifici strumenti dosatori (*devices*) divisi in:

- penne monouso;
- penne multiuso (quindi ricaricabili).

Il processo inizia con la preparazione dei componenti. Inizialmente le cartucce vengono caricate su uno scivolo trasparente (*light table*), un tavolo illuminato dal basso, al fine di evidenziare grossolani difetti del vetro. Le cartucce vengono poi accompagnate da una coclea ad una vasca in cui subiscono un bagno ad ultrasuoni. La vasca è alimentata da un sistema di acqua riciclata calda posta al piano sottostante. Poi un ascensore le aggancia e le riporta in alto per l'asciugatura che avviene tramite aria compressa. In questa fase iniziale avviene la siliconatura del *plunger*, per permettere lo scorrimento di esso all'interno di tutta la cartuccia.

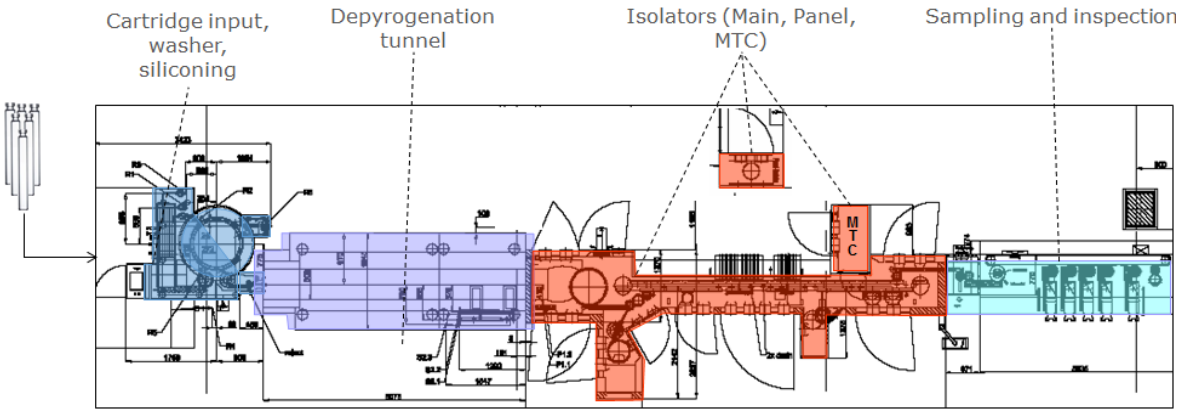


Figura 5: Macchinari utlizzati nella *Filling*

La siliconatura avviene tramite aghi che salgono dal basso verso l'alto, sparano la soluzione (acqua e silicone 1%) e poi tornano in basso verso la loro posizione iniziale. Successivamente i componenti passano attraverso un tunnel di depirogenazione, dove le temperature del calore secco possono superare i 250°. La depirogenazione oltre a distruggere completamente tutti i microrganismi, permette la polimerizzazione del silicone creando una patina invisibile. Successivamente le cartucce vengono raffreddate passando attraverso due zone in cui vengono inondate da aria fresca filtrata. A questo punto i vari componenti della cartuccia arrivano nell'*Isolator* (se si tratta del sito francese) o nel RABS (per gli altri due siti) e tramite guide di trasporto la cartuccia viene composta con il *plunger,* in seguito tramite degli aghi comandati da un sistema di vuoto vengono inserite le *glass bead*. A questo punto la cartuccia è pronta per essere riempita. L'Insulina prodotta durante la fase di *Formulation* arriva dallo *Sterile Tank* e tramite un *Panel Isolator*, fondamentale nel mantenimento della sterilità, viene confluita nell'*Intermediate Filling Tank*.

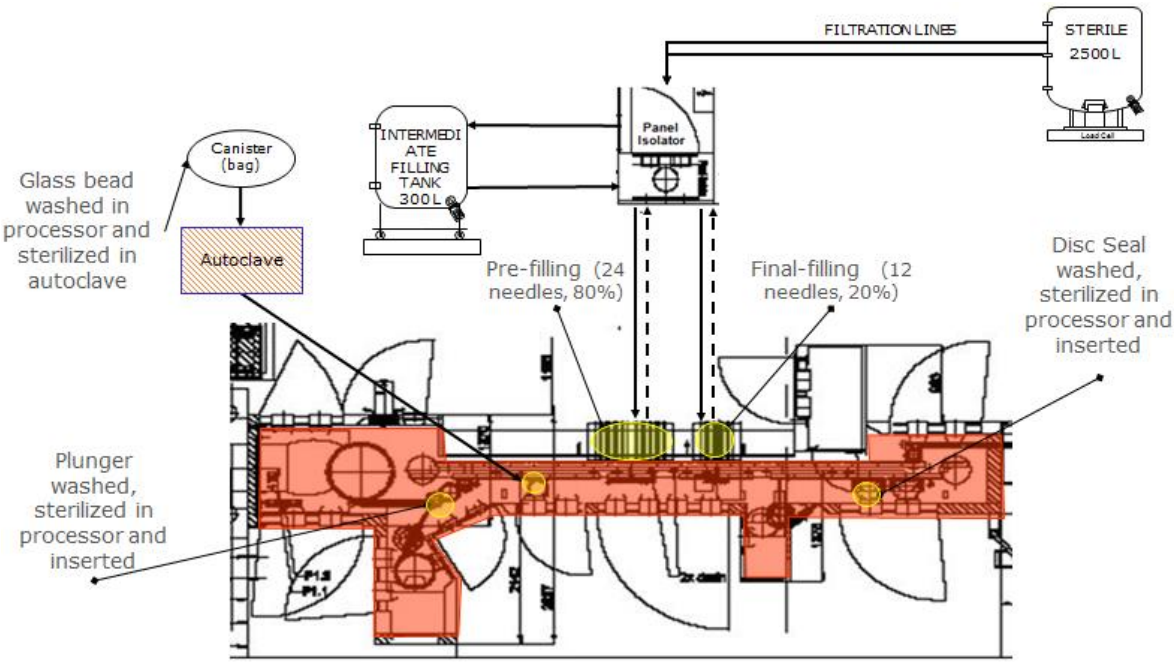


Figura 6: Dettaglio dell'*Isolator*

A questo punto in dosi minori la miscela viene passata a degli aghi che riempiono in un primo momento l'80% della cartuccia e successivamente il restante 20%. Dopo il riempimento, avviene la tappatura della cartuccia con il *disc seal*. Anche questa fase, come la precedente, è caratterizzata dalla presenza di cicli di pulizia e sterilizzazione dei macchinari coinvolti, in maniera analoga a quanto descritto precedentemente.

2.5.3 Sorting

Questa fase si occupa dell'ispezione visiva (conosciuta anche come *Visual Inspection*) delle singole cartucce di insulina con campionamento e del controllo qualità, per identificare eventuali anomalie e criticità nelle cartucce di insulina emerse nelle due fasi antecedenti. Anche questa fase, come la precedente, ha una durata di 2 giorni.

Il controllo avviene tramite un processo automatizzato, utilizzando dei macchinari dedicati, e per ultimo da un controllo manuale, svolto da un personale addetto. E' possibile individuare tre fasi in questo processo:

- AVIM, *Automatic Visual Inspection Machine*;
- SVIM, *Semi-automatic Visual Inspection Machine*;
- AQL, *Acceptable Quality Limit*.

Il primo controllo qualità che viene fatto al termine della fase di *Filling* è effettuato tramite l'AVIM della Seidenader, leader mondiale per la fabbricazione di macchinari per l'ispezione. Con sensori e telecamere ad alta risoluzione l'AVIM controlla che le singole cartucce di insulina siano prive di difettosità.

Figura 7: Macchinario AVIM

Questa macchina che riesce a controllare fino a 600 cartucce al minuto, tramite 7 telecamere riesce a rilevare anomalie come:

- Difetti di chiusura;
- Crepe e graffi nella cartuccia;
- Qualità della guarnizione di chiusura;
- Rottura alla base o al corpo della cartuccia;
- Perdite.

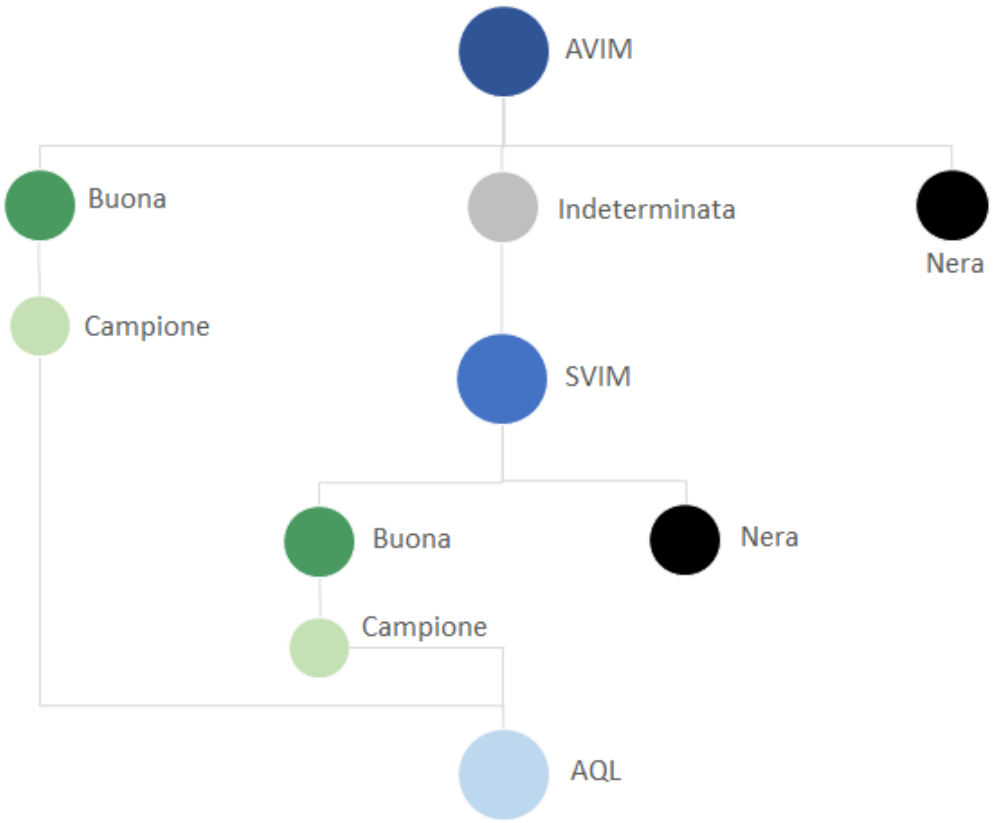


Figura 8: Schema riassuntivo della *Sorting*

Dopo questa prima verifica l'AVIM può restituire tre stati:

1. Buona, ovvero la cartuccia è priva di difettosità e può essere immessa sul mercato;
2. Indeterminata/Grigia, ovvero la macchina non è in grado di dire se la cartuccia sia da scartare o meno, e bisogna ricontrollarla;
3. Nera, la cartuccia deve essere scartata perché difettosa.

Se l'AVIM stabilisce che un lotto è privo di difettosità (tutte le cartucce superano la verifica con esito Buona) viene scelto un campione di cartucce dall'intero lotto, e sottoposto alla verifica di AQL.

Se invece l'AVIM non è in grado di decidere se la cartuccia sia da scartare o meno, si passa a un sistema di ispezione semiautomatico, SVIM, dove il controllo qualità viene effettuato oltre che dalla macchina anche dal personale addetto. Nello specifico la macchina posiziona le cartucce e le fa ruotare in modo tale da favorire al meglio il controllo visivo da parte degli operatori.

Dal controllo fatto con la SVIM la cartuccia può trovarsi in uno stato di Buona o Nera. Nel primo caso, viene scelto un campione, così come viene fatto quando le cartucce superano l'ispezione visiva fatta dall'AVIM, e sottoposte all'ispezione manuale AQL. Nel caso in cui le cartucce non superino la verifica della SVIM vengono scartate definitivamente.

L'AQL è un controllo statistico sulle cartucce classificate come buone da AVIM e SVIM. Viene prelevato un campione di cartucce buone per valutare la precisione delle macchine di ispezione visiva per ogni lotto riempito. È tollerato un certo margine di errore dei controlli AVIM e SVIM nel ritrovamento di difetti sulle cartucce. Se il controllo supera i limiti per cui AVIM e SVIM sono considerate affidabili, si fa un *re-sorting* del lotto (solo delle cartucce dichiarate buone); su questo lotto di *re-sorting* si riesegue la verifica AVIM solo sul difetto per cui l'AQL è uscito dai limiti (lanciando una ricetta ad hoc), e l'AQL di nuovo su tutte le cartucce buone, mentre la fase di SVIM non viene più eseguita.

2.6 Global Batch Review 3.0

Il progetto *Global Batch Review* ha permesso di fornire all'azienda presa in esame strumenti operazionali integrati con le soluzioni di *Business Intelligence* per la supervisione dei processi governati da PMX, GRIP e PA (sistemi informatizzati che hanno la principale funzione di gestire e controllare la funzione produttiva dell'azienda); il centro di interesse per l'analisi è il lotto (*Batch*) di produzione e la sua *compliance* agli standard qualitativi attesi sul prodotto finito (*Review*). Essendo non strettamente un progetto analitico ma operazionale, il suo ruolo è critico per il business in quanto permette di raccogliere dati dalla rete di produzione allo scopo di validare l'intero processo produttivo in termini di qualità raggiunta (validazione *Quality*) molto più rapidamente di quanto lo si possa fare con un sistema di supervisione manuale. Il sistema produttivo è orientato al processo e non al prodotto, in quanto la linea di produzione è in grado di produrre formulazioni diverse dello stesso prodotto a parità di *equipment* produttivo. Il progetto in questione coinvolge tre siti produttivi, uno negli Stati Uniti, uno in Italia e uno in Francia.

Ai fini degli standard GMP, essendo il processo di validazione *Quality* del *batch* fondamentale per la produzione e la sua commercializzazione, il progetto *Global Batch Review* è da ritenersi un sistema di reporting operazionale critico per il business. La base da cui è partito questo progetto è la *Global Batch Review 2.0* che aveva coinvolto i soli siti italiano e americano. Questo progetto aveva previsto la produzione di 4 report, uno per fase produttiva. L'esigenza di globalizzare sempre di più la soluzione proposta ha portato allo sviluppo di tale progetto con una nuova *release* che ha coinvolto un ulteriore sito produttivo, quello francese appunto. La nuova *release* non si è limitata a questo ma si è spinta in maniera decisa anche verso un miglioramento di quanto prodotto con le *release* precedenti, e quindi a delle migliorie sulla soluzione anche per quanto riguarda i siti coinvolti nella *release* precedente.

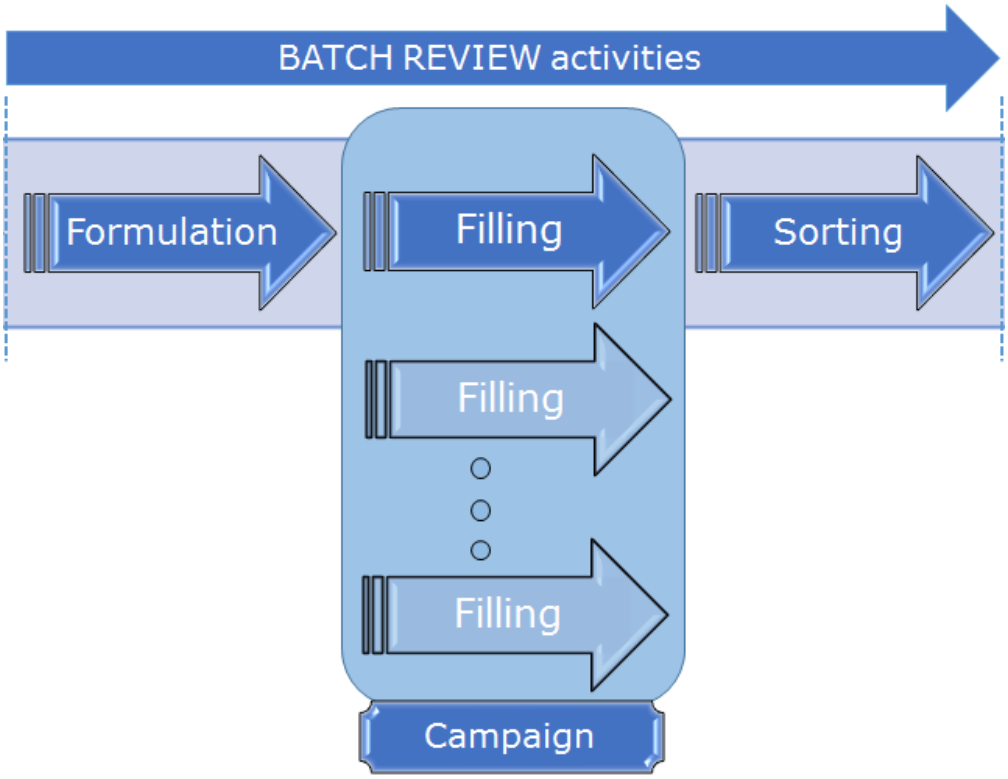


Figura 9: Fasi coinvolte nella *Batch Review 3.0*

La *Global Batch Review 3.0* si compone di 6 macro-report, uno per le fasi di *Formulation* e *Sorting* e due per le fasi di *Filling* e *Campaign*. Il report di *Formulation* è comune a tutti e tre i siti; il report di *Sorting* si riferisce solo al sito americano e a quello italiano, invece quelli di *Filling* e *Campaign* differiscono per tipo di linea produttiva, differenti tra loro per struttura; il sito francese infatti ha un tipo di linea produttiva chiamata ISO (*Isolator*), il sito americano e quello italiano invece hanno un tipo di linea produttiva chiamata RABS (*Restricted Access Barrier System*).

Una volta concluso tale progetto è previsto un ulteriore *upgrade* con la *release* 4.0 che coinvolgerà anche il sito cinese entro il prossimo anno. Tale progetto ha portato all'azienda committente vari benefici sotto più fronti. Innanzitutto ciò che ne ha risentito maggiormente è l'efficienza, il controllo sulla qualità del prodotto è effettuato in sole 2 ore a fronte di 2 giorni. Le persone coinvolte in tale controllo scendono da 3 a 1, e questo si

riflette, soprattutto a lungo andare, sui costi. Ma soprattutto i risultati più soddisfacenti si riflettono sulla qualità del controllo stesso che riducono il margine di errore e l'implementazione di complesse logiche di business.

3 ARCHITETTURA DI DATA WAREHOUSE

Questo capitolo dà una breve introduzione sulle scelte, caratteristiche e aspetti architetturali del processo di *data warehousing*. Viene inoltre presentata l'architettura di *data warehouse* adottata dall'azienda cliente con la descrizione fatta a partire dai sistemi sorgente, l'area di *staging*, il *core* e infine la parte delle *extension*.

3.1 Il Processo di Data Warehousing

Con il termine *data warehouse* si intende una particolare base di dati con le seguenti principali caratteristiche [Albano 12]:

- **Tempificata**. Contiene informazioni sul tempo in cui si verificano gli eventi di interesse. Questa caratteristica permette di realizzare la storicizzazione delle informazioni;

- **Integrata**. I dati non provengono in genere da un'unica sorgente (una base di dati dell'organizzazione), ma sono il risultato di un lungo e costoso processo di integrazione di dati eterogenei;

- **Statica**. I dati vengono usati interattivamente per operazioni di ricerca e non di modifica. Periodicamente ai dati disponibili se ne aggiungono di nuovi o si rimuovono quelli ritenuti obsoleti;

- **Organizzata per soggetti**. Nei sistemi operazionali i dati sono organizzati per eseguire le attività aziendali quotidiane, mentre nei sistemi direzionali i dati sono organizzati per analizzare dei soggetti di interesse che influenzano l'andamento complessivo dell'azienda. Quando i dati riguardano un solo soggetto di interesse si parla di *data mart* e possono essere un sottoinsieme di un *data warehouse* più generale;

- **Finalizzata ad analisi di supporto alle decisioni**. I dati sono organizzati per facilitare le loro analisi al fine di produrre delle opportune sintesi di supporto ai processi decisionali.

Con il termine *data warehousing* si indica il processo attraverso cui i dati sono organizzati in un *data warehouse* e quindi messi a disposizione degli utenti finali utilizzando applicazioni di *Business Intelligence*. Un'applicazione di questo tipo deve permettere la ricerca di aspetti interessanti deducibili dai dati attraverso strumenti di analisi.

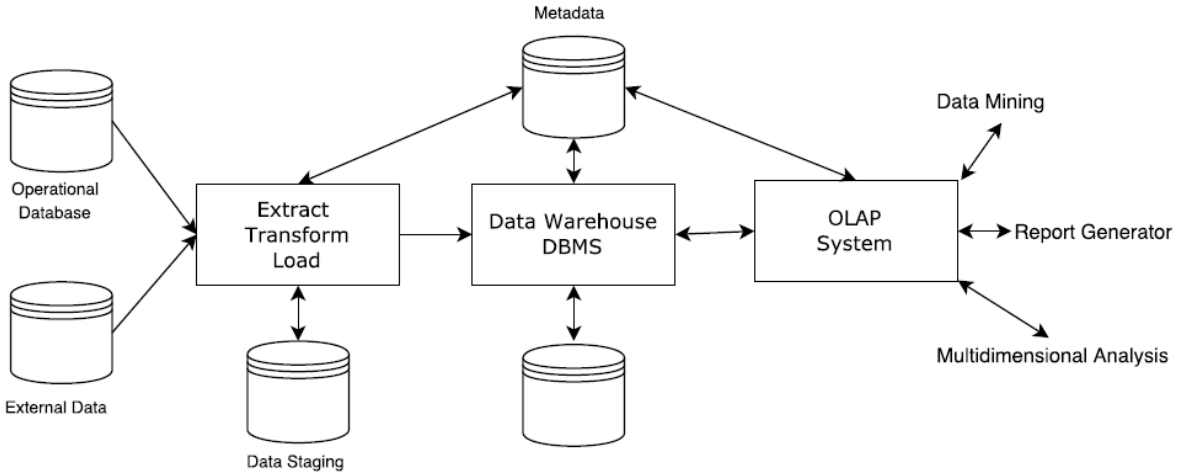


Figura 10: Architettura a 3 livelli

Nel caso qui analizzato, i dati sono organizzati secondo quella che in [Albano 12] è definita come architettura a tre livelli. Prevede infatti il livello delle sorgenti di dati, il livello del *data warehouse* e il livello dei dati intermedi, detto ODS (*Operational Data Store*), che contiene i dati ottenui da integrazione delle diverse sorgenti di dati. Il processo di *data warehousing* è composto da quattro fasi principali:

1) **Sorgenti**. Definizione delle sorgenti dei dati, base di dati operazionale ed eventualmente fonti esterne.

2) ***Extraction, Trasformation, Loading***. Fase in cui i dati provenienti dalle sorgenti vengono opportunamente estratti, modificati e caricati nel sistema per *data warehouse*.

3) ***Data Warehouse***. Sistema in cui sono opportunamente memorizzati i dati con lo scopo di ottimizzare la loro successiva analisi.

4) **Strumenti di Analisi**. Insieme di strumenti che interrogando il *data warehouse* permettono di creare reportistica ad hoc, analizzare i dati in modo multidimensionale ed effettuare operazioni di *data mining*.

Come è noto, questa soluzione garantisce diversi vantaggi:

- separazione tra il DBMS (*database management system*) operazionale e quello finale, che potrà essere un sistema progettato appositamente per il supporto alle decisioni;

- separazione tra le applicazioni transazionali e quelle di *Business Intelligence*, eliminando influenze reciproche sulle prestazioni;

- separazione tra il processo di estrazione e integrazione delle fonti di dati e il processo di riorganizzazione e caricamento dei dati stessi nel *data warehouse* (vero vantaggio aggiuntivo che si ottiene passando dall'architettura a due livelli a quella a tre).

Una volta creato e popolato, il *data warehouse*, necessiterà di fasi di aggiornamento effettuate ad intervalli di tempo regolari per garantire l'allineamento con la base di dati operazionale.

3.2 Architettura Global Manufacturing Data Mart

L'architettura multilivello adottata dalla società cliente prende il nome di *Global Manufacturing Data Mart* (GMDM). Questa architettura , progettata e implementata da SDG Group, oltre a quelle tipiche di un *data warehouse* presenta anche le seguenti proprietà:

- **Globale**. Deve permettere l'integrazione di sorgenti differenti provenienti dai diversi siti produttivi;
- **Estendibile**. Deve essere possibile accogliere nuove applicazioni e sistemi sorgente senza riprogettare integralmente il sistema;
- **Scalabile**. L'architettura *hardware* e *software* deve poter essere facilmente ridimensionata a fronte della crescita nel tempo dei volumi di dati da gestire ed elaborare e del numero di utenti da soddisfare.

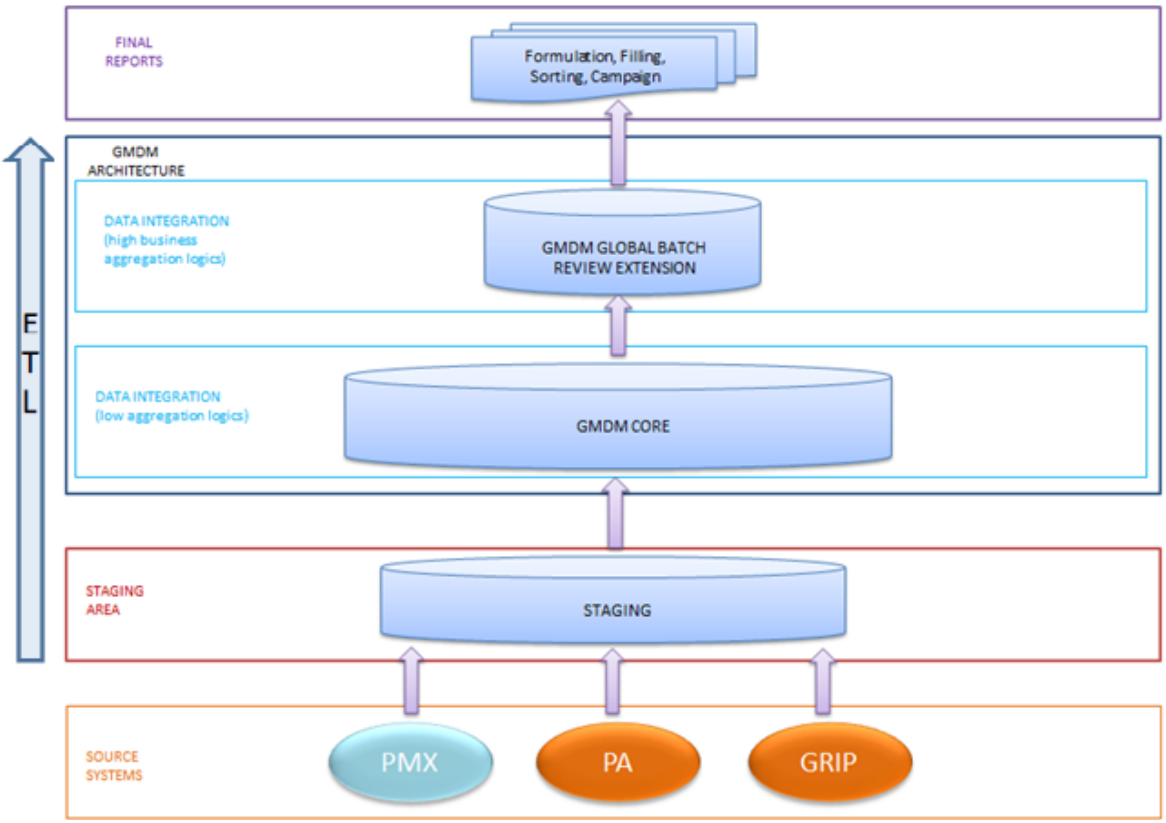


Figura 11: Architettura GMDM

Questa architettura è formata da quattro livelli:

1) il livello delle sorgenti di dati, locali nei vari siti produttivi;

2) il livello di *staging*, che contiene i dati ottenuti dal processo di integrazione delle varie sorgenti di dati;

3) il livello del *data warehouse* (chiamato *core*), a livello globale contenente dati provenienti da tutti i siti produttivi locali;

4) il livello delle *extension*, che contiene i *data mart*, anche questo a livello globale.

Questa architettura, integra i dati provenienti da quattro zone (dette anche regioni) differenti:

- zona 1: Stati Uniti, Canada, Mexico e Sud America;
- zona 2: Europa, Medio Oriente e Africa;
- zona 3: Asia Pacifica;
- zona 4: Porto Rico.

3.3 Source Area

Nello strato inferiore del GMDM, troviamo i sistemi sorgente costituiti da basi di dati operazionali. Le sorgenti sono specifiche per sito produttivo: abbiamo i sistemi sorgente PMX, GRIP e PA del sito italiano, che fa parte della Zona 2, e i sistemi sorgente PMX e GRIP dei siti americano, appartenente alla Zona 1, e francese, anche questo in Zona 2. Successivamente, verranno integrate anche le sorgenti di altri sistemi regionali. Tuttavia non sarà necessario modificare l'architettura GMDM dato che questa è stata progettata in modo parametrico, e non presenta soluzioni ad hoc per una specifica zona o sito produttivo.

3.3.1 Descrizione del Sistema Sorgente PMX

Il sistema informativo PMX (*Production Management System*) viene classificato a livello funzionale come MES (*Manufacturing Execution System*), ovvero un sistema informatizzato che ha la principale funzione di gestire e controllare la funzione produttiva di un'azienda. La gestione coinvolge il dispaccio degli ordini, gli avanzamenti in quantità e tempo, il versamento a magazzino, nonché il collegamento diretto ai macchinari per dedurre informazioni utili ad integrare l'esecuzione della produzione, come a produrre

informazioni per il controllo della produzione stessa. PMX è un prodotto rilasciato dalla società Rockwell Automation, e la sua implementazione rappresenta un sistema informativo integrato all'interno delle soluzioni IT dell'azienda farmaceutica presa in esame in questa tesi, per l'ingegnerizzazione e l'esecuzione del processo di produzione del farmaco. Il sistema PMX è stato implementato nella versione PMX PPN: versione del prodotto in produzione a partire da Gennaio 2014, in alimentazione dati.

Il database operazionale PMX contiene un numero molto alto di tabelle, viste ed altri oggetti (*package*, funzioni e procedure).

3.3.2 Descrizione dei Sistemi Sorgente GRIP e PA

Il sistema informativo GRIP (*Global Resource Integration Platform*) e quello PA (*Process Automation*) indicano un gruppo di tecnologie che forniscono informazioni sulle *Recipe*, *Critical Alarms* e dati sui *Cicli*.

Nello specifico PA è utilizzato unicamente nel sito italiano, in tutti gli altri siti le informazioni contenute in PA sono incluse in GRIP. GRIP e PA prevedono una piattaforma database Oracle con due schemi (PA solo uno) contenenti i dati di magazzino di automazione dei processi e i dati di configurazione di GRIP.

Il modulo degli *Alarms* utilizza i dati di allarme presenti nel database di *SQL Server Alarms*. I seguenti attributi sono memorizzati quando un allarme è inserito:

- In/Out Alarm Date and Timestamp (a seconda se si tratta di quando è partito o cessato un allarme);
- Tag Name: un identificativo dell'allarme;
- Severity: il grado di severità associato;
- Alarm Description: una descrizione dell'allarme;
- Unit: l'unità su cui è scattato l'allarme;
- Batch: il lotto su cui è scattato l'allarme;
- Cycle: il ciclo associato all'allarme;
- EquipmentGUID;
- FlexField.

Per quanto riguarda i *Critical Alarms* inoltre, quando questi vengono ravvisati, sono memorizzati anche gli attributi:

- Alarm Ack Date and Timestamp;
- Username;
- Full Name;
- Meaning;
- Client.

Le *Recipe* invece sono memorizzate all'infuori di GRIP, nel *Process Automation Warehouse* (PAW). Le informazioni sono strutturate su due livelli, le *Recipe Headers* e le *Recipe Details*.

I dati raccolti dal *Data Collection Control* sono i dati che riguardano i cicli sui *batch*. Questi dati sono memorizzati nel *data warehouse* di *Process Automation* e anche queste informazioni sono strutturate su due livelli, i *Cycle Header* e i *Cycle Detail*. Le prime identificano unicamente le informazioni riguardanti i cicli dei processi, le altre invece contengono valori chiave per i report che riguardano anche in questo caso i cicli dei processi.

3.4 Staging Area

Data la consistente quantità di dati e tabelle coinvolte nel GMDM, è presente una struttura intermedia tra le sorgenti ed il *core* detta *staging* area.

L'area di *staging* è un'area tecnica per rendere più semplice l'alimentazione del *core* vero e proprio. La ragione per la quale esiste un'area di *staging* è che in questo modo si riesce a identificare il sottoinsieme di informazioni utili del *source system*, isolandole prima del caricamento del *data warehouse*, ed evitando di appesantire troppo il sistema delle sorgenti con le elaborazioni complesse del DWH.

In questo livello vengono memorizzate le tabelle provenienti dalla fonte senza implementare alcuna trasformazione. Quindi il livello di dettaglio sarà speculare alle informazioni provenienti dai sistemi sorgente alimentanti. Lo *staging*:

- inserisce i nuovi record registrati sui sistemi sorgente;
- aggiorna i record che hanno subito una variazione ad almeno uno dei campi;
- cancella logicamente eventuali record che sono stati eliminati fisicamente sui sistemi sorgente;
- storicizza l'andamento dei dati nel tempo.

Lo *staging* è caricato tramite una *stored procedure* parametrica scritta in PL/SQL invocata tramite un flusso unico per sorgente e sito produttivo. L'area di *staging* è composta da:

- tabelle di configurazione;
- tabelle dello *staging*.

Le tabelle di configurazione sono fondamentali per il caricamento dei dati dai sistemi sorgente e si basano su una logica *delta* (spiegata successivamente più nel dettaglio). I dati infatti vengono caricati basandosi sulla comparazione del *timestamp* dell'ultimo caricamento così da caricare i nuovi dati o aggiornare quelli già presenti che hanno subito delle modifiche. Tutti i record che sono stati eliminati sui sistemi sorgente verranno cancellati logicamente dalle tabelle dello *staging* modificando il valore del campo RCRD_STS_CD.

Per garantire la globalità del sistema, le tabelle di *staging* contengono i seguenti principali attributi tecnici:

- FCLTY_CD: è il codice univoco che identifica il sito produttivo del record;
- SRC_INST_CD: è il codice univoco che identifica l'installazione della sorgente nel sito produttivo;
- RCRD_STS_CD: identifica lo stato di un record, "A" se il record è attivo o "D" se il record è stato cancellato;
- INS_TMSTMP: identifica il *timestamp* di inserimento del record;
- UPD_TMSTMP: identifica il *timestamp* di aggiornamento del record;
- END_TMSTMP: identifica il *timestamp* di cancellazione del record.

3.5 Core Area

A partire dall'area di *staging* è possibile alimentare il *core*. Questa è la prima area globale, dove è possibile ospitare le trasformazioni, seguendo le regole che definiscono gli oggetti di business necessari alle analisi.

In questa area oltre a mantenere il massimo livello di dettaglio disponibile (i raggruppamenti non sono concessi), si garantisce la normalizzazione su ogni tabella.
Ogni tabella di questa area verrà aggiornata in modalità *delta* riferendosi al *timestamp*, mentre una volta a settimana in modalità *full*.

Non è necessario conservare la storia, quindi i record non aggiornati verranno rimossi tramite una *stored procedure* scritta in PL/SQL, invocata dopo l'aggiornamento di ogni tabella del *core*, oppure tramite un *mapping* di *delete*.

Nel *core* è possibile definire:

- tabelle di configurazione: usate nelle logiche di caricamento delle tabelle del *core*;
- tabelle degli scarti: una per ogni tabella del *core*, che contengono record volutamente filtrati dai flussi di caricamento non rilevanti per le analisi o i record cancellati dalla procedura o dai *mapping* di *delete*;
- tabelle dei fatti;
- tabelle delle dimensioni.

Se possibile, ogni tabella nel *core* manterrà la propria chiave di business altrimenti vengono definite delle chiavi surrogate. Pure in questa area, vengono definiti su ogni tabella degli attributi tecnici:

- FACILITY_CD: è il codice univoco che identifica il sito produttivo del record;
- RCRD_STS_CD: identifica lo stato di un record, "A" se il record è attivo o "D" se il record è stato cancellato;
- SRC_CODE: identifica la sorgente del record;
- MDM_LOAD_ID: è il codice univoco del caricamento;
- MDM_TRNSCTN_TMSTMP: identifica il *timestamp* dell'ultimo aggiornamento del record.

3.6 Extension Area

L'area dell'*extension* è il luogo dove i dati vengono esposti in maniera chiara e compatibile con le regole e gli oggetti di business, tale strato, globale, è accessibile agli utenti finali attraverso gli strumenti di *querying* e di *reporting*, ed è costituito da un raggruppamento di porzioni logiche dei dati disponibili detti *data mart*.

I *data mart*, sono quindi tabelle denormalizzate usate per memorizzare dati aggregati al fine di fornire informazioni precalcolate e di migliorare le performance delle *query* usate nei *tool* di *reporting*.

Ogni *data mart* è aggiornato in modalità *full* e, anche qui, i record non aggiornati vengono eliminati tramite una procedura di *delete*.

Le tabelle di questo livello conterranno gli stessi attributi tecnici di quelle del *core*. Peculiarità di questa architettura è che ogni *data mart*, popolato dal *core*, potrà contenere informazioni provenienti dalla stessa (o diversa) sorgente, appartenenti a regioni distinte, ma finalizzati a soddisfare lo stesso bisogno di analisi.

Nell'area dell'*extension* oltre ai *data mart* è possibile definire:

- tabelle di configurazione: usate nelle logiche di caricamento delle tabelle dell'*extension*;
- tabelle delle dimensioni.

3.7 Nomenclatura

Al fine di garantire un'elevata manutenibilità, il nome di ogni oggetto all'interno del GMDM segue una rigida convenzione che permette di identificarne il tipo e la destinazione d'uso all'interno dell'architettura. Le principali convenzioni utilizzate sono sintetizzate di seguito:

- *staging* area:
  - TBST_L2_<source_system>_<nome_tabella>: per tabelle popolate dai sistemi sorgente;
  - TBSTCNF_<nome_tabella>: per tabelle di configurazione.
- *core* ed *extension* area:
  - MDM_<nome_tabella>_DM: per tabelle *data mart*;
  - MDM_<nome_tabella>_DIM: per tabelle dimensionali;
  - MDM_<nome_tabella>_CNF: per tabelle di configurazione;
  - MDM_<nome_tabella>_E: per tabelle degli scarti;
  - MDM_<nome_tabella>_TMP: per tabelle temporanee;
  - MDM_<nome_tabella>: per tabelle dei fatti.

Anche gli attributi devono seguire una nomenclatura ben definita. Le regole di seguito, valgono per l'area di *staging*, *core* e *extension*:

- <nome_campo>_CD: varchar, campo utilizzato come chiave di business;
- <nome_campo>_ID: integer, intero utilizzato per le chiavi surrogate;
- <nome_campo>_NBR: integer, campo utilizzato per un numerico intero;
- <nome_campo>_DT: date, campo utilizzato per una data;
- <nome_campo>_FLG: integer/varchar, campo utilizzato come *flag*;
- <nome_campo>_VAL: decimal, campo utilizzato per un numerico con virgola;
- <nome_campo>_DESC: varchar, campo utilizzato per un valore di tipo stringa.

4 PROGETTAZIONE DEL DATA MART

Nel presente capitolo si presentano le fasi del processo di progettazione del *data mart*. Più specificamente nel capitolo sono descritti, nell'ordine, il modello seguito per lo sviluppo e le fasi di analisi dei requisiti, progettazione concettuale per poi terminare con la progettazione logica del *data mart*.

Si sottolinea che le fasi qui descritte si riferiscono esplicitamente a quelle presenti nel modello proposto in [Albano 12], modello seguito per lo sviluppo della soluzione e per la stesura di questo documento.

Si individuano in tale modello cinque fasi, che sono state presentate in maniera più compatta nelle seguenti fasi:

- analisi dei requisiti, divisa in raccolta dei requisiti e specifica dei requisiti;
- progettazione concettuale del *data mart*;
- progettazione logica del *data mart*.

4.1 Analisi dei Requisiti

4.1.1 Raccolta dei Requisiti

La raccolta dei requisiti è stata portata avanti con il cliente attraverso numerosi incontri a cui hanno preso parte:

- dipendenti con profili funzionali e/o tecnici di SDG Group;
- dipendenti della direzione centrale sistemi informativi della società cliente;
- futuri utenti del sistema, costituiti da dipendenti della società cliente.

Nel corso degli incontri con il committente sono stati individuati alcuni requisiti che la soluzione è tenuta a soddisfare. Al fine di comprendere meglio questa sezione, è utile chiarire il concetto di *batch*. Un *batch* è un lotto di insulina accomunato dallo stesso codice materiale di semifinito, prodotto impiegando una *recipe* ed una BOM (*bill of material*) specifica.

Il punto focale delle analisi che verranno descritte più dettagliatamente è rappresentato dai cicli che vengono effettuati sulle varie componenti che fanno parte della linea produttiva che coinvolge le fasi di *Formulation* e *Filling*.

Un ciclo viene svolto su un *equipment* (un macchinario) che ha un nome specifico e una descrizione e per ognuno di questi cicli si vuole tenere traccia delle informazioni che lo caratterizzano quali il codice dello stesso, la tipologia di ciclo (ove presente) e il risultato che ha generato.

Ogni ciclo è associato a un EQO (*Equipment Order*) che è caratterizzato da uno status e da un codice associato. Un ciclo EQO identifica una fase di pulizia svolta sui macchinari e sugli utensili e può interessare più lotti prodotti in processi produttivi differenti. Quindi, un ciclo di pulizia può essere eseguito ogni qual volta viene prodotto un nuovo lotto di insulina, ma riguarda più lotti prodotti. Un EQO a sua volta è associato a un *batch*, cioè un lotto, da cui si possono ricavare informazioni quali il codice di lotto usato per identificarlo all'interno del processo produttivo, oppure lo stato del batch e dell'EQO. A partire dal *batch* è inoltre possibile ricavarci il materiale da cui è composto, quindi il codice e la descrizione di ognuno.

Un ciclo può avere delle informazioni accessorie riguardo la valutazione sull'esito dello stesso effettuata da un utente in una specifica data e un commento allo stesso. Tramite l'EQO è inoltre possibile ricavarci altre informazioni inerenti al ciclo, quali ad esempio la tipologia, il risultato e l'*assestment*. Un ciclo può essere completato con allarme, in quel caso avremo informazioni riguardo quando è scattato l'allarme e quando è stato disattivato, l'utente che se ne è occupato e un commento.

4.1.2 Specifica dei Requisiti

Di seguito sono riportati i requisiti di analisi più importanti emersi.

Requisito di Analisi	**Dimensioni**	**Misure**	**Metriche**
Totale dei cicli con risultato ALARMS per Equipment scattati in un determinato intervallo di tempo	*Alarm (Date In, Date Out, Tag, Comment, Signature), Equipment (Name, Description), Result*	Numero cicli con allarme	Somma (Numero cicli con allarme)
Totale dei cicli effettuati con risultato ALARMS con valutazione positiva in un determinato intervallo temporale	*Evaluation (Type, User, Comment, Date)*	Numero cicli con valutazione	Somma (Numero cicli con valutazione)
Percentuale dei cicli effettuati con risultato COMPLETED per batch associato all'EQO	*EQO (Number, Status, Batch), Result*	Numero dei cicli con risultato COMPLETED	Percentuale (Numero dei cicli con risultato COMPLETED)
Percentuale dei cicli con risultato ALARM per tipologia	*Result, Type*	Numero dei cicli con allarme	Percentuale (Numero ciclo con allarme)

Tabella 1: Requisiti e Dimensioni di Analisi

Come affermato in [Albano 12] la granularità del singolo fatto determina il tipo di analisi che possono essere effettuate sui dati; in generale è preferibile scegliere una

granularità fine in modo da rendere possibili eventuali analisi dettagliate. In questo caso gli utenti sono interessati ai cicli effettuati sui singoli *Equipment*.

Descrizione	Dimensioni Preliminari	Misure Preliminari
Il fatto descrive l'esecuzione di un ciclo di pulizia su un determinato Equipment inerente la produzione di un lotto di insulina	*Alarm, EQO, Equipment, Evaluation, Result, Type*	Numero cicli

Tabella 2: Fatto

Il numero dei cicli è un valore calcolato di volta in volta a seconda del requisito di analisi. Di seguito si descrivono le dimensioni specificando per ognuna di esse il nome, una descrizione e la granularità.

Nome	Descrizione	Granularità
EQO	L'EQO è l'*equipment* order associato al ciclo	Un Equipment Order
Equipment	Macchinario sottoposto al ciclo	Un Macchinario
Alarm	Allarme associato all'esecuzione del ciclo	Un Allarme
Evaluation	Valutazione data da un utente all'esecuzione del ciclo	Una Valutazione
Result	Esito del ciclo	Un Risultato
Type	Tipologia di ciclo effettuato	Una Tipologia

Tabella 3: Dimensioni

Per ogni dimensione si elencano gli attributi e una breve descrizione di ognuno. Per le dimensioni degenere, cioè che non hanno nessun attributo, non è prevista una rappresentazione tabulare.

Attributo	Descrizione
EQO Code	Identificativo dell'Equipment Order
Status	Lo status dell'Equipment Order
Batch	Il codice del lotto associato all'Equipment Order

Tabella 4: EQO

Attributo	Descrizione
Equipment ID Code	Identificativo dell'Equipment
Description	La descrizione dell'Equipment

Tabella 5: Equipment

Attributo	Descrizione
Tag	Codice descrittivo dell'allarme
Date In	Data in cui è scattato l'allarme
Date Out	Data in cui l'allarme è stato disattivato
Signature	Identificativo dell'utente che ha disattivato l'allarme
Comment	Commento relativo all'allarme

Tabella 6: Alarm

Attributo	Descrizione
Type	Esito della Valutazione
User	Utente che ha valutato il ciclo
Comment	Commento relativo alla valutazione
Date	Data in cui è avvenuta la valutazione

Tabella 7: Evaluation

Per quanto riguarda la gestione degli attributi dimensionali che possono subire modifiche nel tempo, è possibile adottare differenti strategie di storicizzazione. Si valutano quattro possibilità, delle quali le prime tre si considerano quando qualche attributo cambia raramente (*slowly changing dimensions*):

- **Tipo 1 (oggi per ieri):** il valore di un attributo dimensionale che cambia va trattato come un valore errato da sostituire con il nuovo valore. E' la soluzione più semplice, ma si perde la storia dei valori;

- **Tipo 2 (oggi o ieri):** Si vuole conservare la storia dei valori e quindi vanno memorizzati sia i vecchi che i nuovi dati. È la soluzione più comune anche se aumenta i dati della dimensione;

- **Tipo 3 (oggi e ieri):** Si vuole sia la storia dei valori che la data in cui si verifica il cambiamento;

- **Tipo 4:** gli attributi dimensionali cambiano frequentemente. In questo caso è preferibile adottare strategie diverse dalle precedenti. E' possibile per esempio optare per la separazione della dimensione in due tabelle, una contente gli attributi destinati a non cambiare nel tempo e una contenente i rimanenti, organizzati in intervalli di valori.

Nel caso di studio presentato non è prevista la necessità di storicizzare dei cambiamenti se non per il caso in cui un record non è più presente nel sistema sorgente. In questo caso nel *data warehouse* viene settato un attributo *flag* che va ad indicare che quella tupla è stata cancellata a livello della sorgente.

In particolare, non interessa mantenere lo storico delle informazioni in quanto i report che verranno prodotti devono fornire informazioni riguardanti solo i lotti di insulina

che sono stati prodotti (secondo gli standard GMP) e per i quali ancora non è stato deciso se possono essere immessi sul mercato.

4.2 Progettazione Concettuale del Data Mart

La raccolta dei requisiti e la traduzione formale di questi ha permesso di identificare requisiti specifici sul fatto, sulle misure associate e sulle dimensioni.

È possibile esprimere e rappresentare graficamente una sintesi delle informazioni raccolte attraverso il formalismo DFM (*dimensional fact model*). Si tratta di un formalismo grafico molto intuitivo e leggibile da tutti i tipi di utenti. Il fatto è modellato con un rettangolo posto in posizione centrale e diviso in due parti che contiene superiormente il nome e di seguito la misura oggetto di analisi. Tutte le dimensioni sono collegate al fatto tramite archi, che a loro volta presentano degli attributi dimensionali collegati tramite altri archi disposti a raggio. Alcuni archi presentano un taglio in quanto rappresentano dimensioni opzionali che non sempre hanno un valore.

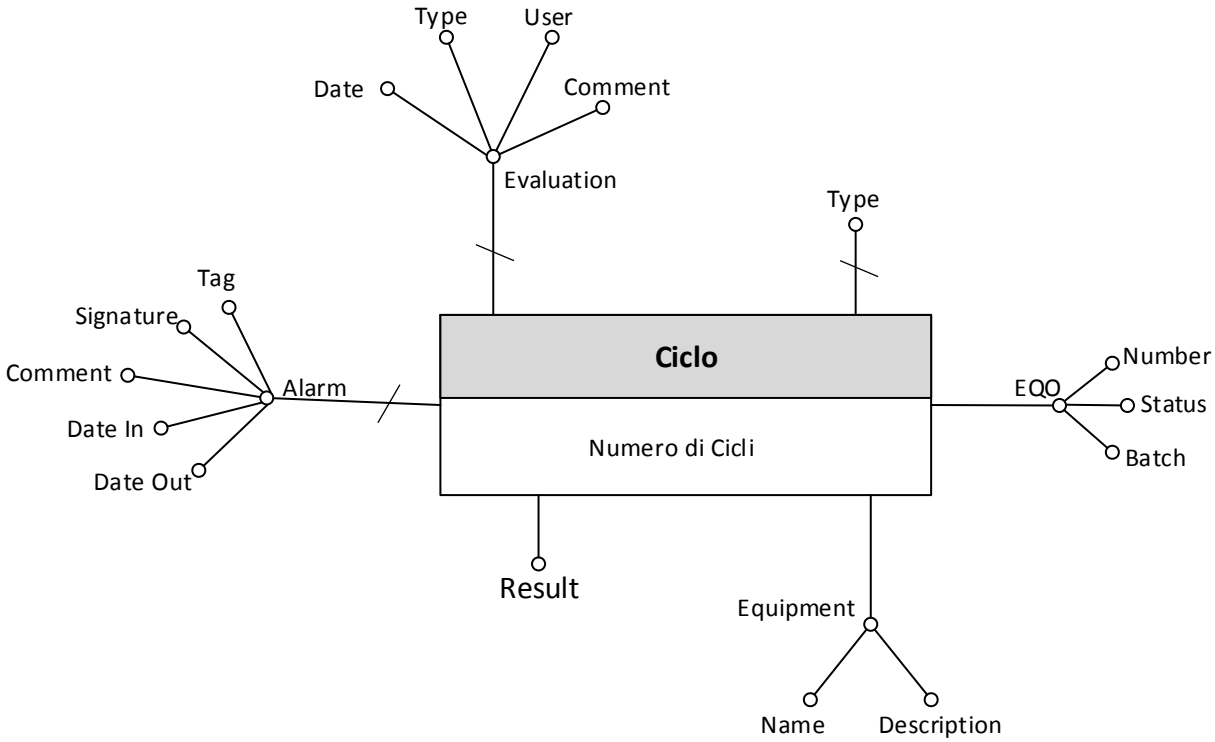


Figura 12: Modello Concettuale del *Data Mart*

4.3 Progettazione Logica del Data Mart

Definito il modello concettuale, si passa alla definizione del modello logico. La realizzazione dello schema logico avverrà secondo i seguenti passi:

- Il fatto è modellato con una tabella;
- Ogni dimensione viene modellata con una tabella;
- Le chiavi primarie delle tabelle dimensionali vengono utilizzate come chiavi esterne nella tabella del fatto;
- L'opzionalità di alcune dimensioni è implementata tramite la creazione di valori/tuple di default (con codici N/D e descrizioni del tipo Non Disponibile, Altro etc).

Il modello logico utilizza una struttura a fiocco di neve (*snowflake schema*) in quanto oltre alla tabella centrale del fatto collegata a quelle che rappresentano le dimensioni, una di queste è collegata a sua volta con un'altra tabella dimensionale che non dipende direttamente da quella del fatto. Nello specifico è la tabella MDM_BATCH_PARENTERAL_DIM, che contiene tutte le informazioni riguardanti un lotto di insulina, è collegata alla tabella MDM_BATCH_EQO_AS_DIM, che contiene i dati degli EQO e li associa ai lotti. Quest'ultima tabella ha infatti l'unico compito di associare gli EQO ai lotti di insulina. Le informazioni relative agli allarmi associati ai cicli sono ricavate a partire dalla tabella MDM_ALARMS che contiene informazioni riguardanti l'arco temporale in cui l'allarme è scattato e riguardo all'utente che se ne è occupato con i relativi commenti. Nella tabella MDM_EQUIPMENT_DIM invece possiamo trovare informazioni inerenti l'*equipment*, cioè sul macchinario, che è stato interessato dal ciclo. Nella tabella MDM_CYCLES ci sono le informazioni generali inerenti ai cicli, come ad esempio la data di inizio e fine (che ci sono utili per popolare il report). Infine, la tabella MDM_PROCESS_VALUE contiene informazioni associate al ciclo quali la tipologia, l'esito e la valutazione dello stesso.

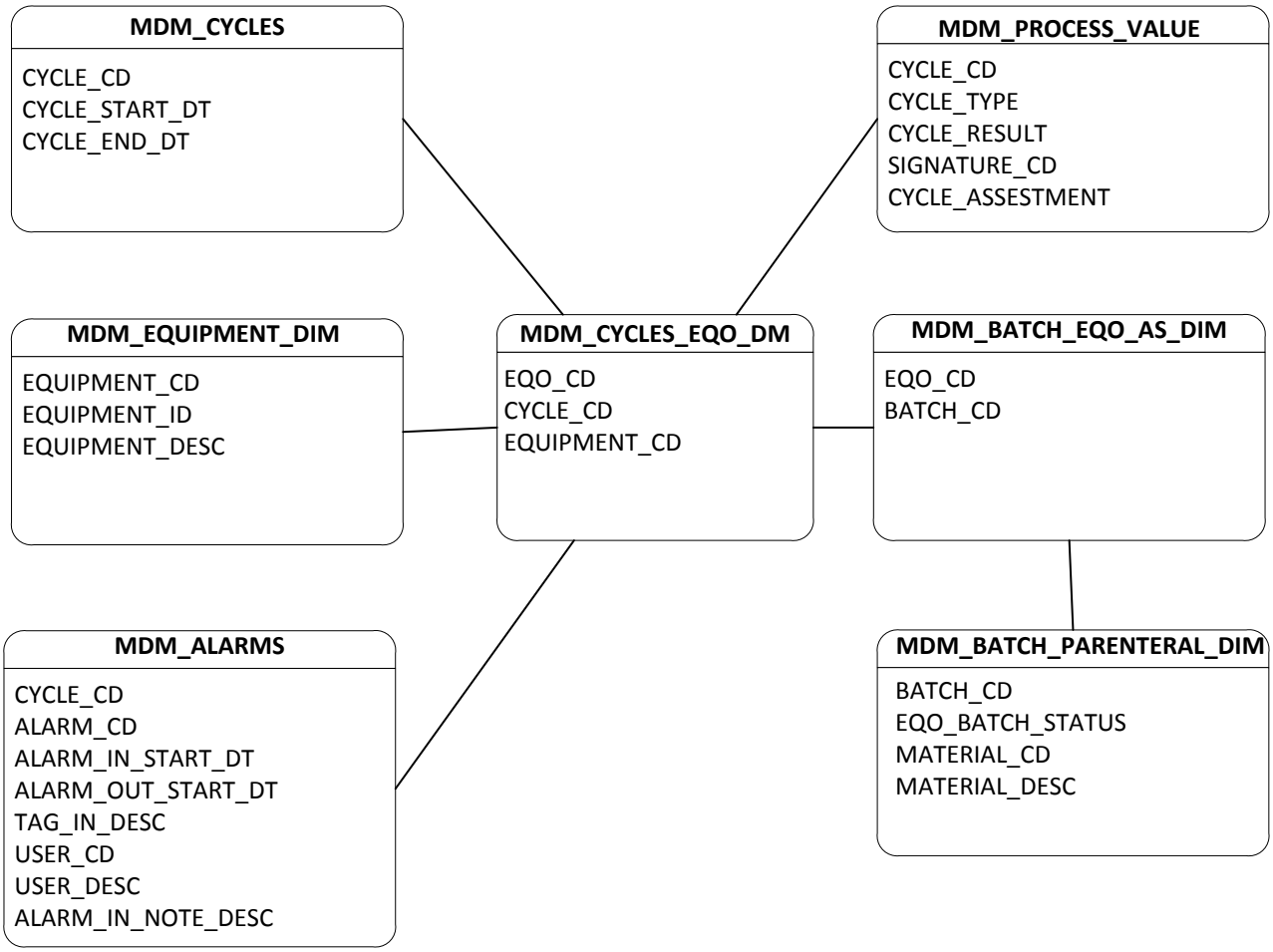


Figura 13: Modello logico del *data mart*

5 AMBIENTE DI SVILUPPO

In questo capitolo verranno descritti i principali strumenti software scelti dalla società cliente che sono stati utilizzati per le fasi di sviluppo e caricamento del *data warehouse* e realizzazione dei report.

Verranno introdotti il *database management system* (DBMS) Oracle, utilizzato per trattare i sistemi sorgente, facendo riferimento al *software Oracle Sql Developer* per la gestione e lo sviluppo di soluzioni basate sull'utilizzo del *database* Oracle. Il punto focale però si sposta sul *database* IBM PDA (*Pure Data for Analytics*, detto anche Netezza) che sarà utilizzato per trattare l'area di *staging* e il *core* con le *extension*, facendo riferimento al *software Aginity Workbench for PDA*.

Di seguito sarà trattato anche *Informatica PowerCenter*, lo strumento usato per le operazioni di estrazione, trasformazione e caricamento. Infine per la reportistica verrà introdotto *SAP Business Object*.

5.1 Aspetti Generali

Ogni tipo di applicazione *software* attraversa più fasi nel proprio ciclo di vita. Alcune di queste fasi si definiscono come preliminari in quanto precedono lo sviluppo di codice vero e proprio (analisi dei requisiti, pianificazione delle attività e progettazione del sistema), altre invece sono incentrate interamente sullo sviluppo del codice (implementazione, collaudo e rilascio), oltre a fasi complementari non trascurabili come quelle di documentazione e di manutenzione dei sistemi. Per portare avanti le diverse fasi di gestione del *software* solitamente si usano differenti ambienti. In particolare per questo progetto sono usati i seguenti ambienti:

1. **Sviluppo**: ambiente in cui si svolge la fase di implementazione. Si tratta di un ambiente in cui sono presenti il minimo indispensabile di strutture dati necessarie per la realizzazione dei componenti, e i dati presenti in questo ambiente non sono reali o comunque realistici. É proprio in questa fase che si ha la concreta realizzazione della soluzione per poi essere collaudata e presentata in seguito;

2. **Test:** ambiente in cui si testa ciò che è stato prodotto nella fase precedente, per verificare che i requisiti siano rispettati in modo corretto e in cui sono presenti dati reali;

3. **Produzione:** ambiente finale in cui il *software* è nella sua versione definitiva dopo aver superato la fase di test. In questo ambiente ci trovaimo nel contesto reale con tutti i dati necessari al funzionamento del sistema.

Una volta arrivati nell'ambiente di produzione è possibile che siano riscontrati da parte del cliente dei comportamenti indesiderati, delle anomalie o semplicemente possono sopraggiungere delle richieste di modifiche dipendenti da fattori esterni. In tal caso tutte le modifiche necessarie sono implementate in sviluppo, testate nuovamente in test, e infine rilasciate in produzione. Questa struttura è replicata su tutte le piattaforme coinvolte nel progetto. Generalmente gli ambienti di sviluppo e test sono costituiti da istanze diverse presenti su una stessa piattaforma *hardware*, mentre gli ambienti finali sono installati su macchine dedicate e di elevata qualità, per questioni prestazionali e di sicurezza.

5.2 Database Oracle

Oracle Corporation è una multinazionale del *software* che deve buona parte del suo successo ad un'intuizione della fine degli anni settanta: nel 1979, infatti, l'azienda fondata da Larry Ellison diffuse in commercio il primo *database* relazionale al mondo.

Con i suoi prodotti commerciali, Oracle è tra i primi fornitori di sistemi per *data warehouse* e basi di dati in termini di fatturato, e si posiziona tra i leader del mercato nel *Magic Quadrant* di Gartner. [Gartner 16]

Figura 14: Logo di Oracle

Tra i clienti Oracle ci sono oltre duemila istituzioni ed enti pubblici in tutto il mondo, il suo successo è legato all'alta affidabilità, alla sicurezza dei dati e alla potenza dell'architettura su cui si basa. Un server Oracle è rappresentato fondamentalmente da due strutture, il database e l'istanza. Con il termine database si indicano i *file* fisici in cui sono memorizzati i dati, mentre per istanza si intende l'insieme delle aree di memoria e dei processi di *background* necessari per accedere al *database*. Ogni *database* deve obbligatoriamente fare riferimento almeno ad un'istanza, è anche possibile avere più di un'istanza per *database*. Ciascun *database* consiste di strutture logiche di memorizzazione, per immagazzinare e gestire i dati, e di strutture fisiche di memorizzazione che contengono le strutture logiche.

Figura 15: *Magic Quadrant* di Gartner per *Operational* DBMS

Il sistema utilizzato in questa implementazione è *Oracle Database 11g Release 2*, un database management system (DBMS) relazionale a oggetti, che supporta le principali funzioni SQL analitiche. Tra le varie potenzialità la principale è la programmabilità, ossia la possibilità di poter memorizzare ed eseguire procedure e funzioni in linguaggio Oracle PL/SQL.

5.3 Oracle SQL Developer

Oracle SQL Developer è un ambiente di sviluppo integrato che fornisce agli sviluppatori di *database* un semplice modo per connettersi a un *database schema*, gestire gli oggetti, eseguire istruzioni SQL e *script* , sviluppare ed eseguire il codice PL/SQL ed esportare i dati. Nel progetto in questione, SQL Developer è stato utilizzato per gestire il *database* Oracle ed eseguire *query*.

Oracle SQL Developer è un'applicazione *client* di *database server*, sviluppato da Oracle Corporation per l'interazione con i *database*, dotata di tutte le caratteristiche di un ambiente di sviluppo integrato (IDE) per la programmazione SQL. Lo scopo principale di Oracle SQL Developer è quello di facilitare la creazione, la modifica e la gestione degli oggetti di un *database* relazionale e di rendere ciò comprensibile anche a chi non conosce

il linguaggio SQL. Questa applicazione funge quindi anche da terminale *client* che presenta un'interfaccia grafica che permette di compiere tutte le operazioni che si possono eseguire con il linguaggio SQL. SQL Developer permette di interfacciarsi con tutti i *database* Oracle anche di precedenti versioni.

Nella figura sottostante viene mostrata una schermata tipo in cui a sinistra sono presenti i vari schema a cui ci si può connettere e a destra una finestra in cui è possibile interrogare il *database* o modificare le strutture dati.

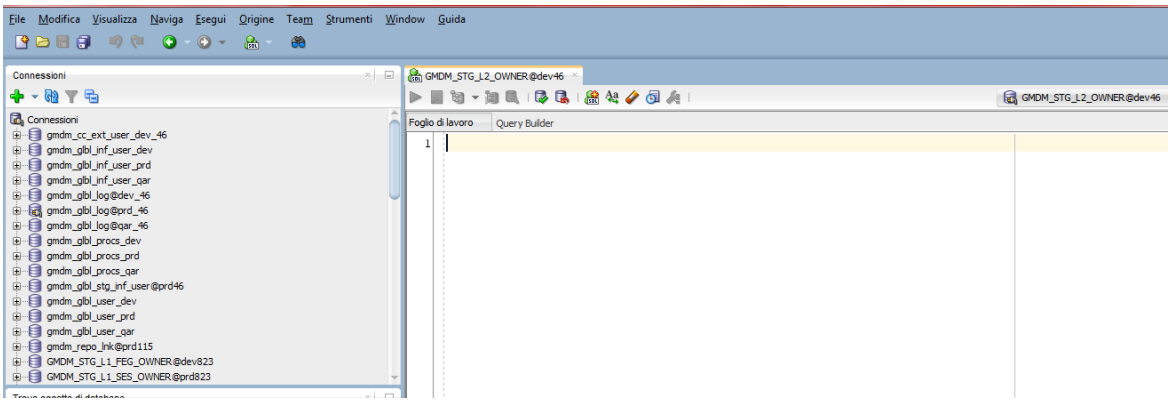


Figura 16: Schermata principale di SQL Developer

5.4 Database IBM PDA

IBM (*International Business Machines Corporation*), chiamata anche *Big Blue*, è una compagnia multinazionale tecnologica americana che ha origine nel 1884 e che al giorno d'oggi opera in più di 170 paesi al mondo.

Nel 1955, IBM presenta il suo primo calcolatore, benché partita in ritardo, grazie ai suoi modelli organizzativi, *Big Blue* riesce a conquistare un ruolo da leader nel settore. Nel 1981 viene lanciato l'IBM Personal Computer che era dotato dell'Intel 8088 a 16bit è una vera ventata di novità, considerando che il mercato è dominato da CPU ad 8bit, così come lo è la separazione della tastiera dal corpo centrale, al contrario della tendenza del periodo.

Con i suoi prodotti commerciali, IBM è tra i primi fornitori di sistemi per *data warehouse* e basi di dati in termini di fatturato, e si posiziona tra i leader del mercato nel Magic Quadrant di Gartner [Gartner 16].

Il sistema utilizzato in questa implementazione è *Netezza 7.2*. Netezza (o *Pure Data Analytics*) è un'*appliance*, ovvero un unico dispositivo, che include un DBMS e un *hardware* dedicato, ottimizzata per il *data warehouse*, interrogabile in SQL standard via ODBC da gran parte dei *client* e dei *front end* di *Business Intelligence*.

NETEZZA® an IBM® Company

Figura 17: Logo di Netezza

La tecnologia Netezza è basata sull'elaborazione a parallelismo massivo dei dati (*Massive Parallel Processing*, MPP) in cui ci sono N nodi di calcolo con *hardware* dedicato che lavorano tutti in parallelo per ogni *query*, ogni nodo è infatti *Shared Nothing*, cioè che non condivide nulla con gli altri. Per ottimizzare i nodi questi devono avere un carico omogoneo di dati.

Davanti ai nodi c'è l'*host*, quello con cui l'utente si interfaccia. Lui traduce la nostra *query* in codice C, e la manda in *broadcast* a tutti i nodi. Ogni nodo riceve il suo codice C, e lo prende in pasto (*snippet*, un valore >10 indica una *query* complessa).

L'esecuzione del codice è quindi sempre parallela, e coinvolge sempre tutto l'*hardware* disponibile.

In ogni nodo, le operazioni elementari sui dati vengono eseguita dall'FPGA (*Field Programmable Gate Array*, circuito integrato le cui funzionalità sono programmabili via *software*) per ridurre il carico di lavoro della CPU e ottimizzare il *throuput* totale.

Rispetto ad Oracle e ai RDBMS tradizionali su Netezza:

- non ci sono indici, quindi ogni *query* esegue un *full table scan* (interrogazioni puntuali su singoli record potrebbero essere più lente);

- i *constraint* (*uniqueness*, *primary key*) non sono validati (ad eccezione del NOT NULL) e quindi si possono inserire record multipli con la stessa chiave;

- ogni connessione è associata ad uno specifico *database* (*Cross database access*), si può leggere da qualsiasi *database* ma scrivere solo su quello a cui si è connessi;

- è prevista una *Zone Map*, per ogni blocco del disco Netezza riserva uno spazio per scrivere il valore minimo e massimo *integer* (*integer* o *data*) contenuto in quel blocco;

- le *delete* su Netezza vengono solo contrassegnate come cancellate con un *flag integer*, perché fa prima a contrassegnarla che a cancellarla fisicamente. Per cancellarle fisicamente si usa saltuariamente il comando GROOM;

- Netezza, da un punto di vista delle prestazioni, preferisce fare una *query* grossa piuttosto che tante piccole *query*.

5.5 Aginity Workbech for PDA

Aginity Workbench for PDA (da ora Aginity) è una ambiente di sviluppo integrato che fornisce agli sviluppatori di *database* un semplice modo per connettersi a un *database* schema, gestire gli oggetti, eseguire istruzioni SQL e *script*, sviluppare ed eseguire il codice PL/SQL, esportare i dati e molte altre attività di base. Aginity è stato utilizzato per gestire il *database* PDA ed eseguire *query*.

Aginity è un'applicazione facile da usare di elevate performance ed efficienza quando si lavora con un *data warehouse* di tipo MPP (*Massively Parallel Processing*).

Riconosciuto come il migliore *query tool* MPP, fornisce un potente set di strumenti basati su GUI con più di 60.000 sviluppatori, DBA, e analisti di dati.

Questo *tool* offre funzionalità uniche e un'interfaccia utente molto intuitiva che permette di creare, gestire e manipolare *query* SQL e interi schema di *database*. Tra le funzionalità di Aginity le principali sono:

- Auto-completamento: permette di creare facilmente e velocemente le query SQL.
- Query Parametrizzate: creando dei template con valori parametrizzati è facile richiamare la stessa *query* per essere utilizzata in diversi contesti.
- Stored Procedure Wizard: permette di creare rapidamente delle *stored procedure*.

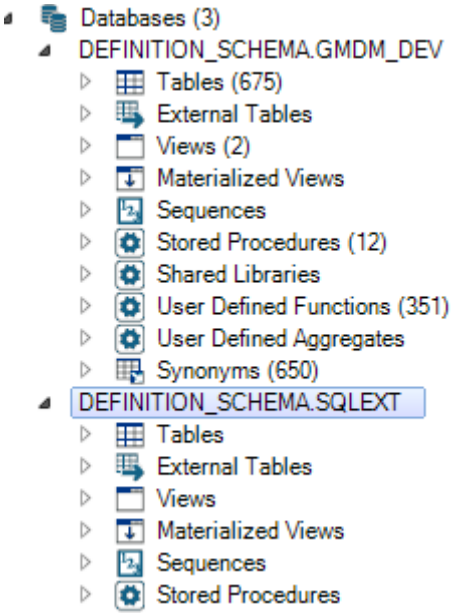


Figura 18: Pannello di navigazione di Aginity

Nella schermata iniziale è presente una finestra dove, tramite dei parametri di connessione, ci si può collegare al *database*. Una volta connessi si può navigare tra le varie strutture dati e operare in SQL. Nella figura sottostante notiamo l'elenco a tendina che compare lateralmente e che presenta le strutture a cui si ha accesso divise per schema, nello specifico abbiamo quelle appartenenti all'area GMDM *core* (GMDM_DEV) e quelle appartenenti all'area *extension* (SQLEXT). Allo stesso tempo possono essere aperte più connessioni ed interrogare il *database* con vari utenti.

5.6 Informatica PowerCenter

Informatica PowerCenter [Informatica 13] è uno strumento di ETL per lo sviluppo di *data warehouse* prodotto dalla società Informatica Corporation, considerato il principale fornitore indipendente di *software* per l'integrazione dei dati.

Figura 19: Logo di Informatica PowerCenter

Secondo il *Magic Quadrant* di Gartner per i *tool* di *data integration*, Informatica continua a crescere e a mantenere la propria posizione da leader del Mercato da ormai 10 anni consecutivi [Gartner 16].

Figura 20: *Magic Quadrant* di Gartner per *Data Integration Tools*

PowerCenter permette di accedere e quindi estrarre i dati provenienti da diverse fonti eterogenee di aziende e organizzazioni di qualsiasi dimensione, applicando una varietà di trasformazioni secondo la logica di business desiderata, per poi integrarli nelle strutture informative predisposte come un *data warehouse*, tutto questo con elevate prestazioni in termini di precisione e tempestività.

Informatica PowerCenter è suddiviso in quattro componenti: Repository Manager, Designer, Workflow Manager e Workflow Monitor.

5.6.1 Architettura

L'architettura può essere divisa in due macro aree, il dominio e gli strumenti *client*. Il dominio è l'unità primaria per la gestione e l'amministrazione all'interno di PowerCenter. È la raccolta di tutti i *server* necessari per supportare le funzionalità di PowerCenter. Ogni volta che si desidera utilizzare i servizi di PowerCenter si invia una richiesta al *server* di dominio.

Repository Service e Integration Service sono componenti del dominio di PowerCenter. Il Repository Service gestisce le connessioni tra il *client* e il *repository*. È un separato processo *multi-threaded*, che recupera, inserisce, e aggiorna i metadati nelle tabelle del *repository*. Esso assicura la coerenza dei metadati nel *repository*.

Quando si avvia un flusso di lavoro (ovvero un insieme di istruzioni che descrive come e quando eseguire compiti legati all'estrazione, alla trasformazione e al caricamento dei dati) parte l'Integration Service che legge le informazioni dal *repository* per recuperare i metadati. Assegna e materializza i valori delle variabili/parametri estratti dalle sorgenti, applica le regole di trasformazioni configurati nel flusso di lavoro (*workflow*) e memorizza i dati. Infine, carica i dati trasformati nelle destinazioni. Questo servizio riesce a combinare dati provenienti da diverse piattaforme e sorgenti eterogene. Inoltre può caricare i dati in diverse piattaforme e destinazioni eterogene.

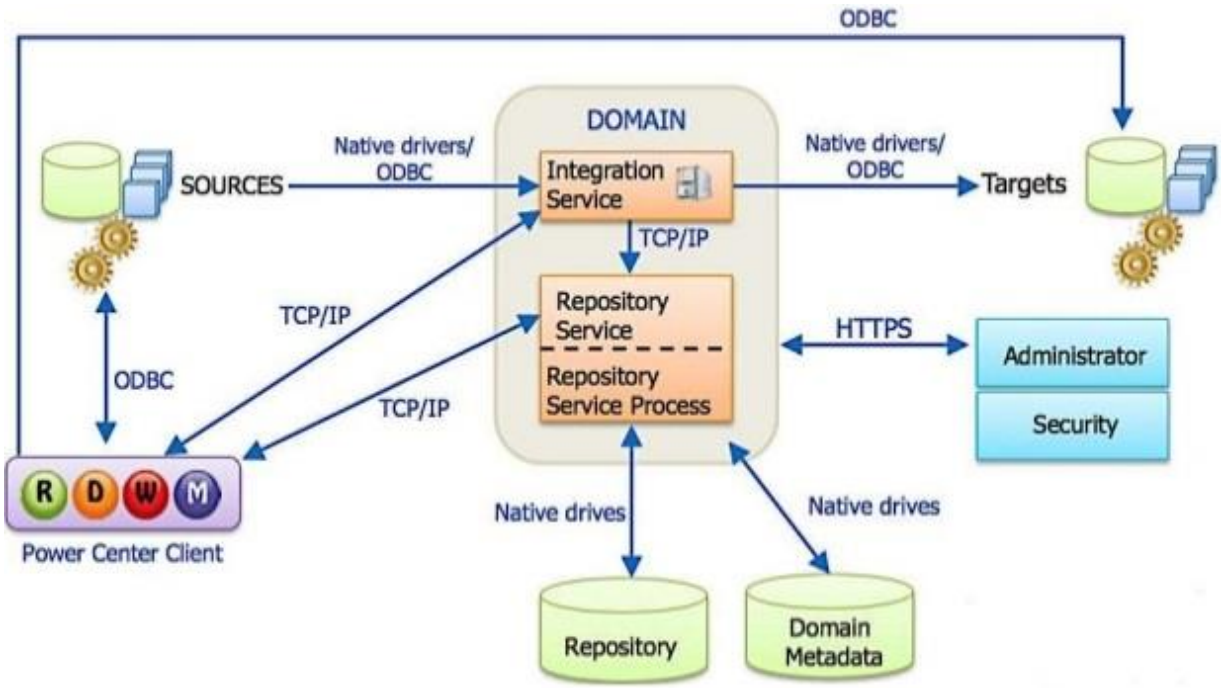


Figura 21: Architettura di Informatica PowerCenter

Gli strumenti *client* sono utilizzati per gestire gli utenti, definire le origini e le destinazioni e creare flussi di lavoro con logiche di trasformazione. Fanno parte degli strumenti client:

- **Repository Manager:** che si occupa delle attività di gestione dei *repository*, assegnando privilegi e autorizzazioni a utenti o gruppi, permettendo di visualizzare i metadati dei *repository*;

- **Designer:** serve a creare le mappature o i flussi dati che contengono le istruzioni di trasformazione per l'Integration Service;

- **Workflow Manager:** utilizzato per creare, gestire, pianificare e eseguire i *workflow*;

- **Workflow Monitor:** utilizzato per monitorare i *workflow* schedulati e in esecuzione per ogni Integration Service.

5.6.2 Repository Manager

Attraverso il Repository Manager è possibile osservare gli oggetti contenuti su un *repository* e svolgere su di essi le operazioni fondamentali di ricerca, confronto, analisi delle dipendenze, convalida, creazioni di chiavi di accesso.

Ogni *repository* è navigabile attraverso il Repository Manager e appartiene ad un dominio. Prima di poter analizzare il *repository* è quindi necessario fornire le informazioni per la connessione al dominio. Una volta effettuato l'accesso a un *repository* è possibile navigare tra gli oggetti dello stesso tipo creati con il Designer (*source*, *target*, trasformazioni, *mapplet* e *mapping*) o con il Workflow Manager (*task*, *session*, *worklet* e *workflow*) organizzati e archiviati allinterno di cartelle distinte a seconda della tipologia. La finestra principale del Repository Manager, infine, visualizza informazioni riguardanti l'oggetto selezionato nella finestra di navigazione permettendo un'analisi dettagliata delle sue proprietà.

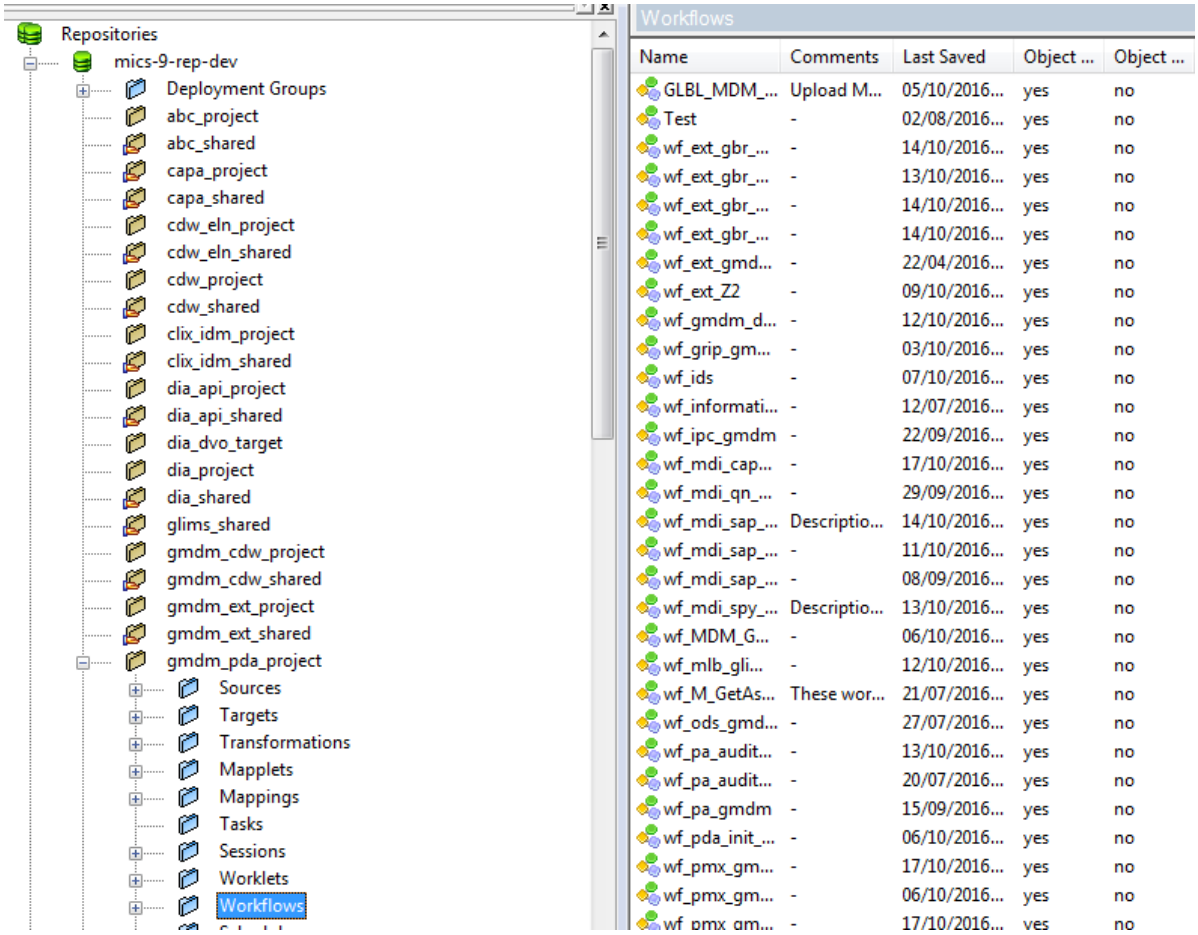


Figura 22: Schermata principale del Repository Manager

5.6.3 Designer

Il Designer permette di progettare i *mapping* e i *mapplet*. Un *mapping* è un insieme di definizioni di origine e di destinazione collegati da oggetti di trasformazione che definiscono le regole per la trasformazione dei dati. Il *mapplet* è un oggetto riutilizzabile, che contiene una serie di trasformazioni e consente di riutilizzare la logica di trasformazione in più *mapping*.

Un *mapping* rappresenta il flusso di dati tra origine e destinazione. Quando l'Integration Service viene eseguito, una *session* (definita e creata nel Workflow Manager) utilizza le istruzioni configurate nel *mapping* per leggere, trasformare e scrivere i dati. Ogni *mapping* deve contenere i seguenti componenti:

- La definizione della Sorgente, che descrive le caratteristiche di una tabella di origine o di un *file*;
- Le Trasformazioni, per modificare i dati prima della scrittura nella destinazione. Si possono utilizzare diversi oggetti di trasformazione per svolgere funzioni diverse;
- La definizione della Target, che definisce la tabella o il *file* di destinazione;
- I Links, ovvero i collegamenti tra le sorgenti e il *target*, e le trasformazioni in modo che l'Integration Service sia in grado di spostare i dati che trasforma. Inoltre, un *mapping* può anche contenere uno o più *mapplet*.

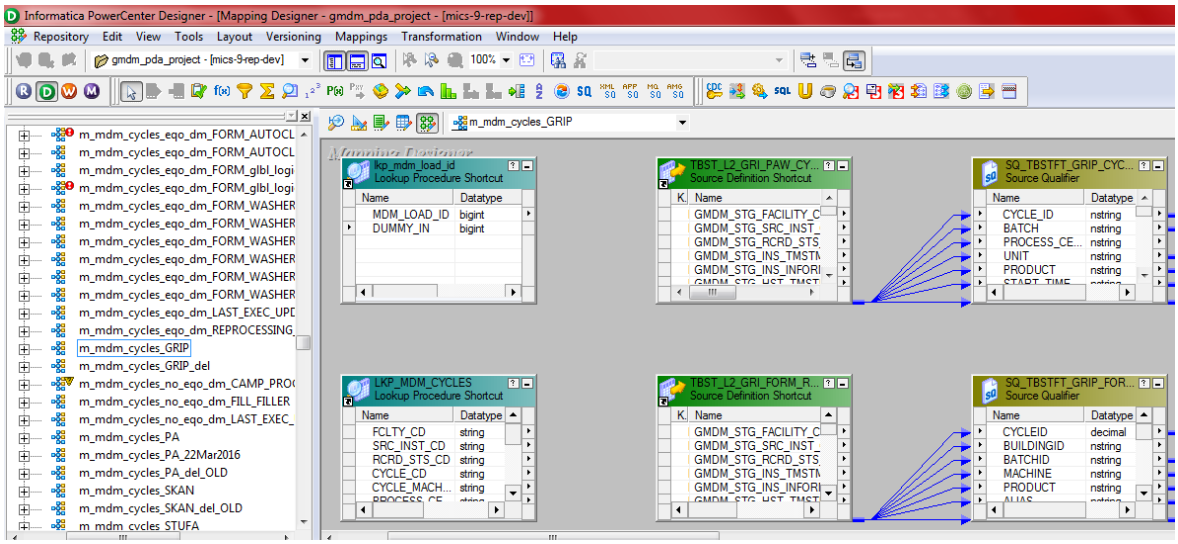


Figura 23: Schermata principale di Designer

Come possiamo vedere dalla figura nel Designer è presente la finestra di navigazione laterale, dove è possibile visualizzare gli oggetti che fanno parte del *repository* a cui si è connessi, permettendo di lavorare su più cartelle e *repository* in una sola volta. A destra invece abbiamo la finestra principale in cui è possibile avere la visione di diversi elementi a seconda dello strumento che si sta utilizzando, nello specifico nella figura è visualizzato un *mapping* in quanto si sta utilizzando il Mapping Designer. Nella figura si

possono visualizzare delle trasformazioni, delle tabelle sorgenti e dei *source qualifier*. Il Designer comprende quattro diversi strumenti:

- **Source Analyzer:** usato per importare o creare gli oggetti di origine;
- **Target Designer:** usato per importare o creare gli oggetti di destinazione;
- **Transformation Developer:** usato per la creazione delle trasformazioni riutilizzabili come le *look up*;
- **Mapplet designer:** usato per creare i *mapplet*;
- **Mapping Designer:** usato per creare i *mapping*.

Quado ci si trova nel Mapping Designer è possibile trascinare all'interno della finestra di lavoro tutti gli oggetti che servono senza limitazioni a condizione che siano già stati importati o definiti.

Per le sorgenti che provengono da un *database* relazionale è stato abilitato automaticamente il *source qualifier*, un'opzione con cui è possibile personalizzare la selezione di dati utilizzando una *query* SQL.

Tutte le trasformazioni (le operazioni da effettuare sui dati) vengono progettate con l'ausilio di un'interfaccia grafica che permette di collegare *source*, trasformazioni e *target* mediante l'uso dei *link* che collegano le porte corrispondenti ai campi di input (*source*) a quelle che corrispondono qi campi di output (*target*).

Le trasformazioni si dividono in passive, ovvero che non cambiano il numero di righe ricevute, e attive, che invece possono cambiare il numero di righe ricevute.

L'elemento caratterizzante delle trasformazioni sono le espressioni. Un'espressione è un'istruzione condizionale o un calcolo che può essere aggiunto a una trasformazione, che consente di modificare, aggiungere o sopprimere singole porte di una singola riga. I dati possono essere modificati utilizzando gli operatori logici e numerici o le funzioni *built-in*. Esempio di trasformazioni gestite attraverso le espressioni, possono essere:

- Manipolazione dei dati, come concatenazione, troncamento, arrotondamento (CONCAT, LTRIM, UPPER, INITCAP);
- Conversione (TO_DECIMAL, TO_CHAR, TO_DATE);
- Pulizia dei dati, come controllo dei valori NULL, sostituzione dei caratteri, controllo di spazi o per numeri (ISNULL, ReplaceStr);
- Calcoli scientifici o Operazioni numeriche, come esponenziali, logaritmi, moduli, elevamento a potenza (LOG, POWER, SQRT);
- Operazioni specifiche dell'ETL, come if, lookup, decode (IIF, DECODE).

Un'espressione è costituita da porte (input, input/output, variabili), funzioni, operatori, variabili, valori di ritorno e costanti. Le espressioni possono essere utilizzate nei seguenti oggetti di trasformazione:

- Expression, trasformazione di tipo passivo per la manipolazione dei dati;
- Aggregator, trasformazione di tipo attivo per eseguire calcoli di aggregazione;
- Filter, trasformazione di tipo attivo per eseguire filtri sui dati;
- Update Strategy, trasformazione di tipo attivo per determinare se inserire, cancellare, aggiornare o rifiutare le righe nella tabella target.

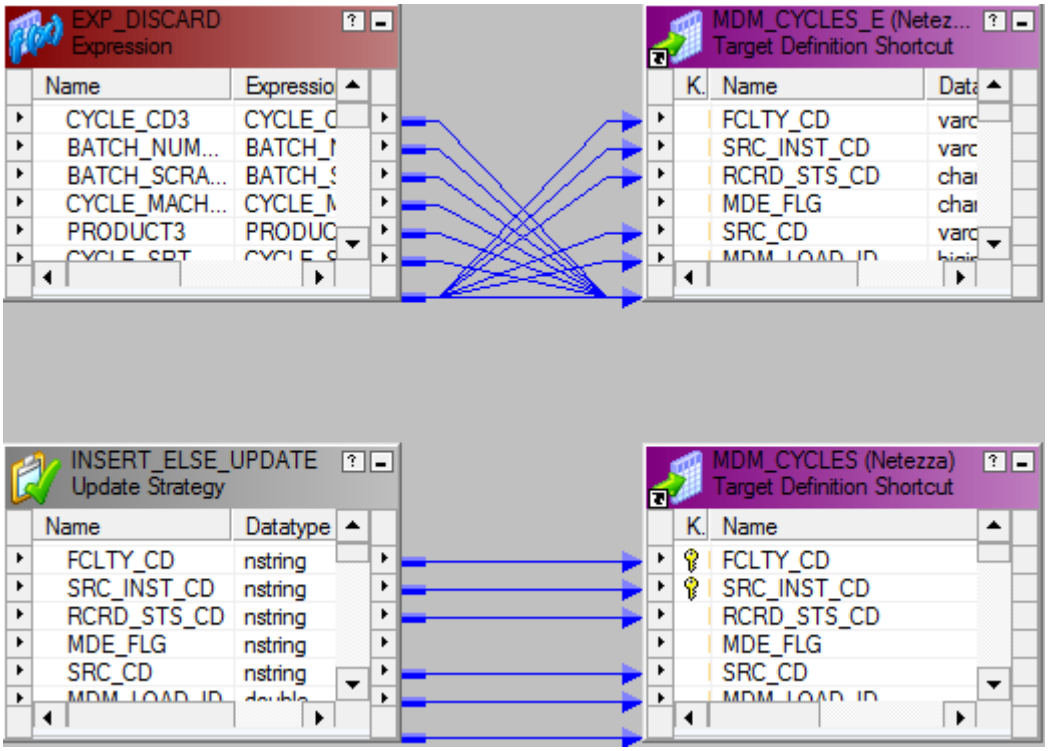


Figura 24: Oggetti di Trasformazione e tabelle Target

Nella figura in alto abbiamo in dettaglio una porzione di *mapping* che presenta due oggetti di trasformzione, l'Expression e l'Update Strategy, e due tabelle *target*, quella nel cui nome presenta il suffisso *_E* indica che si tratta di una tabella degli scarti.

Due oggetti di trasformazione, che meritano attenzione in quanto usati in tutti i *mapping* creati durante il lavoro svolto, sono l'oggetto Lookup e l'oggetto Update Strategy. Gli oggetti indicati, vengono usati in combinazione per andare a definire la politica di inserimento dei dati sulla tabella target. Tramite essi è possibile andare a verificare la presenza o meno degli attributi chiave che si tenta di inserire nella *target*, se non è presente si andrà a inserire l'intera riga, mentre se la chiave è già presente nella tabella *target* si andrà ad aggiornare gli attributi non chiave della corrispondente tupla.

Per prima cosa si andrà a definire l'oggetto Lookup. Questo oggetto di trasformazione può essere sia scollegato che collegato al flusso dati. Una trasformazione sconnessa non

sarà collegata ad altre trasformazioni del *mapping*, ma sarà chiamato in un'altra trasformazione, restituendo un valore per quella trasformazione. L'Integration Service, interroga la fonte di ricerca in base alle porte di ricerca nella trasformazione e di una condizione di ricerca. L'oggetto Lookup restituirà il risultato della ricerca per la *target* o per un'altra trasformazione.

L'oggetto Update Strategy invece permette di specificare la strategia da attuare con i dati in ingresso. Si può infatti definire che si tratti di una *delete* dei dati, oppure di una *insert/update,* quest'ultima solo in caso di dati che sono già presenti sulla tabella *target*. Nella figura sottostante vediamo come nella finestra di *edit* dell'Update Strategy sia definito che se il risultato della Lookup è nullo allora si deve procedere a un inserimento, altrimenti a un aggiornamento del record. A sinistra invece abbiamo un pannello di navigazione da cui si possono scegliere le varie funzioni da applicare.

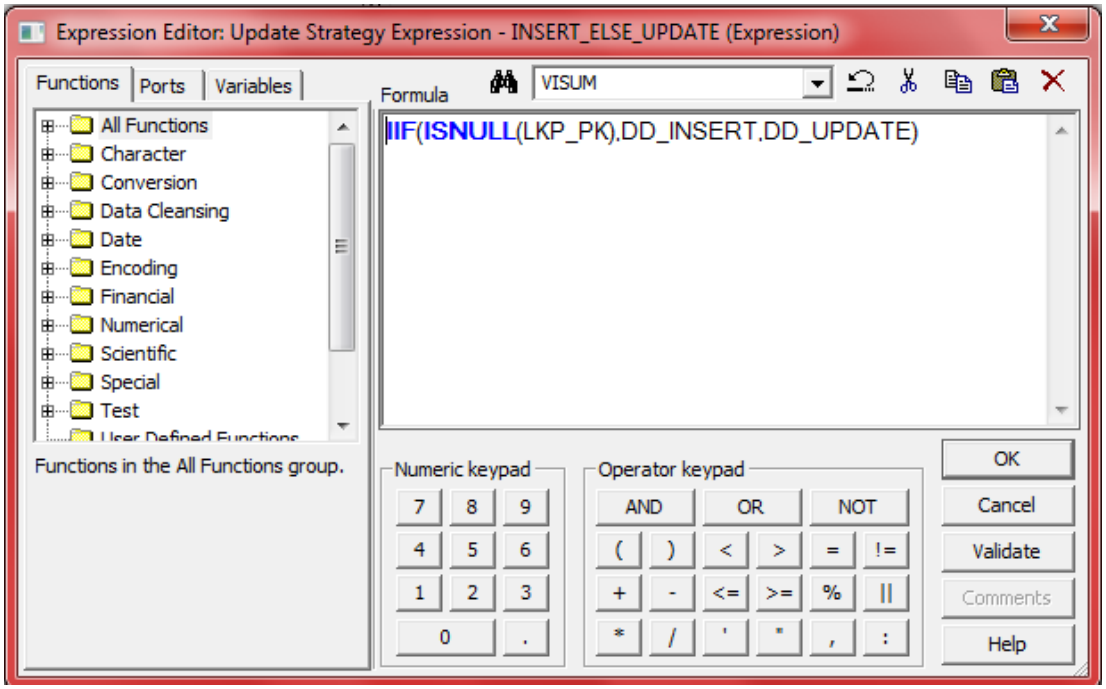


Figura 25: Finestra di *edit* di Update Strategy

5.6.4 Workflow Manager

Nel Workflow Manager, si definisce un insieme di istruzioni chiamato flusso di lavoro (*workflow*), per eseguire i *mapping* definiti con il Designer. Un *workflow* contiene una *session* e qualsiasi altra attività (*task*) che può essere eseguita quando si esegue una *session*. Tramite il Workflow Manager è possibile anche definire le *worklet*. Una *worklet* è un oggetto che raggruppa un insieme di *task*, è simile ad un *workflow*, ma senza informazioni di pianificazione. È possibile eseguire una serie di *worklet* all'interno di un *workflow*. Il Workflow Manager è costituito da tre strumenti utili alla creazione dei *workflow*:

- Task Developer, utilizzato per creare i *task* che si desidera eseguire nel *workflow*;
- Workflow Designer, utilizzato per creare un *workflow*, andando a specificare l'ordine di esecuzione dei *task* mediante *link*;
- Worklet Designer, utilizzato per creare le *worklet*.

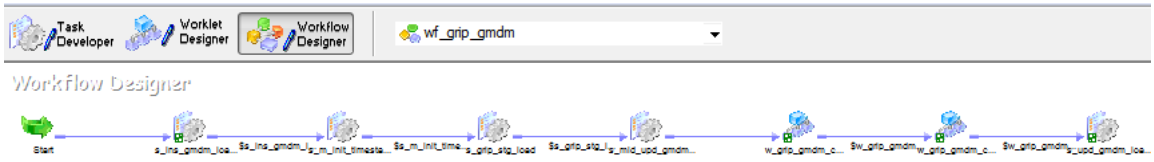


Figura 26: Finestra principale di Workflow Manager

Nella figura 26 vediamo un *workflow* usato per il progetto. Inizialmente è presente l'elemento Start, indicato con una freccia verde, che ci indica da dove ha inizio il *workflow* e poi tramite i *link* si susseguono le *session* e le *worklet*. Per i *workflow* trattati in questo elaborato gli unici *task* che sono stati usati sono le *session*, e le *worklet* (anch'esse contenenti solo *session*).

Una *session* è un insieme di istruzioni che indicano all'Integration Service come e quando spostare i dati dalla sorgente alla *target*. Infatti durante la creazione di una *session* si deve specificare il *mapping* che si desidera eseguire dall'Integration Service.

Quando si crea una *session*, è possibile settare informazioni, come il nome della *session*, l'Integration Service per eseguire la *session*, selezionare le opzioni per eseguire comandi di *shell* pre-*session*, specificare le connessioni degli oggetti del *mapping*, come le sorgenti, i *target* o le *lookup*.

Una volta definiti i *task* e il *workflow* che conterrà i *task* creati, tramite il Workflow Manager è possibile settare la schedulazione. Una schedulazione è un oggetto di *repository* che conterrà una serie di impostazioni di pianificazione. Tramite la schedulazione è possibile pianificare l'esecuzione continua di un *workflow*, o la ripetizione in un determinato momento o intervallo. In questo progetto la schedulazione dei *workflow* non è stata implementata tramite Informatica PowerCenter bensì, a causa di alcuni problemi riscontrati, tramite *crontab* di unix.

5.6.5 Workflow Monitor

Tramite il Workflow Monitor è possibile monitorare in tempo reale i dettagli relativi all'esecuzione dei *workflow* che fanno parte del *repository* su cui si è connessi. È inoltre possibile visionare i *workflow* che hanno già terminato la propria esecuzione e quelli che sono schedulati e quindi prossimi ad essere avviati. Il Workflow Monitor permette di avviare, fermare o bloccare l'esecuzione dei flussi, visualizzare statistiche relative agli oggetti monitorati, analizzare nome, tempo di inizio e stato dei flussi, visualizzare il numero delle righe che sono state caricate con successo nei *target* e visualizzare la schedulazione dei flussi. In caso di errori di esecuzione o per visualizzare dettagli più specifici è possibile visualizzare il *file* di log che fornisce in ordine cronologico tutte le informazioni che riguardono ogni passo dell'esecuzione del *workflow*.

Nella figura 27 notiamo nel dettaglio alcune *session* dove è visualizzato il nome, i dettagli sul tempo di inizio e fine esecuzione, e infine lo status che ci informa se l'esecuzione sia terminata con successo o con qualche errore. In basso abbiamo il dettaglio sulla *session* selezionata con il numero di righe che sono state selezionate dal *source qualifier* e con il numerodi quelle che sono state poi scritte sulla tabella *target*.

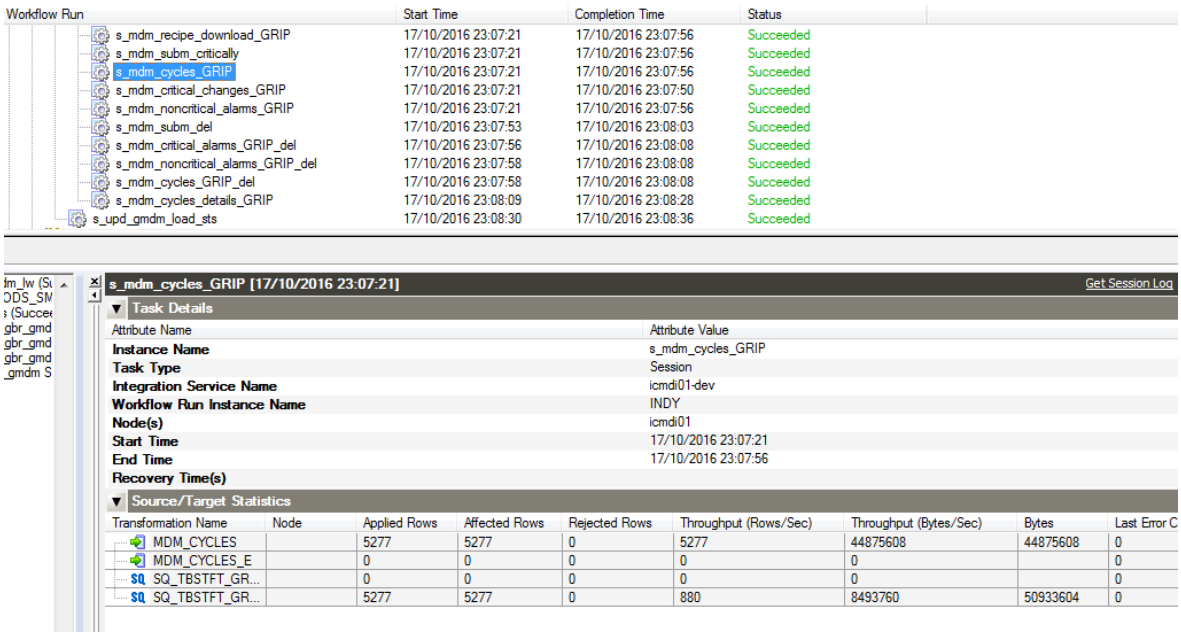


Figura 27: Finestra principale di Workflow Monitor

5.6.6 Nomenclatura in Informatica PowerCenter

Riguardo la nomenclatura da adottare per i costrutti del *tool* Informatica PowerCenter sono state adottate le seguenti regole:

- Mapping: m_<NomeTabellaTarget>;
- Session: s_<NomeTabellaTarget>;
- Worklet: w_<source_system>_gmdm_<fase>_<type>, dove <fase> indica il livello dell'architettura (stg, core, ext), e <type> la tipologia della tabella *target* (AN, FT, TMP, rispettivamente Anagrafica, Fatti e Temporanea);
- Workflow: wf_<source_system>_gmdm_<sito>, dove <sito> indica il sito produttivo.

5.7 SAP Business Objects

SAP Business Objects [SAP 12] è un *software* dedicato alla *Business Intelligence* finalizzato alla produzione di reportistica sui dati. I dati da analizzare sono archiviati su *database* esterni e le diverse entità e le relazioni tra i dati sono descritte negli Universi (metadati).

Figura 28: Logo di SAP Business Objects

Questo strumento consente di:

- modellizzare una base dati;
- realizzare dei report sulla base dei modelli prodotti;
- distribuire i report;
- pubblicare i report su web.

Lo strumento si compone di diversi moduli, ma verranno trattati in seguito unicamente quelli utilizzati durante lo svolgimento di questo progetto.

5.7.1 Universe Design Tool

Universe Design Tool è uno strumento contenuto all'interno della suite di SAP Business Intelligence che permette la creazione degli universi su cui modellare le analisi successive ai fini della reportistica. Un universo di Business Object è uno strato di metadati che serve per fare un *mapping* dei campi delle tabelle di un *database* in oggetti di business, visibili a utenti non esperti. Un universo è un file che contiene:

- I parametri di connessione, per uno o più *middleware database*;
- Gli oggetti, che mappano le strutture SQL nel *database* come le colonne, le tabelle e le funzioni del *database*;
- Uno schema, delle tabelle e delle *join* utilizzate nel *database*. Gli oggetti vengono creati in base alle strutture del *database* incluse nello schema. Lo schema è visibile solo per gli utenti di Universe Design Tool ma non per gli utenti di Web Intelligence.

Universe Design Tool viene usato per creare oggetti che rappresentano strutture di *database*, ad esempio, colonne, funzioni a cui gli utenti devono accedere e su cui devono eseguire *query*, per ottenere informazioni necessarie per soddisfare i requisiti aziendali. Gli oggetti creati nell'universo devono riguardare il campo di attività dell'utente finale e rispecchiarne il vocabolario. Il loro ruolo consiste nel presentare all'utente un'interfaccia che gli è familiare, basata sulle strutture SQL nel *database*.

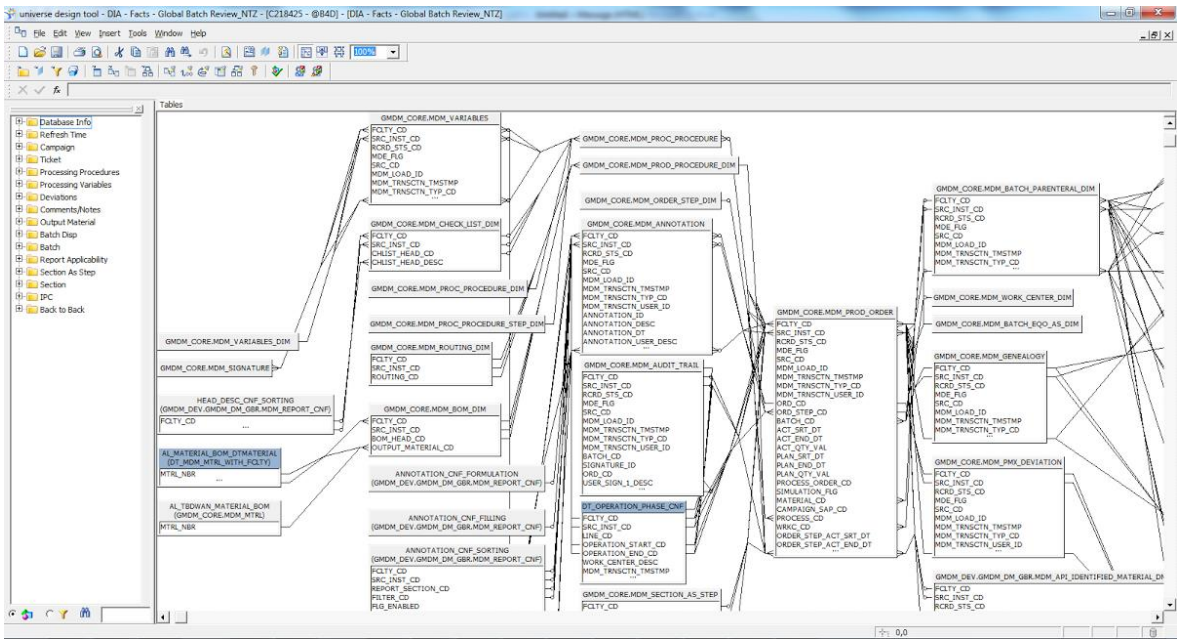


Figura 29: Schermata principale di Universe Design Tool

Nella figura sovrastante vengono visualizzate, sotto forma di diagramma, le varie classi. Nel pannello a sinistra sono presenti varie cartelle che rappresentano le classi disponibili

per la creazione dell'universo, che contengono le varie tipologie di oggetto. Dopo la creazione e la validazione dell'universo creato, un'ulteriore fase nel processo di sviluppo dell'universo riguarda l'esportazione di quest'ultimo su un *repository* centralizzato CMS (*Central Management System*) al fine di rendere disponibile (agli utenti autorizzati) le classi e gli oggetti creati con l'obiettivo di rendere fruibili le informazioni per la reportistica. Le informazioni presenti nell'universo saranno successivamente ereditate in input dal software SAP Business Object Web Intelligence, utilizzato per la reportistica.

5.7.2 SAP Business Objects Web Intelligence (WEBI)

La reportistica realizzata per questo lavoro è stata generata utilizzando il *tool* Sap Business Objects Web Intelligence, dove i dati in ingresso provengono da universi. Lo strumento consente di eseguire le analisi dei dati creando report basati sui dati che si desidera analizzare o aprendo documenti già esistenti. Tramite questo strumento è possibile analizzare i dati dei report, ad esempio eseguendo il filtraggio di alcuni campi, il *drill down* per rivelare maggiori dettagli o visualizzando i dati in grafici. I dati in ingresso al *software* possono provenire da:

- Universi;
- Fornitori di dati personali come *file* di Microsoft Excel o *file* in formato CSV, da servizi Web o da spazi di lavoro di Advanced Analysis;
- Origini dati SAP HANA (*High-Performance Analytical Appliance*).

Nel lavoro trattato i dati provengono esclusivamente da universi. Tramite questo *tool* è possibile modellare i report e creare delle strutture in cui sono visualizzati i dati. Inizialmente si procede a caricare un report da visualizzare e successivamente si ricerca il lotto, per *batch code* oppure per *material code*, con cui si vuole popolare tale report. L'interfaccia permette di visualizzare i report in modalità Lettura, in cui si può solo visualizzare il report, oppure in modalità Progettazione, che permette di modificare i dati visualizzati nel report nella loro struttura oppure nella loro logica inserendo le opportune istruzioni; tale visualizzazione può essere *real time,* con i dati del lotto scelto, oppure semplicemente con la struttura grezza del report senza nessun dato visualizzato ma solo con le istruzioni che rappresentano la logica di popolamento delle varie sezioni del report.

Nella figura 30 vediamo un esempio di report aperto in SAP Web Intelligence. In alto a sinistra si notano le tre sezioni *Home*, *Documents* e *Global Batch Review*; la prima porta nella pagina principale dove sono visualizzati i report aperti di recente da quell'utenza; la seconda permette di navigare tra le cartelle e di cercare i report tra quelli presenti; la terza corrisponde al report aperto che si sta visualizzando. In modalità lettura, nella barra degli strumenti notiamo alcune funzionalità come *Track*, che permette di confrontare i dati presenti con dati che si rifanno allo stesso report ma con dati vecchi mettendo in rilievo le modifiche, *Drill*, che permette di fare operazioni di *drill down* sui dati, *Filter Bar*, che permette di filtrare i dati visualizzati, e *Outline* che permette di visualizzare o meno la

struttura del report. Sulla sinistra si nota poi una barra di navigazione che ci permette di visualizzare rapidamente i diversi tab che compongono il report. Infine in basso, sulla barra di stato possiamo scorrere le pagine del tab corrente, modificare la modalità di visualizzazione e vedere quanto da tempo è stato caricato il report.

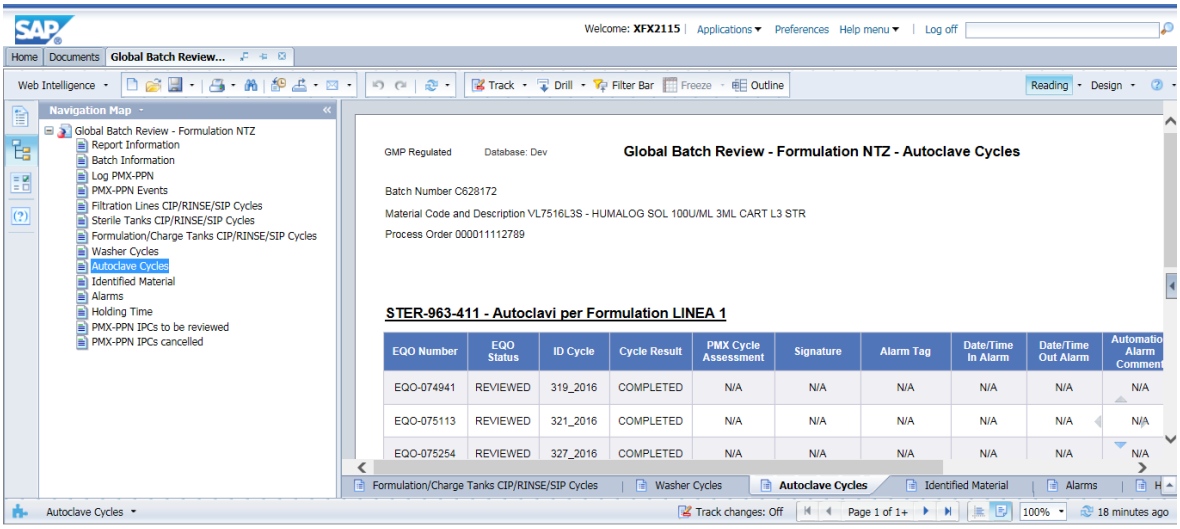


Figura 30: SAP BO Web Intelligence in modalità Lettura

6 ESTRAZIONE, TRASFORMAZIONE E CARICAMENTO

In questo capitolo viene descritto il processo di estrazione, trasformazione e caricamento (ETL) dei dati al fine di popolare il *data warehouse* per la successiva generazione di report che soddisfino le analisi precedentemente definite. Si presentano delle nozioni base dell'ETL, le procedure PL/SQL usate per l'implementazione, lo sviluppo del processo di ETL, e infine i *workflow* coinvolti nel progetto con la metodologia usata per la loro schedulazione. Nello specifico è trattata la *stored procedure* per il caricamento dell'area di *staging* a partire dai sistemi sorgente. Il caricamento del *core* avviene tramite più *workflow* che permettono di popolare le tabelle *target* nel *data warehouse* a partire dalle numerose tabelle sorgenti presenti sullo *staging*.

6.1 Processo di ETL

Il processo di ETL (*Extract-Transform-Load*) è il fondamento su cui si poggiano i sistemi di *data warehouse*: consiste nell'estrazione, trasformazione e caricamento dei dati sorgenti in una base di dati di destinazione, tipicamente un *data warehouse*. Tale processo è diviso nelle seguenti tre fasi:

- Estrazione dei dati da sistemi sorgente eterogenei;
- Pulizia e trasformazione dei dati;
- Caricamento dei dati trasformati in un *database target* (tipicamente un *data warehouse*).

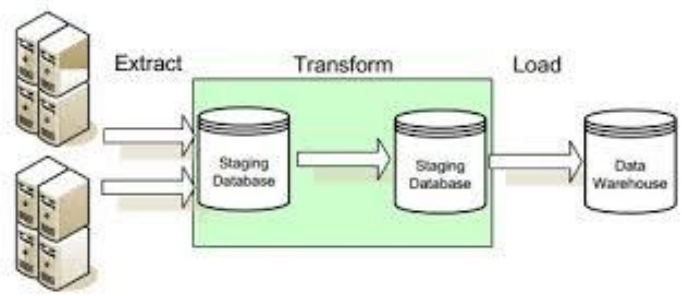


Figura 31: Processo di ETL

La progettazione e la realizzazione delle procedure di ETL sono attività di *back-end*, ovvero poco visibili agli utilizzatori finali. Solitamente le operazioni di pulizia includono una

selezione dei dati, mantenendo solo quelli ritenuti interessanti per le analisi, e una normalizzazione dei dati, eliminando ad esempio i duplicati. Le attività inerenti l'ETL vengono spesso sottovalutate in fase di pianificazione e di allocazione delle risorse: in realtà, secondo Kimball l'ETL "consuma facilmente il 70% delle risorse necessarie all'implementazione e al mantenimento di un tipico *data warehouse*" [Kimball 04]. Solitamente, il processo di ETL prevede che l'attività di trasformazione dei dati avvenga a livello di *staging* area. Tuttavia, in questo lavoro si è scelto di effettuare le trasformazioni sui dati a livello di *data warehouse*, perché il sistema che ospita il *data warehouse* risulta essere più performante rispetto al sistema che contiene l'area di *staging*. In questo caso sarebbe quindi più corretto parlare di ELT anziché di ETL.

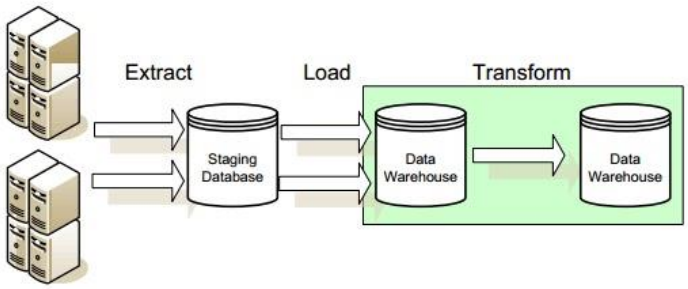


Figura 32: Processo di ELT

Nonstante in questo progetto non venga utilizzata per le operazioni di trasformazione, l'area di *staging* viene comunque mantenuta perché in questo modo si identifica il sottoinsieme di informazioni utili del *source system* isolandole prima del caricamento nell'area del *data warehouse*. Inoltre, si ottiene una separazione tra i sistemi sorgente e il *data warehouse* che evita di appesantire troppo il sistema sorgente (tipicamente costituito da *database* transazionali).

Per quanto riguarda il caricamento è utile definire le due modalità di caricamento di dati che sono adottate in questo progetto, che nello specifico sono:

- **Full:** i dati vengono caricati in toto, solitamente precedentemente vengono cancellati tutti i record presenti così da avere uno scenario che presenta la totalità dei dati allineati alla sorgente da cui originano. Si tratta di un'operazione molto dispendiosa quindi non è eseguita di frequente, soprattutto per tabelle con grande quantità di dati.

- **Delta:** i dati vengono caricati in maniera parziale rispetto a quelli presenti sulla sorgente. Solitamente viene considerato il *timestamp* che si riferisce all'ultimo caricamento avvenuto, in questo modo saranno caricati solo i dati inseriti o modificati più recentemente. Questa modalità è molto meno dispendiosa della precedente, e per questo motivo è eseguita molto più frequentemente.

6.2 Tipologie di ETL

L'elemento ETL viene usato per estrarre, caricare e trasformare i dati. Poiché vi sono diverse aree database (*staging*, *data warehouse* e *data mart*) vengono identificate di conseguenza differenti livelli di ETL:

- **ETL0:** sposta l'informazione dai source system alla *staging* area. In questa fase non viene effettuata nessuna operazione di aggregazione o di elaborazione. L'ETL estrae l'informazione dalle tabelle sorgenti e le copia in quelle di destinazione, che hanno la stessa struttura. Questa operazione viene compiuta allo scopo di velocizzare lo spostamento dei dati e per ridurre al minimo l'impatto sui sistemi sorgente;

- **ETL1:** sposta l'informazione dallo *staging* area al *data warehouse* area, applicando la logica di aggregazione, trasformazione e integrazione;

- **ETL2:** muove l'informazione dall'area di *data warehouse* a quella dei *data mart*. Questi ultimi vengono strutturati appositamente per i report. In questa fase l'ETL compie aggregazioni e calcoli ad alto livello.

6.3 Caricamento Dati dalle Sorgenti all'Area di Staging

Per implementare le procedure di ETL per le tabelle di *staging*, è stata utilizzata una *stored procedure* invocata tramite il *tool* Informatica PowerCenter. Il *mapping* corrispondente è m_call_stg che viene richiamato dalla *session* s_<nome_sistema_sorgente>_stg_load presente nei vari *workflow* corrispondenti ad ogni sistema sorgente. Il *mapping* e la *session* in realtà sono fittizi perché non si occupano in prima persona di caricare i dati sullo *staging* ma semplicemente si limitano a richiamare la *stored procedure*.

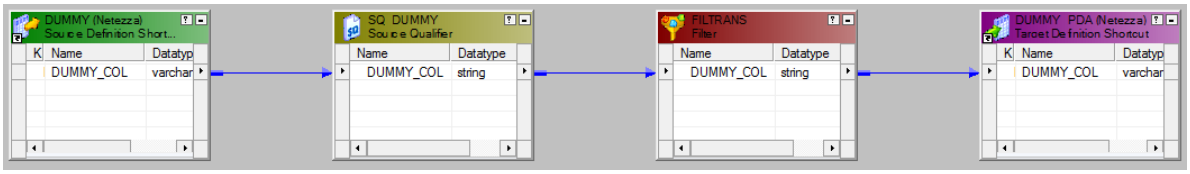


Figura 33: *Mapping* per il caricamento dall'area di *staging*

In questo ambito, la scelta di usare del codice PL/SQL per sviluppare le procedure di ETL anziché utilizzare puramente il *tool* di ETL dedicato, è motivata dal fatto che, sviluppata la *stored procedure* in maniera parametrica, questa può essere utilizzata per caricare i dati in tutte le tabelle di *staging*, qualsiasi sia la zona di appartenenza o il sito produttivo. Quindi non è necessario realizzare delle procedure di ETL dedicate per ogni tabella di *staging*. È da ricordare inoltre che, l'ETL usato per caricare i dati nello *staging* viene classificato come di tipo 0, ovvero vengono caricate le tabelle di staging prendendo i dati dalle sorgenti senza effettuare nessun tipo di trasformazione sui dati stessi. La *stored procedure* carica i campi delle tabelle di *staging* usando una tabella di configurazione, TBSTCNF_LOAD_STG. In questa tabella vengono definiti i parametri necessari per il caricamento delle tabelle di *staging*. Esiste una *stored procedure* per ogni sorgente, nello specifico abbiamo:

- SP_STAGING_PPN;
- SP_STAGING_PA;
- SP_STAGING_GRIP.

Nella tabella TBSTCNF_LOAD_STG oltre alle informazioni usate per il caricamento dati, vengono specificate anche tutte le tabelle che devono essere caricate in quell'istanza di *staging*. Di seguito vengono descritti gli attributi di questa tabella che vengono utilizzati dalla *stored procedure* per caricare le tabelle di *staging*:

- SRC_SYSTEM_CD: identifica la sorgente dalla quale si vanno a reperire i dati;
- SRC_TABLE_NAME_CD: è il nome della tabella nel sistema sorgente dalla quale bisogna estrarre i dati;
- SRC_FACILITY_CD: è il codice che identifica il sito produttivo;
- SRC_INST_CD: è il codice di installazione sul sito produttivo;
- SRC_SCHEMA: è lo schema utilizzato nel sistema sorgente;
- SRC_FILTER_DESC: per specificare eventuali filtri da utilizzare sulla tabella sorgente;
- FLUID_QUERY_ALIAS: è il nome della *fluid query*;
- FULL_LOAD_FLG: è un campo *flag* che, se settato a 'Y', fa si che anche la tabella di *staging* venga caricata in modalità *full*, se settato a 'N' invece è caricata in modalità *delta*.
- STG_SCHEMA_2_LEV: è lo schema da utilizzare per le tabelle di *staging*;
- STG_TABLE_NAME_2_LEV: è il nome della tabella da utilizzare nello *staging*;

- PRIMARY_KEY_DESC: chiavi primarie della tabella;

- LAST_UPDATE_DT: indica la data di ultimo aggiornamento del record sulla tabella di configurazione;

- FULL_SRC_FILTER_DESC: filtri da applicare sulla tabella sorgente quando il giorno della settimana corrisponde a quello settato;

- DAYOFWEEK: giorno della settimana in cui viene effettuato il controllo sui record cancellati.

I campi SRC_SYSTEM_CD , SRC_TABLE_NAME, SRC_FACILITY_CD, SRC_INST_CD fanno da chiave primaria per la tabella in oggetto. Per ogni tabella *target* da caricare viene creata una *session* in Informatica PowerCenter Workflow Manager. Tutte le *session* create fanno riferimento allo stesso *mapping*, m_call_stg, ma con valori differenti per i campi chiave SRC_SYSTEM_CD, SRC_TABLE_NAME, SRC_FACILITY_CD, SRC_INST_CD.

Il processo di caricamento avviene mediante i seguenti passi:

1. Per ogni tabella di *staging* da caricare viene definita una *session* con i campi chiave della tabella di configurazione TBSTCNF_LOAD_STG valorizzati in base alla tabella sorgente dalla quale bisogna prendere i dati;

2. Le session richiamano il *mapping* m_call_stg che altro non fa che invocare la *stored procedure* con i parametri passati dalla *session*;

3. La *stored procedure* carica le tabelle di *staging* e valorizza una tabella di log, TBSTG_LOAD_STG, nella quale viene tenuta traccia delle operazioni compiute.

6.4 Stored Procedure per Caricare lo Staging

Di seguito vengono presentate le *stored procedure* che hanno il compito di caricare le tabelle di *staging*, andando a leggere i parametri di configurazione dalla tabella TBSTCNF_LOAD_STG, e di implementare il meccanismo di *delete*. Tutte le procedure usano questi parametri:

- IN_SRC_SYSTEM_CD VARCHAR, → codice corrispondente al sistema sorgente;

- IN_SRC_TABLE_NAME_CD VARCHAR, → codice corrispondente al nome della tabella;

- IN_SRC_FACILITY_CD VARCHAR, → codice corrispondente alla *facility*;

- IN_SRC_INST_CD VARCHAR, → *source instance code*, codice di installazione sul sito produttivo;

- IN_INFORMATICA_CD BIGINT, → codice identificativo del *run* a cui corrisponde il caricamento;

- IN_TIMESTAMP_RIF TIMESTAMP → *timestamp* della creazione/aggiornamento del record.

I primi 4 parametri sono usati per identificare un record nella TBSTCNF_LOAD_STG. Queste informazioni sono usate per settare le variabili interne alle procedure.

6.4.1 Caricamento dello Staging

Per effettuare le operazioni di caricamento su una specifica tabella, la *stored procedure* utilizza i parametri ricevuti dalla *session* per andare in *join* con la tabella di configurazione TBSTCNF_LOAD_STG per poter inizializzare i parametri necessari al caricamento di quella determinata tabella.

```
SELECT
SRC_FILTER_DESC,SRC_SCHEMA,FLUID_QUERY_ALIAS,STG_SCHEMA_2_LEV,STG_TABLE_NAME_2_LE
V,PRIMARY_KEY_DESC,FULL_LOAD_FLG,FULL_SRC_FILTER_DESC,DAYOFWEEK
INTO
V_SRC_FILTER_DESC,V_SRC_SCHEMA,V_FLUID_QUERY_ALIAS,V_STG_SCHEMA_2_LEV,V_STG_TABLE
_NAME_2_LEV,V_PRIMARY_KEY_DESC,V_FULL_LOAD_FLG,
V_FULL_SRC_FILTER_DESC,V_DAYOFWEEK
FROM   GMDM_STG_L2.TBSTCNF_LOAD_STG
WHERE  SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND
       SRC_TABLE_NAME_CD = IN_SRC_TABLE_NAME_CD  AND
       SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND
       SRC_INST_CD = IN_SRC_INST_CD;
```

Successivamente, la procedura recupera da TBSTLOG_LOAD_STG il *timestamp* che corrisponde all'ultimo caricamento effettuato con successo, quindi con status COMPLETED.

```
SELECT CASE WHEN MAX(END_TMSTP) IS NULL THEN TO_timestamp('01/01/1900
00:00:00','DD/MM/YYYY hh24:mi:ss') ELSE MAX(END_TMSTP) END AS LAST_LOAD
       INTO V_LAST_LOAD
       FROM GMDM_STG_L2.TBSTLOG_LOAD_STG
       WHERE SRC_SYSTEM_CD=IN_SRC_SYSTEM_CD AND
       SRC_TABLE_NAME_CD=IN_SRC_TABLE_NAME_CD AND
       SRC_FACILITY_CD=IN_SRC_FACILITY_CD AND
       SRC_INST_CD=IN_SRC_INST_CD AND
       LOAD_STATUS_CD='COMPLETED';
```

Nella tabella di log TBSTGLOG_LOAD_STG viene tenuta traccia del caricamento fatto nello *staging*, inserendo un nuovo record per ogni tabella interessata dal caricamento. Lo stato in cui si trova il caricamento viene inizialmente impostato a 'RUNNING'.

```
INSERT INTO GMDM_STG_L2.TBSTLOG_LOAD_STG
       (START_TMSTP,END_TMSTP,INFORMATICA_CD,SRC_SYSTEM_CD,SRC_TABLE_NAME_CD,SRC_
       FACILITY_CD,SRC_INST_CD,LOAD_STATUS_CD,LOAD_ERR_CD,STG_L2_ROWS_INS_NBR,STG
       _L2_ROWS_UPD_NBR,STG_L2_ROWS_DEL_NBR,LOAD_START_TMSTMP )
       VALUES
       (V_LAST_LOAD,NULL,IN_INFORMATICA_CD,IN_SRC_SYSTEM_CD,IN_SRC_TABLE_NAME_CD,
       IN_SRC_FACILITY_CD,IN_SRC_INST_CD,'RUNNING',NULL,NULL,NULL,NULL,NOW());
COMMIT;
```

Le procedure usano le *fluid query*, che sono una peculiarità di Netezza e sono query usate per accedere a varie sorgenti di dati. Se il database sorgente è di Oracle, le procedure recuperano i nomi delle colonne dalle tabelle sorgenti corrispondenti a quelle dello *staging*. Le procedure di seguito provvedono a creare una tabella temporanea con le colonne recuperate in precedenza e lo schema dell'area di *staging*. Questa tabella si popola dei record recuperati dalla sorgente tramite la *fluid query*. I dati possono essere filtrati temporalmente a seconda del valore di V_SRC_FILTER_DESC presente sulla TBSTCNF_LOAD_STG.

```
SQL := ' select * from table WITH final
       ('||V_FLUID_QUERY_ALIAS||'('''',''''',''select column_name, data_type
                 from ALL_TAB_COLUMNS
                       where TABLE_NAME='''''||IN_SRC_TABLE_NAME_CD||'''''
                             AND OWNER='''''||V_SRC_SCHEMA||'''''''))';
FOR cur IN EXECUTE SQL LOOP
```

Se il campo FULL_LOAD_FLG è settato 'N', e ogni record della tabella temporanea che è chiave primaria non è presente nella tabella dello *staging* corrispondente allora viene inserito. In seguito la procedura crea una tabella temporanea di *update* in cui ci sono i record che sono stati ricavati dalla *minus* tra la tabella temporanea iniziale e la tabella di *staging* per poi procedere con la seguente operazione nella tabella di *staging*.

Setta il campo GMDM_STG_RCRD_STS_CD = 'S', per tutti quei record che sono presenti nella tabella temporanea di *update*; Inserisce poi tutti i record presenti nella tabella temporanea di *update* con RCRD_STS_CD = 'A' nella tabella dello *staging*.

Il controllo sui record cancellati avviene settimanalmente in base al giorno settato sul campo DAYOFWEEK nella tabella TBSTCNF_LOAD_STG che specifica appunto il giorno della settimana in formato numerico.

Se invece Se il campo FULL_LOAD_FLG è settato 'Y' il caricamento è in *full*. Nel caso in cui il caricamento dei dati sia in modalità *full* bisogna cancellare i record presenti prima di poterne inserire di nuovi.

6.4.2 Popolamento della Tabella di Log TBSTGLOG_LOAD_STG

Al termine delle operazioni di caricamento dello *staging*, la *stored procedure* carica i dati inerenti le informazioni sulle operazioni di caricamento nella tabella di log TBSTGLOG_LOAD_STG. Ad ogni lancio della *stored procedure* viene inserito un nuovo record nella tabella di log, impostando lo stato di quel record a 'RUNNING'. Al termine delle operazioni di caricamento, lo stato del record passa a 'COMPLETED'. Le altre informazioni che vengono salvate in questa tabella indicano la data in cui il caricamento ha avuto termine eil numero di record inseriti, aggiornati ed eliminati per la tabella di *staging*. La *stored procedure* va ad aggiornare i campi della tabella di log nel record precedentemente inserito in fase di inizializzazione delle variabili.

```
UPDATE GMDM_STG_L2.TBSTLOG_LOAD_STG
       SET
             LOAD_STATUS_CD = 'COMPLETED',
              END_TMSTP = IN_TIMESTAMP_RIF,
              STG_ROWS_NBR = V_STG_ROWS_NBR,
             STG_L2_ROWS_INS_NBR = v_STG_ROWS_INS_NBR,
             STG_L2_ROWS_UPD_NBR = v_STG_ROWS_UPD_NBR,
             STG_L2_ROWS_DEL_NBR = v_STG_ROWS_DEL_NBR,
              LOAD_END_TMSTMP = NOW()
              WHERE SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND
              SRC_TABLE_NAME_CD = IN_SRC_TABLE_NAME_CD  AND
              SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND
              SRC_INST_CD = IN_SRC_INST_CD AND
              INFORMATICA_CD=IN_INFORMATICA_CD;
       COMMIT;
```

6.5 Caricamento del Core e del Data Mart

A differenza del caricamento dell'area di *staging*, le singole tabelle che costituiscono il *data warehouse* vengono popolate da singoli flussi ETL progettati attraverso il *tool* Informatica PowerCenter.

A differenza del caricamento dati da sistema sorgente a *staging* area, quando i dati vengono caricati dal livello dei dati intermedio al *data warehouse* subiscono delle trasformazioni, che non è possibile standardizzare per tutte le tabelle che vengono coinvolte. Esistono infatti numerosi *mapping*, che poi sono invocati nei *workflow* coinvolti nel progetto, modellati secondo le esigenze richieste del cliente e delle tabelle dell'area *core* coinvolte. Si ricorda che si è scelto di implementare un sistema ELT e non ETL in quanto le trasformazioni a livello del *data warehouse* sono risultate più performanti. Allo scopo di rispettare i vincoli di integrità fra le tabelle, vengono caricate prima tutte le tabelle dimensionali, e solo se il caricamento è avvenuto senza errori, la tabella del fatto.

6.6 Workflow

Tramite dei *workflow* vengono caricati i dati, che risiedono sui sistemi sorgente, sul GMDM Core e poi successivamente sul *data mart* attraverso vari steps. Inizialmente i dati sono caricati nell'area di *staging* tramite delle procedure in PL/SQL. Queste procedure sono richiamate in ogni *workflow* tramite un *mapping* fittizio. Di seguito ogni *workflow* ha varie *session* che permettono di popolare le varie tabelle del *core* a partire da tabelle sorgenti corrpisondenti alle tabelle sullo *staging*. I dati sono caricati con la logica *delta*, e quindi rifacendosi alla data dell'ultimo caricamento, più volte durante l'arco della settimana a seconda delle schedulazioni del flusso. Inoltre una volta a settimana si ha un caricamento in *full* a partire dal sistema sorgente. Oltre ai *workflow* che caricano i dati sul *core* viene utilizzato un altro *workflow* che serve a caricare i dati sull'*extension*. I *workflow* sono unici e sono usati per tutti i siti configurati grazie alla presenza di differenti *file* dei parametri (nell'appendice se ne può vedere un esempio). Nello specifico i *workflow* coinvolti nel progetto di *Global Batch Review* sono:

- wf_ppn_gmdm;
- wf_pa_audit_gmdm;
- wf_grip_gmdm;
- wf_ext_gbr_gmdm (*extension*).

I primi tre *workflow* si riferiscono al caricamento dei dati a partire dai sistemi sorgente fino ad arrivare a popolare il *core*, l'ultimo invece si riferisce al caricamento del *data mart*.

6.7 Schedulazione dei Workflow

La schedulazione definita con il Workflow Manager di Informatica PowerCenter, ha il difetto di disabilitarsi automaticamente qualora un flusso si arresti per errore. Per questo motivo, la società ha deciso di ricorrere al *crontab* di unix.

I flussi saranno pianificati per girare una volta al giorno, inoltre come facilmente ci si aspetta il flusso dell'*extension* girerà dopo l'esecuzione del *core*. Questa scelta è motivata dal fatto che essendo il progetto nato per monitorare quotidianamente ciò che accade durante un processo produttivo, i dati devono essere quanto più possibile aggiornati.

La schedulazione dei flussi in *crontab* è partizionata prima per flusso e poi per sito produttivo. Inizialmente sono indicati i minuti, poi l'ora di esecuzione, che determinano il numero di esecuzioni giornaliere, e infine i giorni, indicati progressivamente con i numeri da 0 a 6 a partire dalla domenica. Poi ci sono una serie di parametri che indicano i comandi per eseguire il flusso e il *path* in cui salvare il log, con le informazioni inerenti l'esecuzione.

Nell'esempio riportato in figura si ha il flusso che si riferisce a PMX, wf_ppn_gmdm, per i tre siti coinvolti. Si può notare come ogni sito presenti due entrate mirate a differenziare la frequenza delle esecuzioni durante la settimana e durante il fine settimana.

```
# begin wf_ppn_gmdm

# FEG
00 03,11,19 * * 0,1,2,3,4,6 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd sta
LDER} -rin FEG -nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log

00 09 * * 5 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd startworkflow -sv $
nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log 2>&1'

# SES
00 07,22 * * 1,2,3,4 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd startworkf
rin SES -nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log 2>&1'

00 22 * * 0,5,6 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd startworkflow -
ES -nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log 2>&1'

# INDY
00 01,13 * * 1,2,3,4 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd startworkf
rin INDY -nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log 2>&1'

00 13 * * 0,5,6 export TERM=""; ksh -c '. ~/.kshrc ; pmcmd startworkflow -
NDY -nowait wf_ppn_gmdm >> $GMDM_logdir/scripts/wf_ppn_gmdm.log 2>&1'
```

Figura 34: Crontab di unix per flussi di PMX

7 REPORTISTICA

L'obiettivo del progetto *Global Batch Review* presentato in questo lavoro di tesi, è quello di fornire una reportistica che sia di supporto agli utenti finali in merito alla decisione di rilasciare sul mercato un determinato lotto di insulina al termine del suo processo produttivo. In questo capitolo si descrive in grandi linee la reportistica e l'universo, che costituisce la sorgente dati per il *tool* di *front-end* SAP Business Object Web Intelligence. In fine l'attenzione verte su alcuni esempi di report prodotti.

Il sottoscritto non è stato impegnato in prima persona nella realizzazione di tali report e quindi non sarà presente una descrizione approfondita dei metodi e degli strumenti usati. Quest'ultima fase del progetto, che porta alla realizzazione della soluzione, ha previsto una stretta collaborazione con i membri del team di lavoro addetti a questa fase e ad assecondare le loro richieste per delle modifiche a monte, a livello del *data warehouse* e del *data mart*, affinchè le informazioni presenti sui report potessero essere veritiere e conformi alle richieste del committente.

7.1 Reportistica

La fase di *reporting* è fondamentale per rendere comprensibile la lettura dei dati e i risultati delle analisi. Le informazioni fornite devono essere molto chiare e presentate in maniera intuitiva in quanto sono fondamentali per dare un supporto alle decisioni che dovrà prendere l'utente circa la bontà del prodotto, fattore ancora più determinante in ambito farmaceutico.

Tutti gli strumenti di reportistica consentono sia di formulare un'analisi dei dati senza che sia necessario scrivere un'interrogazione in SQL, sia di produrre una rappresentazione grafica dei risultati. Esistono diversi modi per visualizzare i risultati come ad esempio:

- Report tradizionale: dove i dati vengono organizzati per colonne, intestazioni e uno o più livelli di riepilogo parziali;

- Tabella a doppia entrata: in questo caso le informazioni vengono mostrate aggregando le misure rispetto ai valori delle dimensioni di analisi lungo gli assi cartesiani;

- Grafici: i dati vengono rappresentati, ad esempio, sotto forma di istogrammi o diagrammi lineari.

Come detto in precedenza, per sviluppare la reportistica è stato utilizzato lo strumento SAP Business Object Web Intelligence (WEBI).

7.2 Progettazione dell'Universo

L'universo è uno strato semantico che viene interposto tra il *data warehouse* e il pannello delle *query* per rendere più semplice la creazione di report e analisi. Ciò consente all'utente finale di utilizzare la terminologia tipica del business aziendale per accedere ai dati presenti nel *data warehouse* senza l'utilizzo di linguaggi e terminologie tecniche. La finalità dell'universo è quella di creare un collegamento tra il *data warehouse* e un'interfaccia semplice da utilizzare per gli utenti che, senza una preparazione tecnica informatica, vogliono effettuare ricerche all'interno di un database, creare report oppure analizzare dei dati.

La prima parte della fase di implementazione si traduce nella creazione di uno schema ovvero una rappresentazione grafica della struttura del *database*. Uno schema è composto da tabelle e *join*. All'interno di un universo viene definita la cardinalità tra le tabelle, specificando quanti elementi di una tabella hanno una corrispondenza con gli elementi dell'altra. Le colonne delle tabelle rappresentano gli oggetti con cui l'utente finale crea i report. Poiché nello schema devono essere inserite le tabelle le cui colonne corrispondono agli oggetti che gli utenti finali utilizzano per fare i report, è assolutamente necessario conoscere le esigenze e le richieste degli utenti finali, i quali devono contribuire con un *feedback* continuo durante lo sviluppo dell'universo.

Per il progetto di Global Batch Review sono stati utilizzati 3 universi, e quello usato come riferimento per le analisi svolte, chiamato *DIA - Facts - Global Batch Review_NTZ*, è comprensivo di 39 Classi, 385 Oggetti, 35 Tabelle e 47 Join.

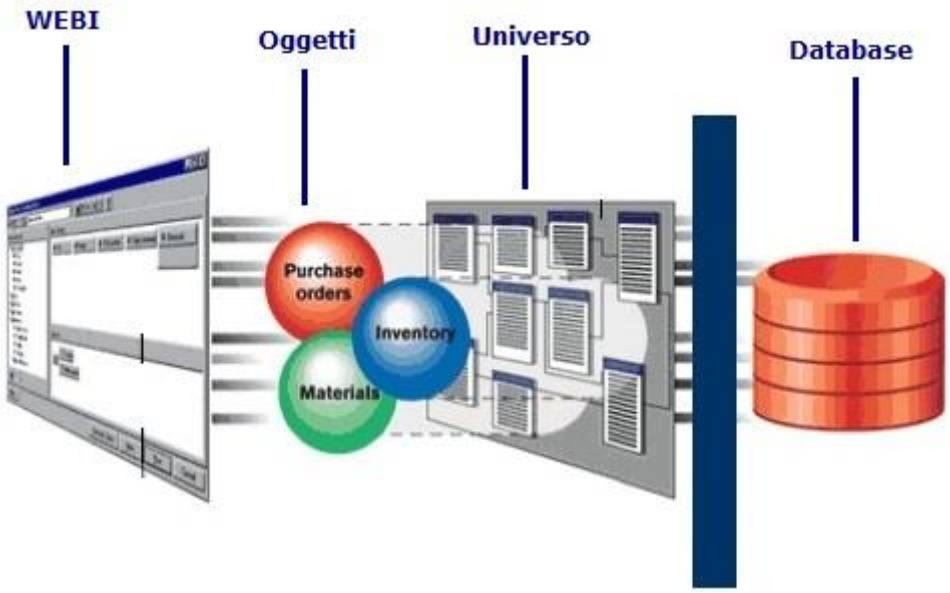
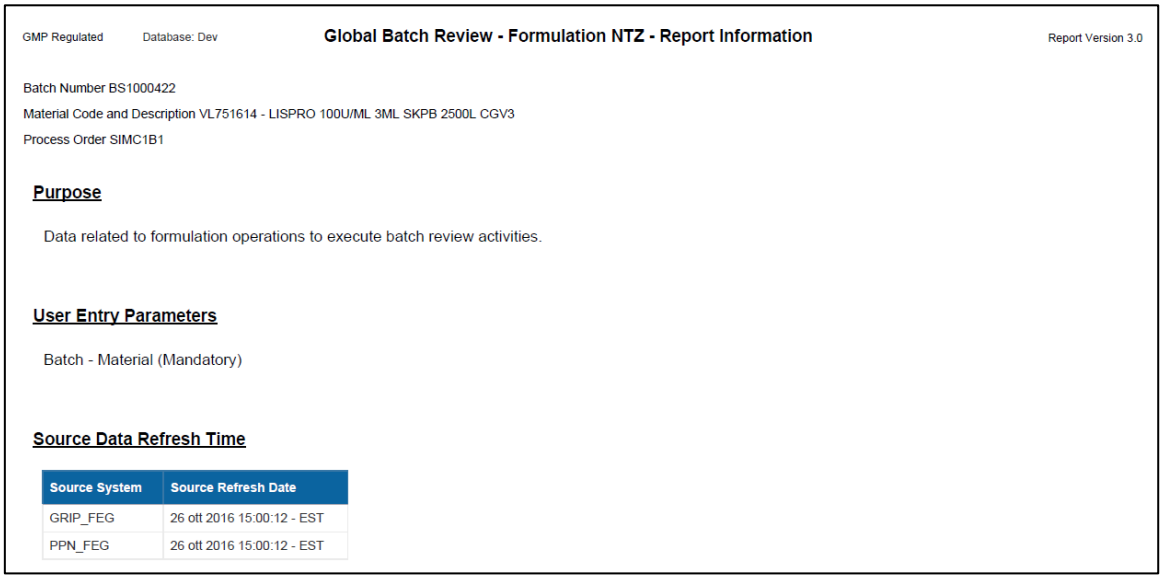


Figura 35: Diagramma SAP BO

7.3 Report Realizzati

A titolo di esempio, vengono mostrati i tab di alcuni report realizzati mediante WEBI. Prima di poter usufruire dei dati mostrati dai report, l'utilizzatore deve inserire il codice identificativo del lotto di insulina sul quale si desidera fare delle analisi. Un report è formato da vari tab, ognuno con informazioni specifiche a seconda della tipologia di report in esame. Nello specifico in questo paragrafo verrà analizzato un report inerente la fase di *Formulation*, ossia la fase iniziale nella produzione dei flaconi di insulina, per un *batch* prodotto nel sito francese.


GMP Regulated Database: Dev **Global Batch Review - Formulation NTZ - Report Information** Report Version 3.0

Batch Number BS1000422
Material Code and Description VL751614 - LISPRO 100U/ML 3ML SKPB 2500L CGV3
Process Order SIMC1B1

Purpose

Data related to formulation operations to execute batch review activities.

User Entry Parameters

Batch - Material (Mandatory)

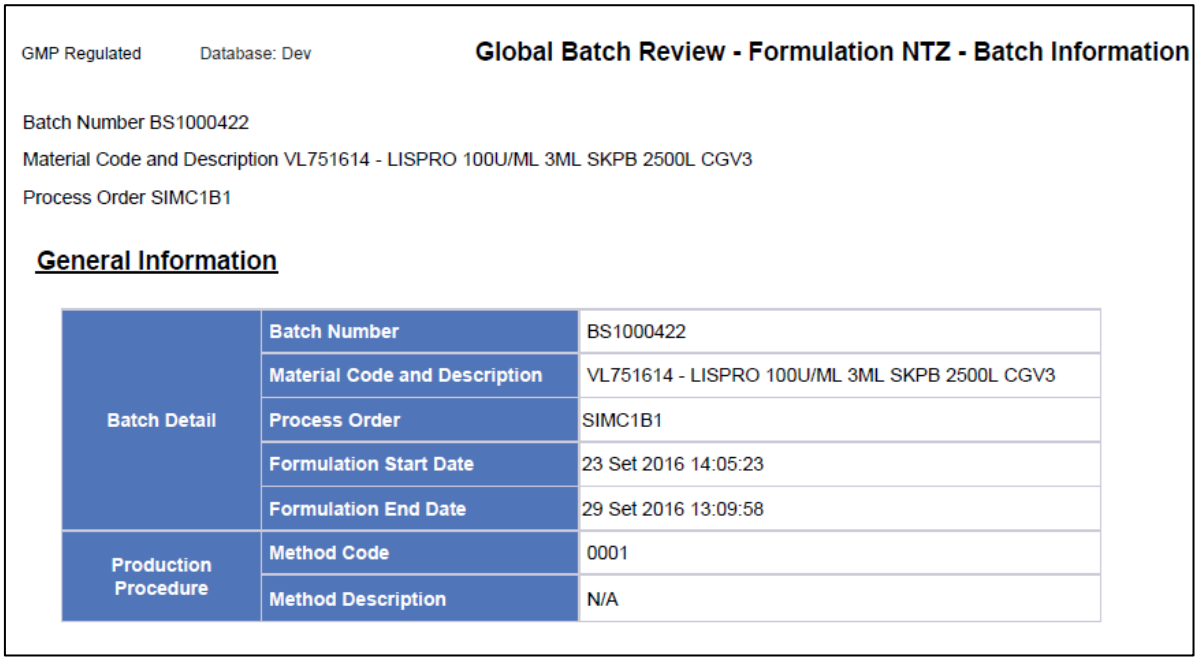
Source Data Refresh Time

Source System	Source Refresh Date
GRIP_FEG	26 ott 2016 15:00:12 - EST
PPN_FEG	26 ott 2016 15:00:12 - EST

Figura 36: Tab iniziale (Report Information) del report di Formulation

Il primo tab evidenzia le informazioni generali che riguardano il report e il *batch* in esame. Nell'intestazione si può notare che si tratta di un report della *Global Batch Review versione 3.0*. Dall'intestazione è anche possibile notare anche in quale tab ci troviamo. Di seguito sono riportati l'identificativo del *batch* di riferimento, il codice del materiale con la descrizione e il *process order* associato. Tra le sorgenti di dati da cui sono state ricavate le informazioni, rappresentate in maniera tabulare, notiamo come i dati siano originati dai sistemi sorgente GRIP e PMX, in quanto il sito francese non ha PA tra i suoi sistemi sorgente, e la data in cui i dati sono stati aggiornati. Non visibile nell'immagine, a piè di pagina invece ci sono informazioni che ci dicono la data in cui il report è stato fatto girare, l'utente che l'ha fatto girare, la sezioni in cui ci troviamo e quando ha girato per l'ultima volta il *workflow* a cui compete aggiornare i dati presente nell'*extension* su cui si basa questo report. Molte di queste informazioni appena descritte saranno presenti in maniera identica anche in tutte le altre pagine del report.

Il tab di *Batch Information*, come suggerisce il nome, ci da delle informazioni dettagliate sul *batch*. Oltre a quelle già precedentemente evidenziate dal tab precedente abbiamo anche la data di inizio e fine della fase del processo produttivo a cui si riferisce il report, oltre alle informazioni sulla procedura di produzione. In forma tabulare sono anche presenti le informazioni specifiche di ogni step all'interno della fase produttiva.



GMP Regulated Database: Dev **Global Batch Review - Formulation NTZ - Batch Information**

Batch Number BS1000422

Material Code and Description VL751614 - LISPRO 100U/ML 3ML SKPB 2500L CGV3

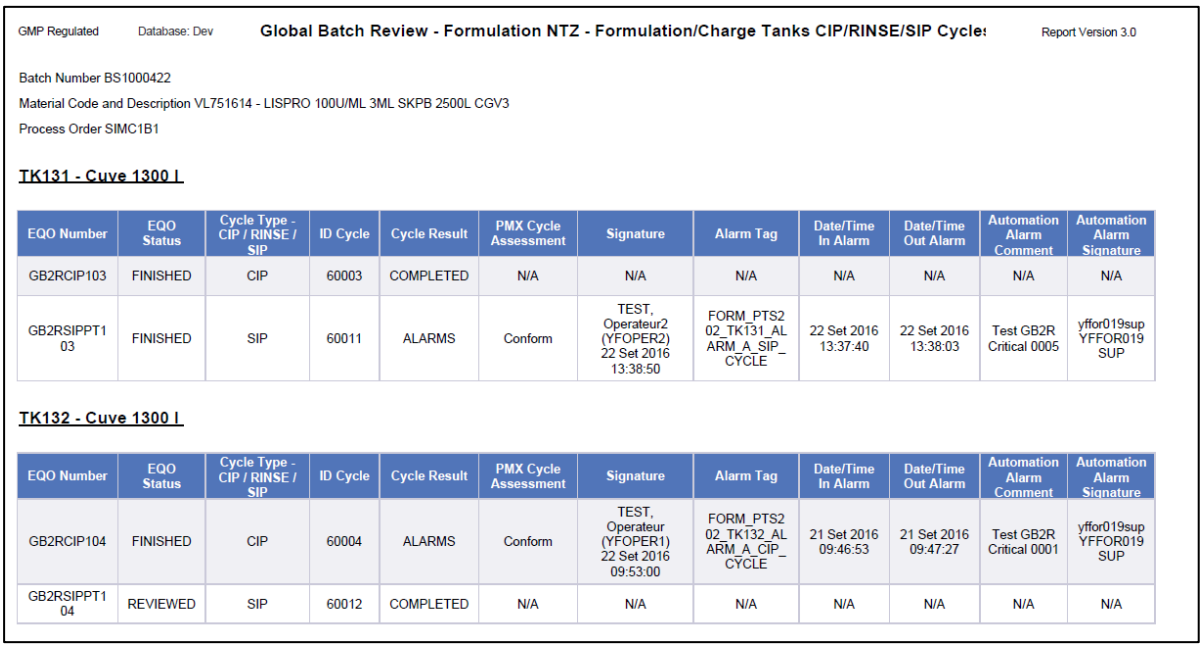
Process Order SIMC1B1

General Information

Batch Detail	**Batch Number**	BS1000422
	Material Code and Description	VL751614 - LISPRO 100U/ML 3ML SKPB 2500L CGV3
	Process Order	SIMC1B1
	Formulation Start Date	23 Set 2016 14:05:23
	Formulation End Date	29 Set 2016 13:09:58
Production Procedure	**Method Code**	0001
	Method Description	N/A

Figura 37: Tab di Batch Information

Sul report di Formulation sono presenti vari tab dedicati ai cicli. Il tab *Filtration Lines CIP/RINSE/SIP Cycles*, il tab *Sterile Tanks CIP/RINSE/SIP Cycles*, il tab *Charge Tanks CIP/RINSE/SIP Cycles*, il tab *Washer Cycles* e il tab *Autoclave Cycles*.



GMP Regulated Database: Dev **Global Batch Review - Formulation NTZ - Formulation/Charge Tanks CIP/RINSE/SIP Cycles** Report Version 3.0

Batch Number BS1000422

Material Code and Description VL751614 - LISPRO 100U/ML 3ML SKPB 2500L CGV3

Process Order SIMC1B1

TK131 - Cuve 1300 l

EQO Number	EQO Status	Cycle Type - CIP / RINSE / SIP	ID Cycle	Cycle Result	PMX Cycle Assessment	Signature	Alarm Tag	Date/Time In Alarm	Date/Time Out Alarm	Automation Alarm Comment	Automation Alarm Signature
GB2RCIP103	FINISHED	CIP	60003	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GB2RSIPPT103	FINISHED	SIP	60011	ALARMS	Conform	TEST, Operateur2 (YFOPER2) 22 Set 2016 13:38:50	FORM_PTS202_TK131_ALARM_A_SIP_CYCLE	22 Set 2016 13:37:40	22 Set 2016 13:38:03	Test GB2R Critical 0005	yffor019sup YFFOR019 SUP

TK132 - Cuve 1300 l

EQO Number	EQO Status	Cycle Type - CIP / RINSE / SIP	ID Cycle	Cycle Result	PMX Cycle Assessment	Signature	Alarm Tag	Date/Time In Alarm	Date/Time Out Alarm	Automation Alarm Comment	Automation Alarm Signature
GB2RCIP104	FINISHED	CIP	60004	ALARMS	Conform	TEST, Operateur (YFOPER1) 22 Set 2016 09:53:00	FORM_PTS202_TK132_ALARM_A_CIP_CYCLE	21 Set 2016 09:46:53	21 Set 2016 09:47:27	Test GB2R Critical 0001	yffor019sup YFFOR019 SUP
GB2RSIPPT104	REVIEWED	SIP	60012	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figura 38: Tab sui Cicli CIP/SIP/RINSE

Tutti questi tab presentano la medesima struttura ma chiaramente si riferiscono a macchinari differenti. Si ha una rappresentazione in forma tabulare con intestazione per ogni singolo macchinario coinvolto. Nell'intestazione è presente l'*Equipment Id Code* e l'*Equipment Description* che lo identificano univocamente. Poi in tabella sono presenti i dati inerenti l'EQO, il numero e lo status, il tipo (ove presente), l'identificativo e l'esito del ciclo, e infine tutta una serie di informazioni che sono presenti solo nel caso in cui il ciclo sia concluso con allarme, e cioè i dati di chi ha ravvisato l'allarme, l'identificativo dell'allarme, data di inizio e fine dei commenti.

GMP Regulated Database: Dev **Global Batch Review - Formulation NTZ - Autoclave Cycles**

Batch Number BS1000422

Material Code and Description VL751614 - LISPRO 100U/ML 3ML SKPB 2500L CGV3

Process Order SIMC1B1

AUT017-01 - Autoclave - This Class is specific for the Isolator Material Preparation

EQO Number	EQO Status	ID Cycle	Cycle Result	PMX Cycle Assessment	Signature	Alarm Tag	Date/Time In Alarm	Date/Time Out Alarm	Automation Alarm Comment	Automation Alarm Signature
GB2R AUT101	FINISHED	60022	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GB2R AUT102	REVIEWED	60023	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A
WRAU007	REVIEWED	41424	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A
WRAU010	REVIEWED	41692	COMPLETED	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figura 38: Tab sui Cicli Autoclave

8 CONCLUSIONI

Il lavoro presentato è stato svolto presso l'azienda di consulenza SDG Group e si è occupato della realizzazione di un sistema di *Business Intelligence* che ha previsto la progettazione e l'implementazione di un *data warehouse* capace di raccogliere e integrare le informazioni generate dal processo produttivo di insulina di una multinazionale farmaceutica di origine americana.

La soluzione finale ha permesso agli utenti a cui era destinata di avere accesso a una serie di report mirati a certificare la bontà del farmaco prodotto.

Durante lo sviluppo di questo progetto sono entrato a stretto contatto, oltre che con gli strumenti informatici, anche con le attività produttive dell'azienda committente e in particolare ho avuto l'opportunità di vedere da vicino le fasi che costituiscono il processo produttivo dell'insulina. Il principale centro di interesse per le analisi sono state le attività che vengono svolte durante la produzione di un lotto di insulina, e la conformità che queste attività hanno nei confronti delle disposizioni normative e ai codici di condotta in vigore nel contesto della produzione di farmaci.

Gli sforzi maggiori sono stati sostenuti per definire una corretta rappresentazione e integrazione dei numerosi dati provenienti dalle diverse fonti a disposizione e di standardizzare i processi di integrazione in modo da poter inserire in futuro altri siti produttivi in altre parti del mondo senza operare grandi cambiamenti strutturali. Questo è stato ottenuto soprattutto grazie ad un accurata definizione e strutturazione della fase di ETL, migliorando la *release* precedente del progetto e ponendo delle solide basi per la futura *release* prevista per il prossimo anno. Tali sviluppi prevedranno inoltre modifiche sui report fin qui prodotti integrando i dati provenienti da analisi quantitative oltre che qualitative.

Il progetto è stato realizzato in un team di consulenti che mi ha visto lavorare principalmente sulla fase di *back-end*, ma soprattutto il progetto è stato realizzato costantemente a contatto con il cliente e questo ha permesso una maggiore interattività e definizione delle attività di sviluppo.

La soluzione finale ha complessivamente soddisfatto il cliente e le sue aspettative. Per questo motivo i successivi sviluppi saranno affidati ancora una volta ad SDG Group.

Personalmente l'esperienza nel complesso è stata positiva e interessante, mi ha permesso infatti di utilizzare molti tra i concetti affrontati nel corso dei miei studi e in particolare la modellazione di un *data warehouse* e la realizzazione di procedure ETL. Inoltre ho avuto la possibilità di apprendere l'utilizzo di nuovi strumenti quali Informatica Power Center e Aginity Workbench, approfondire le conoscenze sul DBMS Oracle, del PL/SQL e di conoscere Netezza IBM.

APPENDICE

Stored Procedure Caricamento Staging

```
CREATE OR REPLACE PROCEDURE GMDM_STG_L2.SP_STAGING_GRIP(CHARACTER VARYING(50),
CHARACTER VARYING(50), CHARACTER VARYING(50), CHARACTER VARYING(50), BIGINT,
TIMESTAMP)

RETURNS INTEGER

LANGUAGE NZPLSQL AS

BEGIN_PROC

DECLARE

       IN_SRC_SYSTEM_CD ALIAS FOR $1;

       IN_SRC_TABLE_NAME_CD ALIAS FOR $2;

    IN_SRC_FACILITY_CD ALIAS FOR $3;

       IN_SRC_INST_CD ALIAS FOR $4;

       IN_INFORMATICA_CD ALIAS FOR $5;

       IN_TIMESTAMP_RIF ALIAS FOR $6;

       V_SRC_SCHEMA VARCHAR;

       V_SRC_FILTER_DESC VARCHAR;

       V_FLUID_QUERY_ALIAS VARCHAR;

       V_PRIMARY_KEY_DESC VARCHAR;

       V_STG_SCHEMA_2_LEV    VARCHAR;

       V_STG_TABLE_NAME_2_LEV   VARCHAR;

       V_FULL_LOAD_FLG CHAR(1);

       V_STG_ROWS_NBR INTEGER;

       V_STG_ROWS_INS_NBR INTEGER;

       V_STG_ROWS_UPD_NBR INTEGER;

       V_STG_ROWS_DEL_NBR INTEGER;

       SQL VARCHAR;

       FLUID_QUERY VARCHAR;

       QUERY_INSERT VARCHAR;

       columns_CLOB_sub varchar;

    cur RECORD;

       columns varchar;

       cont integer;

       maxTMSTMP varchar(20);

       ERR_MSG VARCHAR(4000);

       TEMP_TABLE_NAME VARCHAR;

       TEMP_TABLE_NAME_UPD VARCHAR;
```

```
	TEMP_TABLE_NAME_DEL VARCHAR;

	FINALCOMMENT VARCHAR;

BEGIN

	columns :='';

	columns_CLOB_sub :='';

	cont :=0;

	v_STG_ROWS_INS_NBR :=0;

	v_STG_ROWS_UPD_NBR :=0;

	v_STG_ROWS_DEL_NBR :=0;

	v_STG_ROWS_NBR := 0;

-- LOG INIZIALIZATION---------------------------------------------

	insert into GMDM_STG_L2.TBSTLOG_LOAD_STG

	(START_TMSTP,END_TMSTP,INFORMATICA_CD,SRC_SYSTEM_CD,SRC_TABLE_NAME_CD,SRC_
FACILITY_CD,SRC_INST_CD,LOAD_STATUS_CD,

	LOAD_ERR_CD,STG_L2_ROWS_INS_NBR,STG_L2_ROWS_UPD_NBR,STG_L2_ROWS_DEL_NBR,LO
AD_START_TMSTMP )

	values ('1900-01-01',
NULL,IN_INFORMATICA_CD,IN_SRC_SYSTEM_CD,IN_SRC_TABLE_NAME_CD,IN_SRC_FACILITY_CD,I
N_SRC_INST_CD,'RUNNING',

	NULL,NULL,NULL,NULL,NOW());

	COMMIT;

	--RETRIEVE CONFIGURATIONS FROM TABLE TBSTCNF_LOAD_STG---------------

	SELECT
SRC_FILTER_DESC,SRC_SCHEMA,FLUID_QUERY_ALIAS,STG_SCHEMA_2_LEV,STG_TABLE_NAME_2_LE
V,PRIMARY_KEY_DESC, FULL_LOAD_FLG

	into
V_SRC_FILTER_DESC,V_SRC_SCHEMA,V_FLUID_QUERY_ALIAS,V_STG_SCHEMA_2_LEV,V_STG_TABLE
_NAME_2_LEV,V_PRIMARY_KEY_DESC,V_FULL_LOAD_FLG

	FROM GMDM_STG_L2.TBSTCNF_LOAD_STG

    WHERE SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND SRC_TABLE_NAME_CD =
IN_SRC_TABLE_NAME_CD  AND SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND  SRC_INST_CD =
IN_SRC_INST_CD;

--RETRIEVE COLUMN NAMES-------------------------------------------

	SQL := ' select *

		from table WITH final

			('||V_FLUID_QUERY_ALIAS||'(''''','''',''select column_name,
data_type

			from ALL_TAB_COLUMNS

			where
TABLE_NAME='''''||IN_SRC_TABLE_NAME_CD||'''''ANDOWNER='''''||V_SRC_SCHEMA||''''''
''))';

	FOR cur IN EXECUTE SQL LOOP

--- SKIP DATA TYPE NOT SUPPORTED

	if not(cur.DATA_TYPE like 'ROWID' or cur.DATA_TYPE like '%RAW%' or
cur.DATA_TYPE like 'BLOB'
```

```
            or cur.DATA_TYPE like '%BINARY%' or cur.DATA_TYPE like 'ANYDATA' or
cur.DATA_TYPE like 'UNDEFINED') then

--- first iteraction

      if cont=0 then

      if cur.DATA_TYPE LIKE '%CLOB%' THEN

            columns_CLOB_sub:='dbms_lob.substr("'||cur.COLUMN_NAME||'",4000,1)';

      ELSE

            columns_CLOB_sub:='"'||cur.COLUMN_NAME||'"';

      END IF;

      columns:='"'||cur.COLUMN_NAME||'"';

      else if cur.DATA_TYPE LIKE '%CLOB%' THEN

            columns_CLOB_sub:=columns_CLOB_sub ||
,dbms_lob.substr("'||cur.COLUMN_NAME||'",4000,1)';

      ELSE

      columns_CLOB_sub:=columns_CLOB_sub || ',"'|| cur.COLUMN_NAME||'"';

      END IF;

      columns:=columns||',"'|| cur.COLUMN_NAME||'"';

      end if;

      cont:=cont+1;

      end if;

      END LOOP;

-- STAGING ON TEMPORARY TABLE LOAD

      TEMP_TABLE_NAME:=IN_SRC_TABLE_NAME_CD||IN_SRC_FACILITY_CD||IN_SRC_INST_CD|
|'_temp';

      EXECUTE IMMEDIATE 'create temp table '||TEMP_TABLE_NAME || ' '

                                          ||'as select '||columns||' from
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV||' limit 1';

      EXECUTE IMMEDIATE 'delete from '||TEMP_TABLE_NAME;

--insert data into LEV 1 temp table

                  FLUID_QUERY := 'insert into '||TEMP_TABLE_NAME

                  || ' select * '

                  || ' from table WITH final '

                  || '('||V_FLUID_QUERY_ALIAS||'('''',''''),''select
'||COLUMNS_clob_sub||' from '||V_SRC_SCHEMA||'.'||IN_SRC_TABLE_NAME_CD||' ''));';

                  execute immediate FLUID_QUERY;

                  V_STG_ROWS_NBR := ROW_COUNT;

--  BEGIN CASE OF TABLE SOURCE WITH INCREMENTAL LOAD
--

      IF V_FULL_LOAD_FLG='N' THEN

--INSERT
```

```
            execute immediate 'insert into
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||'
('||columns||',GMDM_STG_FACILITY_CD,GMDM_STG_SRC_INST_CD,GMDM_STG_RCRD_STS_CD,GMD
M_STG_INS_TMSTMP,GMDM_STG_INS_INFORMATICA_CD) '

            ||' select '

            ||columns||','

            ||''''||IN_SRC_FACILITY_CD ||''' as GMDM_STG_FACILITY_CD, '

            ||''''||IN_SRC_INST_CD ||''' as GMDM_STG_SRC_INST_CD, '

            ||'''A'' as GMDM_STG_RCRD_STS_CD, '

            ||' '''||IN_TIMESTAMP_RIF||''' as GMDM_STG_INS_TMSTMP,'

            ||''||IN_INFORMATICA_CD||' as GMDM_STG_INS_INFORMATICA_CD'

            ||' from
'||IN_SRC_TABLE_NAME_CD||IN_SRC_FACILITY_CD||IN_SRC_INST_CD||'_temp n'

            ||' where ('||V_PRIMARY_KEY_DESC||') in ('

            ||' SELECT '||V_PRIMARY_KEY_DESC||' FROM '||TEMP_TABLE_NAME

            ||' MINUS'

            ||' SELECT '||V_PRIMARY_KEY_DESC||' FROM
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV||' where
GMDM_STG_RCRD_STS_CD=''A'''

            || ' AND GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''')';

            v_STG_ROWS_INS_NBR := ROW_COUNT;

--END INSERT

--UPDATE


      TEMP_TABLE_NAME_UPD:=IN_SRC_TABLE_NAME_CD||IN_SRC_FACILITY_CD||IN_SRC_INST
_CD||'_upd_temp';

            execute immediate 'create temp table '||TEMP_TABLE_NAME_UPD ||

            ' as

            (

            SELECT * FROM '||TEMP_TABLE_NAME ||

            ' MINUS

            SELECT '||columns||'

            FROM '||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV||'

            where GMDM_STG_RCRD_STS_CD=''A'' AND
GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''')';

        execute immediate 'update
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||' set
GMDM_STG_RCRD_STS_CD=''S'',GMDM_STG_HST_TMSTMP='''||IN_TIMESTAMP_RIF||''',GMDM_ST
G_HST_INFORMATICA_CD='||IN_INFORMATICA_CD

            ||' where ('||V_PRIMARY_KEY_DESC||') in (select
'||V_PRIMARY_KEY_DESC||' from '||TEMP_TABLE_NAME_UPD||')'
```

```
            ||' AND GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''' '

            || 'AND GMDM_STG_RCRD_STS_CD=''A''';

            v_STG_ROWS_UPD_NBR := ROW_COUNT;

            execute immediate 'insert into
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||'
('||columns||',GMDM_STG_FACILITY_CD,GMDM_STG_SRC_INST_CD,GMDM_STG_RCRD_STS_CD,GMD
M_STG_INS_TMSTMP,GMDM_STG_INS_INFORMATICA_CD) '

            ||' select '

            || columns||','

            ||''''||IN_SRC_FACILITY_CD ||''' as GMDM_STG_FACILITY_CD, '

            ||''''||IN_SRC_INST_CD ||''' as GMDM_STG_SRC_INST_CD, '

            ||'''A'' as GMDM_STG_RCRD_STS_CD, '

            ||' '''||IN_TIMESTAMP_RIF||''' as GMDM_STG_INS_TMSTMP,'

            ||IN_INFORMATICA_CD||' as GMDM_STG_INS_INFORMATICA_CD'

            ||' from '||TEMP_TABLE_NAME_UPD||' n';

--END UPDATE

--DELETE


      TEMP_TABLE_NAME_DEL:=IN_SRC_TABLE_NAME_CD||IN_SRC_FACILITY_CD||IN_SRC_INST
_CD||'_del_temp';

            execute immediate 'create temp table '||TEMP_TABLE_NAME_DEL ||

            ' as

            (   SELECT '||columns||' FROM
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||' where GMDM_STG_RCRD_STS_CD=''A'' AND
GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''' '

            ||' MINUS '

            ||' select * FROM '||TEMP_TABLE_NAME||')';

                        execute immediate 'update
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||' set
GMDM_STG_RCRD_STS_CD=''S'',GMDM_STG_HST_TMSTMP='''||IN_TIMESTAMP_RIF||''',GMDM_ST
G_HST_INFORMATICA_CD='||IN_INFORMATICA_CD

            ||' where ('||V_PRIMARY_KEY_DESC||') in (select
'||V_PRIMARY_KEY_DESC||' from '||TEMP_TABLE_NAME_DEL||')'

            ||' AND GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''' '

            || 'AND GMDM_STG_RCRD_STS_CD=''A''';

                        execute immediate 'insert into
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

            ||'
('||columns||',GMDM_STG_FACILITY_CD,GMDM_STG_SRC_INST_CD,GMDM_STG_RCRD_STS_CD,GMD
M_STG_INS_TMSTMP,GMDM_STG_INS_INFORMATICA_CD) '
```

```
              ||' select '
              || columns||','
              ||''''||IN_SRC_FACILITY_CD ||''' as GMDM_STG_FACILITY_CD, '
              ||''''||IN_SRC_INST_CD ||''' as GMDM_STG_SRC_INST_CD, '
              ||'''D'' as GMDM_STG_RCRD_STS_CD, '
              ||' '''||IN_TIMESTAMP_RIF||''' as GMDM_STG_INS_TMSTMP,'
              ||IN_INFORMATICA_CD||' as GMDM_STG_INS_INFORMATICA_CD'
              ||' from '||TEMP_TABLE_NAME_DEL||' n';
                          --END DELETE
                    -- END CASE OF TABLE SOURCE WITH INCREMENTAL LOAD
--
-- BEGIN CASE OF TABLE SOURCE WITH FULL LOAD
--
             ELSIF V_FULL_LOAD_FLG='Y' THEN
                    --TRUNCATE 2 LEVEL STAGING
             maxTMSTMP:='2001-01-01 00:00:00';
             execute immediate 'delete from
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV ||
             ' where GMDM_STG_FACILITY_CD='''||IN_SRC_FACILITY_CD||''' AND
GMDM_STG_SRC_INST_CD='''||IN_SRC_INST_CD||''' ;';
                          EXECUTE IMMEDIATE 'INSERT INTO
'||V_STG_SCHEMA_2_LEV||'.'||V_STG_TABLE_NAME_2_LEV

      ||'('||columns||',GMDM_STG_FACILITY_CD,GMDM_STG_SRC_INST_CD,GMDM_STG_RCRD_
STS_CD,GMDM_STG_INS_TMSTMP,GMDM_STG_INS_INFORMATICA_CD)'
             ||' select '
             ||' n.*,'
             ||''''||IN_SRC_FACILITY_CD ||''' as GMDM_STG_FACILITY_CD, '
             ||''''||IN_SRC_INST_CD ||''' as GMDM_STG_SRC_INST_CD, '
             ||'''A'' as GMDM_STG_RCRD_STS_CD, '
             ||' '''||IN_TIMESTAMP_RIF||''' as GMDM_STG_INS_TMSTMP,'
             ||''||IN_INFORMATICA_CD||' as GMDM_STG_INS_INFORMATICA_CD'
             ||' from '||TEMP_TABLE_NAME||' n';
             v_STG_ROWS_INS_NBR := ROW_COUNT;
      END IF;
-- END CASE OF TABLE SOURCE WITH FULL LOAD
--
             UPDATE GMDM_STG_L2.TBSTLOG_LOAD_STG
             SET
               START_TMSTP='1900-01-01',
               END_TMSTP = '2045-01-01',
            LOAD_STATUS_CD = 'COMPLETED',
```

```
              STG_ROWS_NBR = V_STG_ROWS_NBR,
            STG_L2_ROWS_INS_NBR = v_STG_ROWS_INS_NBR,
            STG_L2_ROWS_UPD_NBR = v_STG_ROWS_UPD_NBR,
            STG_L2_ROWS_DEL_NBR = v_STG_ROWS_DEL_NBR,
              LOAD_END_TMSTMP = NOW()
             WHERE SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND  SRC_TABLE_NAME_CD =
IN_SRC_TABLE_NAME_CD  AND
             SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND  SRC_INST_CD =
IN_SRC_INST_CD AND INFORMATICA_CD=IN_INFORMATICA_CD;
             COMMIT;
             RETURN 0;
      EXCEPTION WHEN OTHERS THEN
      ERR_MSG := SUBSTR(SQLERRM, 1, 4000);
      IF ERR_MSG LIKE '%cannot EXECUTE NULL query%' THEN
             UPDATE GMDM_STG_L2.TBSTLOG_LOAD_STG
            SET START_TMSTP='1900-01-01',
                    END_TMSTP = '2045-01-01',
                LOAD_STATUS_CD = 'COMPLETED',
                    STG_ROWS_NBR = V_STG_ROWS_NBR,
                STG_L2_ROWS_INS_NBR = V_STG_ROWS_INS_NBR,
                STG_L2_ROWS_UPD_NBR = V_STG_ROWS_UPD_NBR,
                STG_L2_ROWS_DEL_NBR = V_STG_ROWS_DEL_NBR,
                    LOAD_END_TMSTMP = NOW()
            WHERE SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND  SRC_TABLE_NAME_CD =
IN_SRC_TABLE_NAME_CD  AND
            SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND  SRC_INST_CD =
IN_SRC_INST_CD AND INFORMATICA_CD=IN_INFORMATICA_CD;
            COMMIT;
              RETURN 0;
      ELSE
             ROLLBACK;
             UPDATE GMDM_STG_L2.TBSTLOG_LOAD_STG
            SET
                    START_TMSTP='1900-01-01',
                    END_TMSTP = '2045-01-01',
                LOAD_STATUS_CD = 'ERROR',
                    LOAD_ERR_CD = ERR_MSG,
                    STG_ROWS_NBR = V_STG_ROWS_NBR,
                STG_L2_ROWS_INS_NBR = V_STG_ROWS_INS_NBR,
                STG_L2_ROWS_UPD_NBR = V_STG_ROWS_UPD_NBR,
```

```
                STG_L2_ROWS_DEL_NBR = V_STG_ROWS_DEL_NBR,

                     LOAD_END_TMSTMP = NOW()

            WHERE SRC_SYSTEM_CD = IN_SRC_SYSTEM_CD AND  SRC_TABLE_NAME_CD =
IN_SRC_TABLE_NAME_CD  AND

            SRC_FACILITY_CD = IN_SRC_FACILITY_CD AND  SRC_INST_CD =
IN_SRC_INST_CD AND INFORMATICA_CD=IN_INFORMATICA_CD;

            COMMIT;

               RAISE EXCEPTION 'ERROR --> %', ERR_MSG;

               RETURN 1;

      END IF;

      END;

END_PROC;
```

File dei Parametri

```
[Service:icmdi04-prd]
$PMCacheDir=$PMRootDir/gmdm/cache
$$PATH_DIR=$PMRootDir/gmdm
$PMSessionLogDir=$PMRootDir/gmdm/sesslogs/ppn
$PMBadFileDir=$PMRootDir/gmdm/badfiles/ppn
$PMWorkflowLogDir=$PMRootDir/gmdm/workflowlogs
$PMTempDir=$PMRootDir/gmdm/log/ppn
$DBConnectionMDM=GMDM_GLBL_INF_USER
$DBConnectionMDMEXT=GMDM_GCPL_EXT_INF_USER
$DBConnectionSTG2L=GMDM_STG_USER_Z2
$$FCLTY_CD=0071
$$SRC_INST_CD=DEF
$$SchemaSTG2L=GMDM_STG_L2_OWNER
$$SchemaMDM=GMDM_GLBL_OWNER
$$SchemaMDMLOG=GMDM_GLBL_DISCARD_OWNER
$$SchemaMDMEXT=GMDM_GCPL_EXT_OWNER
$$INTRFC_TYP_CD=PPN_SES
$$APPLCTN_CD=PPN_SES
$$SRC_CD=PPN_SES
$$DAYOFWEEK_FULL=6
[s_ins_gmdm_load_sts]
$PMSessionLogFile=s_gmdm_ppn_ins_load_sts.log
$BadFile1=mdm_load_sts_ins_ppn.bad
$BadFile2=gmdm_stg_load_sts_ppn.bad
[s_upd_gmdm_load_sts]
$PMSessionLogFile=s_gmdm_ppn_upd_load_sts.log
$BadFile1=mdm_load_sts_upd_ppn.bad
```

BIBLIOGRAFIA

[Albano 12] Antonio Albano, *Basi di Dati di Supporto alle Decisioni*, Appunti delle lezioni del corso di Sistemi Informatici Direzionali, 2012.

[Gartner 16] Gartner Inc., *Magic Quadrant for Data Integration Tools*, https://www.gartner.com/doc/3393017/magic-quadrant-data-integration-tools, 2016.

[Gartner 16] Gartner Inc., *Magic Quadrant for Operational Database Management Systems*, https://www.gartner.com/doc/3467318/magic-quadrant-operational-database-management, 2016.

[Gebel 03] Erika Gebel, *Making Insulin. A behind-the-scenes Look at Producing a Lifesaving Medication*, http://www.diabetesforecast.org/?referrer=http://www.diabetesforecast.org, 2003.

[Holland 04] Sarah Holland, e Bernardo Bátiz-Lazo, *The Global Pharmaceutical Industry*, 2004.

[Informatica 13] Informatica Corporation, *Informatica PowerCenter Express*, User guide, 2013.

[Kimball 04] Ralph Kimball, *The Data Warehouse ETL Toolkit*, Wiley, 2004.

[SAP 12] SAP, *SAP Business Objects*, User guide, 2012.

[SDG Group 16] *Profilo aziendale di SDG Group*, http://www.sdggroup.com, 2016.

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia, mamma, papà, Daniela senza di voi tutto ciò non sarebbe stato possibile. Ma anche a mio zio e ai miei cugini. Grazie per avermi sempre incoraggiato e sostenuto in questi anni.

Ringrazio tutti i colleghi più cari con cui in questi anni ho condiviso gioie, dolori e giornate tra i banchi o a studiare per un esame. Li ringrazio per l'appoggio e per tutte le volte che mi hanno aiutato a passare le difficoltà incontrate durante il mio percorso accademico. Grazie mille Ciccio, Fausto, Tommaso, Paolo, Andrea, Alfredo e Massimo.

Ringrazio i miei coinquilini storici, Biagio e Francesco, che mi sopportano da anni, ma anche i cari Angelo, Mimmo e Biondo, che hanno reso più piacevole la mia permanenza a Pisa.

Un ringraziamento va anche a tutti i professori che ho incontrato durante il mio percorso di studi, è grazie alla preparazione e passione di alcuni di loro se sono riuscito a raggiungere questo traguardo.

Ringrazio gli amici di vecchia data "non pisani" che mi sono stati vicini nonostante la distanza geografica che ci ha separato in questi anni universitari. Grazie Gaetano, Roberto, Luigi, Francesco, Riccardo, Francesca Amat, Pepi, Giuseppe, Francesca La Cò, Luca, Elisa, Valeria e Paolo.

Ringrazio tutta la gente che ho avuto l'opportunità di conoscere a Pisa in questi anni, veramente tante persone, alcune mi sembra di conoscerle da una vita. Vi ringrazio per avermi sostenuto e saputo ascoltare sia nei momenti belli che in quelli brutti. Grazie Alessandra, Tiziana, Maria Lara, Giulia, Massi, Rosy e Alessandro. Una menzione speciale va fatta a dei grandi amici con cui ho vissuto tante avventure. Gli amici della Dama, un grande gruppo di amici che oltre a trionfare sui campi di calcetto pisani si vuole un bene dell'anima. Grazie a tutti coloro che sono identificabili in questa categoria che rappresenta più che una squadra di calcetto, ma soprattutto a Michelangelo, Simone, Massi, Andrea, Pierluigi, Tava, Antonino, Gigi e Giuseppe.

Ringrazio ESN Pisa che ha allietato i miei ultimi anni pisani, mi ha permesso di crescere tantissimo dal punto di vista personale e di conoscere tante belle persone che vivono sparse per il globo. Tra loro in particolare ringrazio Javier, Mathieu, Adam, Aline, Tharanga, Laura, Piotr, Oscar, Willy, Eric e Anna. Ma soprattutto in associazione ho conosciuto tante belle persone con cui ho contribuito in questi anni a creare qualcosa di bello che ci ha permesso di toglierci tante soddisfazioni. Tante sono le persone che hanno contribuito e che ringrazio ma particolarmente ringrazio gli amici Marco, Angelo, Daniele, Sara, Gloria, Andrea e Fabrizio.