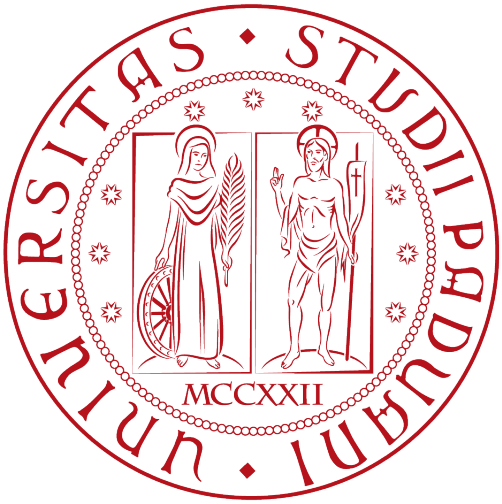

Università degli studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica
A.A. 2017/2018

Studio di fattibilità dell'aggiornamento del sistema CRM: importazione, esportazione e monitoraggio dei dati tra Oracle e SqlServer

Laureando:
Iris Balaj

Relatore:
Mauro Conti

Azienda ospitante:
INDUSTRIES SPA

Tutor Aziendale:
Fabrizio Pittalis

Sommario

Il presente documento descrive obiettivi e attività dello stage curriculare svolto dal laureando Iris Balaj presso l'azienda INDUSTRIES S.p.A . La durata complessiva dello stage è stata di circa 304 ore.
Lo scopo principale dello stage è stato quello di effetturare uno studio di fattibilità della reingegnerizzazione di un software, una valutazione dei vantaggi e dei punti critici nello sviluppo di una versione aggiornata, attraverso nuove tecnologie.
Il passo successivo allo studio di fattibilità è stato quello di sviluppare un prototipo funzionante che includesse al suo interno le funzionalità minime richieste dal sistema. Nel resto del documento saranno descritti tutti i concetti alla base del progetto CRM e veranno prese in analisi le tecnologie utilizzate e i problemi affrontati durante lo sviluppo.

Convenzioni tipografiche

- Le parole in lingua inglese senza corrispettivo italiano saranno scritte in *corsivo*;
- Le parole che necessitano di una spiegazione esplicita sono marcate da una **g** pedice per segnalarne la presenza nel **Glossario** a fine documento

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

Lo stage, svoltosi presso l'azienda **Industries S.P.A.** e con la supervisione del tutor Fabrizio Pittalis, consisteva nello studio di fattibilità dell'aggiornamento del progetto CRM, per capire se reingegnerizzare il progetto in maniera più efficace.
Oltre allo studio di fattibilità, era richiesto lo sviluppo di una versione dimostrativa del progetto basata su un set ristretto di dati, che potesse mostrare l'utilità delle tecnologie selezionate dallo stagista, comprensiva di una struttura di reportistica di log da sviluppare con software utilizzati dall'azienda.
Implicito nello sviluppo di tale **demo$_g$**, era richiesto lo studio di tecnologie proprie di Industries S.P.A, tipiche di un'azienda tessile di larga scala.

1.1 L'Azienda

L'azienda è una sede amministrativa di Moncler, marchio italiano di alta moda specializzato in vestiario invernale.
Nata francese nel 1952, Moncler diventa italiana nel 1992 e ad oggi vanta circa 3500 dipendenti ed oltre 200 punti vendita in tutto il mondo; è quotata nella borsa di Milano dal 2013, e nel 2017 ha superato 1,1 miliardi di euro di fatturato.
Le sedi amministrative principali sono situate in Italia a Milano, Trebaseleghe (Padova, sede dello stage) e Piacenza, e nel resto del mondo in Giappone e Stati Uniti.

1.2 Obiettivi di stage

L'obiettivo dello stage era di valutare la complessità di un possibile aggiornamento del progetto CRM con tecnologie più recenti.
Il progetto CRM consiste nell'alimentazione di un database con dati relativi a capi vendibili in una determinata campagna vendite a cadenza stagionale.
Il progetto era stato inizialmente sviluppato nel 2007 e consiste nell'estrazione dei dati dal database Oracle aziendale e popolamento di file di testo che poi vengono inviati ad un fornitore, il quale si occupa di caricare i dati contenuti in tali file nei database a cui fanno riferimento le campagne vendite. Queste campagne nello specifico sono degli showroom situati a Milano, New York e Tokyo e ricevono dati diversi a seconda delle politiche relative ad ogni stato.
La richiesta dei ***Service Manager$_g$*** era quella di ridurre la dispersività del programma, data dalla creazione di numerosi file, mantendendo ovviamente la consistenza e possibilmente aumentando la velicità dell'esecuzione.

1.3 Principali problematiche

La maggior parte delle aziende nel settore della moda utilizza un gestionale particolare, Stealth 3000, del quale non esiste documentazione ufficiale di pubblico dominio, dato che viene personalizzato per ogni azienda che ne abbia la licenza, e ciò implica che tutte le logiche dell'azienda per salvare ed estrarre i dati relativi ad ogni aspetto del settore della moda debbano essere spiegate da una persona dedicata, nel caso di questo stage dal tutor aziendale, per cui l'accesso ad informazioni non legate a tecnologie e programmazione, era necessariamente rallentato, nonché di non banale comprensione data l'enorme vastita di contesti.

2 Analisi e Pianificazione

2.1 Analisi del contesto

I motivi principali per cui si stesse discutendo un approccio diverso al trasferimento dei dati relativi al progetto CRM, erano legati alla eccessiva dispersività del progetto esistente, che crea molti file di testo e necessita dell'affidamento ad un fornitore che si occupi del caricamento dei dati nei ***database***$_g$ relativi ad ogni stato (Italia, USA e Giappone).
La natura del sistema esistente era tale da consistere di schedulazioni impostate a diverse fasce orarie da parte di aziende diverse. La prima schedulazione era relativa alla creazione dei *file* da parte di Industries, e la seconda era impostata dal fornitore a 6 ore di distanza, di comune accordo, per il caricamento effettivo dei dati. Sebbene l'esecuzione del programma di Industries che genera i *file* fosse molto breve, si era deciso di mantenere un discreto margine di tempo per intervenire in caso di errori prima del caricamento dei dati da parte del fornitore.
Nell'ottica di interrompere le relazioni con tale fornitore, si è deciso di considerare l'ipotesi di caricare i dati direttamente nei *database* a cui si riferiscono gli showroom, avendone nel tempo ottenuto l'accesso diretto. Tali database sono di proprietà di un fornitore che si occupa della creazione anche dei ***software***$_g$ installati nei vari punti vendita, principalmente tablet, grazie ai quali i vari clienti possono visualizzare il campionario ed eseguire gli evenutali ordini. Gli ordini dei clienti vengono poi trasferiti nello stesso modo dei caricamenti, sempre via file e tramite un fornitore, con un processo inverso rispetto a quello della popolazione dei database con i campionari, generando dei *report* interni ad Industries, tramite Oracle Reports di cui l'azienda possiede la licenza.

2.2 Piano di Lavoro

Piano di lavoro

Lo stage svolto in azienda ha avuto una durata stabilita di circa 310 ore, suddivise in 10 settimane da 4 giorni, alla fine della maggior parte delle quali era prevista una milestone.

2.2.1 Pianificazione delle attività

La pianificazione delle attività è stata la seguente:

Settimana	Task
06-09 Agosto (1a Settimana)	Studio del programma esistente relativo all'export delle informazioni necessarie al CRM, con supporto degli sviluppatori di quel progetto (16h) e creazione di schemi riassuntivi (16h).
21-23 Agosto (2a Settimana)	Studio del programma esistente relativo alla fase di import ordini, con supporto degli sviluppatori di quel progetto (16h) e creazione schemi riassuntivi (16h).
27-30 Agosto (3a Settimana)	Analisi ed eventuali test delle diverse modalità offerte dal mercato per il passaggio dei dati (28h) e discussione con il Responsabile riguardo la validità della scelta(4h).
03-06 Settembre (4a Settimana)	Sviluppo di una versione dimostrativa del programma di export con la nuova tecnologia, con funzionalità e set di dati minime (32h).
10-13 Settembre (5a Settimana)	Sviluppo di una versione dimostrativa del programma di export con la nuova tecnologia, con funzionalità e set di dati minime (32h).
17-20 Settembre (6a Settimana)	Stesura documentazione tecnica di rilascio della versione dimostrativa di export (32h).
24-27 Settembre (7a Settimana)	Sviluppo di una versione dimostrativa del programma di export con la nuova tecnologia, con funzionalità e set di dati minime (32h)
01-04 Ottobre (8a Settimana)	Sviluppo di una versione dimostrativa del programma di export con la nuova tecnologia, con funzionalità e set di dati minime (32h)

08-11 Ottobre (9a Settimana)	Stesura documentazione tecnica di rilascio della versione dimostrativa di import (32h).
15-18 Ottobre (10a Settimana)	Verifica ed unione delle documentazioni (16h) e discussione con il Responsabile ed i Service Manager sulla qualità del prodotto sviluppato, fattibilità in termini di tempistiche e complessità, e sul'eventuale realizzazione del progetto per la prossima campagna vendite (4h)

Tabella 1: Pianificazione: Distribuzione Settimanale

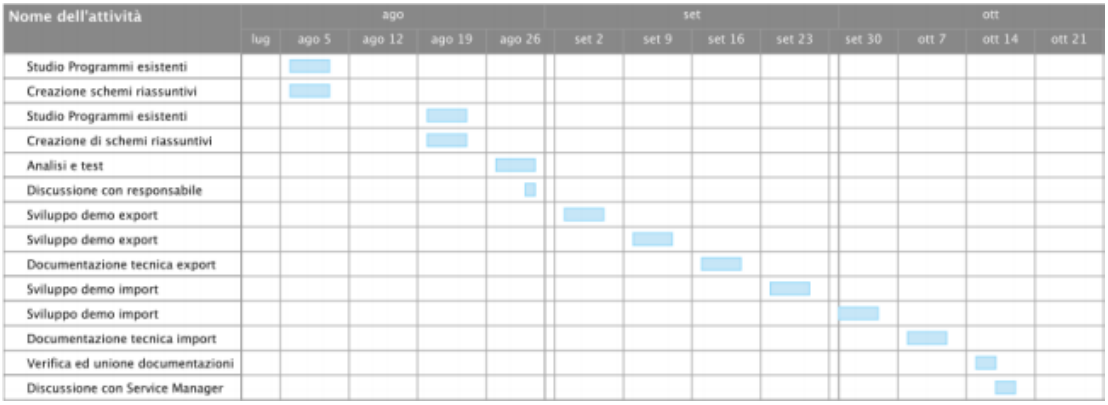

Figura 1: Diagramma delle attività

2.2.2 Obiettivi

Gli obiettivi dello stage sono codificati con le seguenti notazioni:

- <min> per gli obiettivi minimi, vincolanti in quanto richieste primarie del committente;
- <max> per gli obiettivi massimi, inclusi quelli desiderabili ed opzionali, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- <for> per gli obiettivi formativi, rappresentanti valore aggiunto in termini culturali e di conoscenze da acquisire dallo stagista.

Minimi

- Min1: ***Reverse Engeneering***$_g$ del software di interfaccia attuale;
- Min2: Individuazione della corretta architettura necessaria per la nuova implementazione;
- Min3: Autonomia nei test di estrazione ed importazione dati su Sql Server.

Massimi

- Max1: Capacità di ottimizzazione dei processi;
- Max2: Superamento dei *bug* e dei limiti contenuti nei programmi esistenti;
- Max3: Individuazione delle soluzioni alle problematiche rilevate.

Formativi

- Acquisizione di abilità funzionali sulla gestione degli orgini su ERP;
- Acquisizione di conoscenze tecniche su strumenti di ETL;
- Interazione con i Service Manager;
- Stesura di documentazione Tecnica.

3 Studio delle Tecnologie

3.1 Stealth 3000

La maggior parte delle aziende tessili di larga scala in Italia utilizza un ***ERP$_g$*** specifico, Stealth 3000, sviluppato dall'azienda italiana Dedagroup.
In quanto ERP connette tutti gli applicativi utilizzati dall'azienda, oltre ad essere un gestionale disegnato specificatamente per il mercato della moda.
Si tratta di un'***apple$_g$***t Java, accessibile solo internamente all'azienda da Internet Explorer e Firefox.
Di seguito verrà illustrato il funzionamento di Stealth 3000 negli ambiti che sono stati toccati dal progetto CRM, che sono solo una minima parte di quanto offerto dall'ERP. Gli ambiti in questione sono i Soggetti, ovvero i clienti in generale, e le loro condizioni commerciali, gli oggetti e le loro classificazioni, ovvero gli articoli che vengono venduti ed infine i listini di vendita.
Tutti i contenuti anagrafici dei dati rappresentati sono fittizi, ottenuti dai manuali d'uso a scopo formativo.

3.1.1 Soggetto

Sono una qualunque entità fisica, giuridica, gestionale, organizzativa con cui l'azienda intrattiene rapporti di business.
Esempi di Soggetti sono: Clienti, Fornitori, Agenti,Terzisti, Importatori, Distribution Centers, Reparti interni.
L'archivio dei soggetti ne contiene i dati anagrafici fissi nel tempo (Ragione Sociale, Partita IVA, Indirizzo, ecc). Il soggetto ha una lista di indirizzi associati che ne definiscono punti di riferimento, ad esempio potrebbe avere un indirizzo di fatturazione ed un indirizzo di spedizione diversi tra loro.
Il soggetto può avere contemporaneamente più **Ruoli**; Esso rappresenta il tipo di rapporto che il Soggetto ha con l'Azienda, e può essere di tre tipi: **Cliente**, **Fornitore** o **Agente**. A seconda del ruolo ci sono differenti **Condizioni Commerciali**. Di seguito in *figura 2* la form di anagrafica Soggetto vista da Stealth 3000:

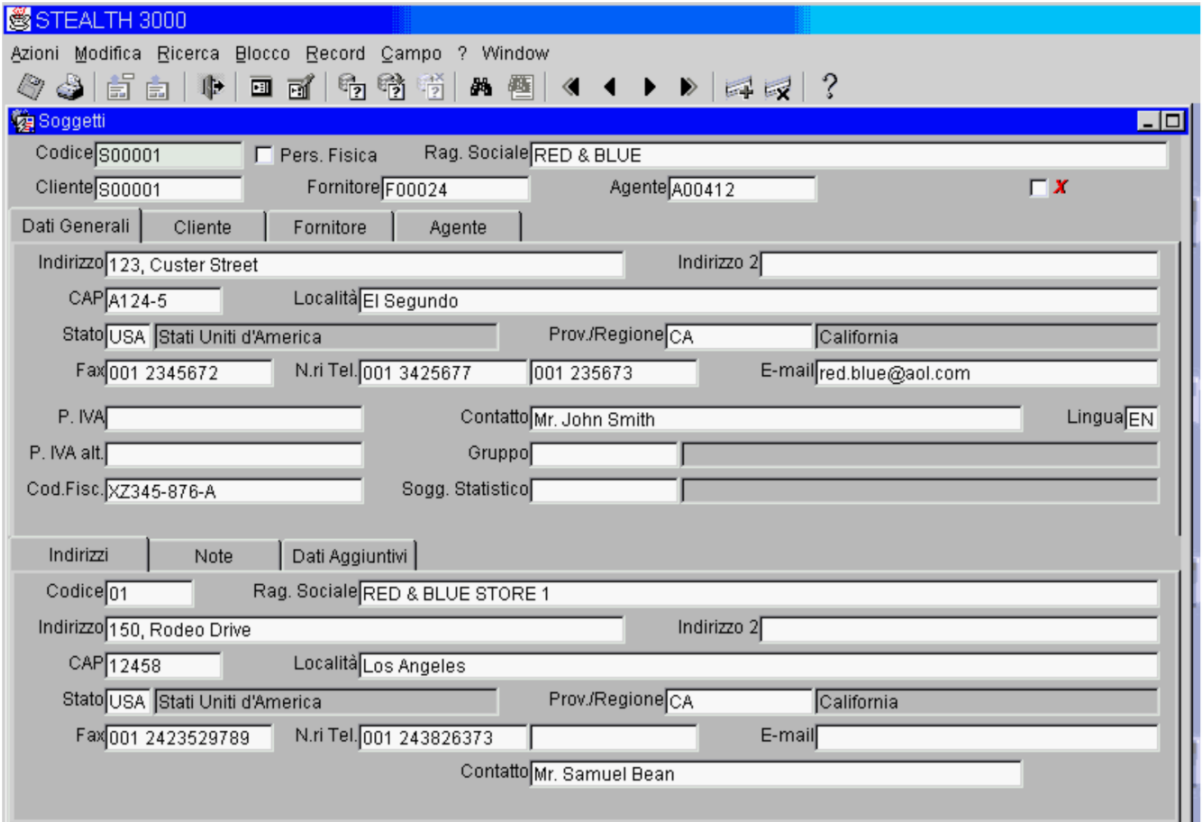

Figura 2: Anagrafica Soggetti

Dati di testata:

Dato	Descrizione
Codice	Codice Soggetto: in fase di creazione il codice viene attribuito automaticamente all'uscita del campo da un numeratore pubblico ma può essere forzato dall'utente. Il sistema effettuerà in automatico un controllo di unicità del codice all'interno del database.
Persona Fisica	Flag che indica che il soggetto è una persona fisica anzichè giuridica, per cui il campo della Ragione sociale dovrà essere sostituito dall'inserimento di Cognome e Nome.
Ragione Sociale	Descrizione della missione aziendale.
Cliente	Codice corrispondente al ruolo cliente (se esistente) del soggetto. Il codice sarà assegnato in automatico alla generazione del ruolo ed il campo presente servirà per ricerche mirate ai soli ruoli cliente.
Fornitore	Codice corrispondente al ruolo Fornitore (se esistente) del soggetto. Il codice sarà assegnato in automatico alla generazione del ruolo ed il campo presente servirà per ricerche mirate ai soli ruoli Fornitore.
Agente	Codice corrispondente al ruolo agente (se esistente) del soggetto. Il codice sarà assegnato in automatico alla generazione del ruolo ed il campo presente servirà per ricerche mirate ai soli ruoli agente.

Tabella 2: Soggetti: Dati di testata

Dati Generali:

Dato	Descrizione
Indirizzo	Primo campo dell'indirizzo: è un campo ad inserimento di testo libero, lungo fino a 50 caratteri ed è il campo principale di esposizione dell'indirizzo sintetico del cliente.
Indirizzo 2	Secondo campo dell'indirizzo: è un campo aggiuntivo di testo libero che viene utilizzato quando il primo sia insufficiente oppure si voglia riportare dati su una riga diversa dell'etichetta completa del soggetto.
CAP	Indicazione del Codice di avviamento postale della nazione a cui appartiene il soggetto. L'obbligatorietà di questo campo è determinata da un parametro della tabella stati, in corrispondenza dello stato che sarà indicato più sotto.
Località	Città, paese, frazione di identificazione dell'indirizzo. In molte visualizzazioni sintetiche dei codici soggetti accompagna la ragione sociale.
Stato	Codice corrispondente nella tabella degli stati a cui sono collegati molti controlli sull'inserimento degli altri dati nella form come Codice Fiscale, Partita IVA, Prov/Reg., Formato del codice Bancario, Valuta di default.
Prov/Regione	Valore che può essere reso obbligatorio per lo stato inserito con controllo di relazionalità con la tabella collegata a quella degli stati.
Fax– N.Tel–E-mail	Dati di libero inserimento che potranno essere presentati su altre form, stampe oppure essere utilizzati da procedure personalizzate.

P.IVA	Codice di Partita IVA del soggetto. È obbligatoria e all'uscita del campo attiva i controlli formali sul codice (secondo il paese di appartenenza) e di codici duplicati già presenti nel database (controllo non bloccante).
Contatto	Campo libero di inserimento dei dati utili all'utente.
Lingua	Lingua di default per la gestione dei documenti del soggetto, viene proposta in automatico dalla tabella degli stati, ma può essere modifica dall'utente.
P.IVA alt	Indicazione del codice di Partita IVA internazionale che solitamente è formata dal codice ISO dello stato di appartenenza concatenato con il codice di Partita Iva inserito nel campo precedente.
Codice Fiscale	Può essere obbligatorio per lo stato e per la natura fiscale del soggetto.
Gruppo	Codice di soggetto a cui il soggetto corrente è legato da rapporti di gruppo.
Sogg. Statistico	Codice Soggetto a cui legare più soggetti al fine di analisi statistiche raggruppate.
Società Intercompany	Codice societario assegnato al cliente se appartiene al dominio delle società definite "Intragruppo". Questi soggetti avranno particoli processi di trattamento per i rapporti attivi e passivi.

Tabella 3: Soggetti: Dati Generali

3.1.2 Condizioni di Vendita

Di seguito in *figura 3* vediamo la form delle Condizioni di Vendita (colloquialmente, Condizioni Commerciali) accessibili dalla schermata dei Soggetti in *figura 1*.

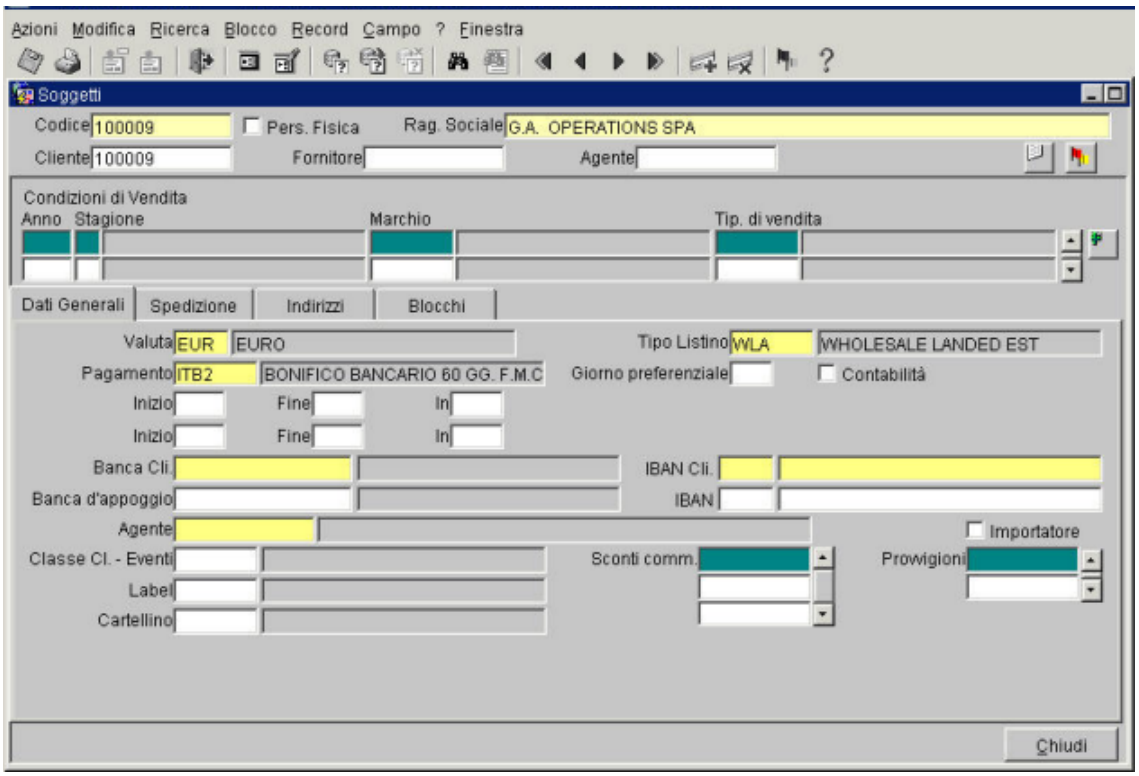

Figura 3: Anagrafica Condizioni di Vendita

Tramite questa form è possibile gestire i dati di vendita del cliente con ***record***$_g$ multipli la cui chiave è: Anno/Stagione/Marchio/Tipologia di Vendita.
Con questa configurazione è possibile memorizzare per ogni cliente diverse configurazioni di dati per la vendita in dipendenza di diverse Stagioni e/o Marchi e/ Tipologie di vendita; nel caso in cui uno o più campi chiave siano vuoti il loro significato corrisponde a "tutte le ricorrenze corrispondenti". Ad esempio nel caso in *figura 3* le condizioni di vendita sono nominalmente valide per tutte le stagioni, tutti i marchi e tutte le tipologie di vendita, mentre se fosse stato valorizzato l'anno/stagione, sarebbero state valide per tutti i marchi/tipologie di vendita di quel determinato anno/stagione.

Descrizione dei dati generali delle Condizioni di Vendita:

Dato	**Descrizione**
Valuta	Valuta di defaut che sarà proposta in tutti i documenti attivi generati per il cliente di fatturazione.
Tipo Listino	Codice del tipo listino che sarà utilizzato per il reperimento dei prezzi nella generazione dei documenti attivi.
Pagamento	Codice di pagamento che sarà proposto di default per i documenti attivi generati per il cliente quando questo è il cliente intestatario.
Giorno Preferenziale	Giorno del mese di preferenza per le scadenze dei pagamenti che saranno generati al cliente.
Contabilità	Questo *flag* determina quale tra le condizioni di vendita inserite sarà la fonte dei dati che saranno trasmessi al sistema amministrativo. Per questa ragione ci può essere una sola Condizione di Vendita con questo flag alzato.
Inizio Fine In	Tramite questa coppia di dati è possibile stabilire due periodi dell'anno (Da Giorno/Mese a Giorno/Mese) in cui le scadenze di pagamento dovranno essere ricondotte al Giorno/Mese indicato nela campo 'In'.
Banca Cli.	Codice della tabella banche del Soggetto da utilizzare come banca trassata preferenziale dei pagamenti del cliente di fatturazione.
IBAN Cli.	Codice IBAN (Diviso tra prefisso naz.-Cin) e codice vero e proprio tra quelli disposti nella tabella banche del Soggetto.
Banca di appoggio - IBAN	Codice della banca e IBAN del conto corrente preferenziale sul quale appoggiare i pagamenti del cliente. Nel caso di più linee di credito dell'Azienda è possibile così pilotare gli effetti da emettere presso l'Istituto Bancario più conveniente per i rapporti con la banca del Cliente.

Agente	Codice Agente abilitato al cliente corrente per la stagione/Marchio/Tipo di vendita prescelti.
Importatore	Flag che indica come il cliente sia da considerare Importatore per cui l'emissione del documento di vendita al cliente di fatturazione dovrà essere riferito a condizioni di vendita diverse da quelle applicate ai clienti di destinazione.
Classe Cl.–Eventi	Codice di classe del cliente a cui può essere abilitato uno o più eventi di vendita.
Label	Codice aggiuntivo dei prodotti che sarà aggiunto automaticamente alle righe ordini in maniera da caratterizzare puntualmente il fabbisogno, la disponibilità e relative assegnazioni al cliente specifico (Forniture speciali).
Cartellino	Codice aggiuntivo dall'uso uguale al precedente.
Sconti Comm.	Percentuali di sconto multiple, che saranno applicate in cascata a tutti gli ordini di cui il cliente è intestatario.
Provvigioni	Percentuali di sconto multiple, che saranno applicate in cascata all'agente abilitato al cliente.

Tabella 4: Condizioni di Vendita: Dati Generali

3.1.3 Modelli

Gli Oggetti nel package Stealth 3000 sono tutti i tipi di prodotto che possono essere gestiti come beni d'acquisto (materie prime e componenti di Distinta Base), acquistati e/o prodotti, gestiti a magazzino, venduti e così via.
Le anagrafiche Oggetti (Modello, Parte, Modello/Parte) sono gestite in due livelli: a livello di 'Gruppo', ovvero pubbliche a tutte le società e a livello 'Società', per cui l'appartenenza è limitata alla stessa, che è l'unica in grado di apportare modifiche. La maggior parte dei dati delle anagrafiche sono a livello Pubblico, mentre solo alcuni dati di Vendita e Gestione sono a livello Società.

Il Modello (*Style*) è l'oggetto che descrive la forma generica con cui classificare i Prodotti da vendere (Abiti, Giacche, Gonne ecc...). Hanno taglie e misure ma non colori o tessuti associati, non si rappresentano oggetti fisici.

Il codice del Modello è una combinazione di caratteristiche anagrafiche, nello specifico [Codifica Anno/Stagione][Linea][Codice Modello][Variante] in rispettivamente 2, 3, 5 e 2 caratteri, per un totale di 12.
Di seguito in *figura 4* si può vedere un'anagrafica di gestione dei Modelli da Stealth 3000:

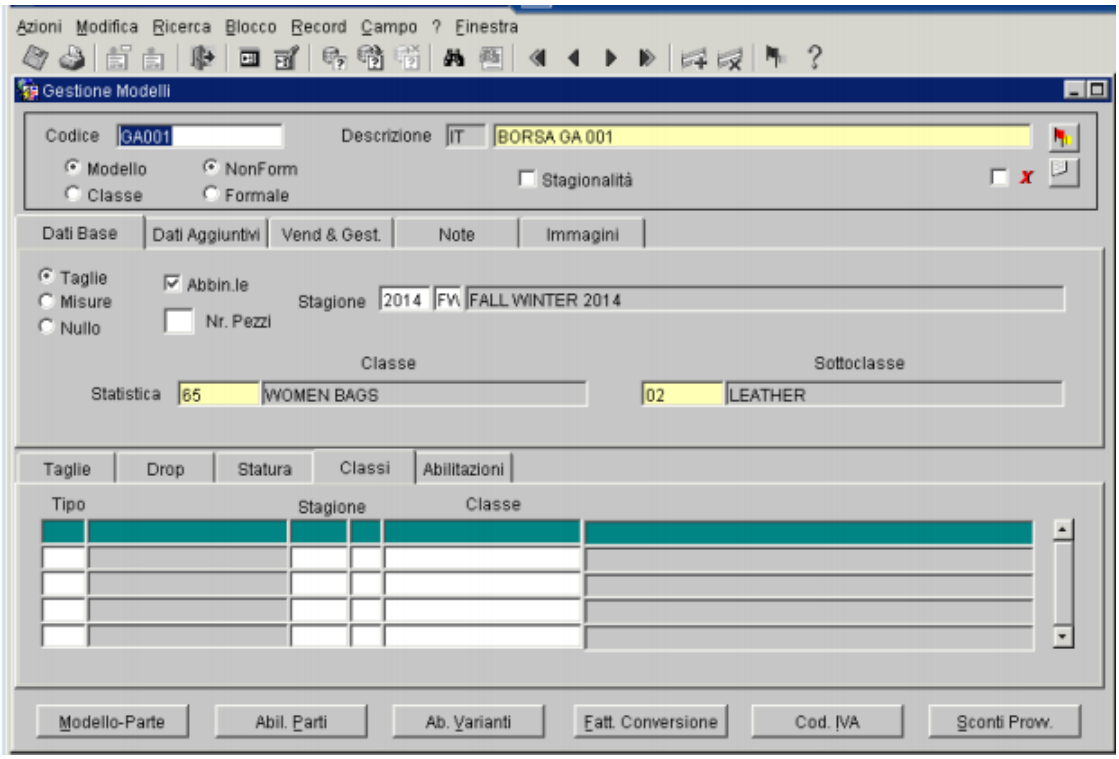

Figura 4: AnagraficaModello

Dati di Testata:

Dato	Descrizione
Codice	Codice Oggetto: l'utente può caricare un codice a suo piacimento, ma può anche saltare l'inserimento di dati in questo campo. In questo caso il sistema proporrà un codice in automatico da parte di un numeratore da configurare a sistema. In ogni caso all'uscita del campo avverrà anche un controllo di unicità del codice all'interno dell'archivio Modelli.
Descrizione	Descrizione del modello. Sarà proposta la lingua di gestione dell'utente, ma sarà possibile inserire anche decrizioni alternative nelle lingue previste a sistema, tramite il pulsante laterale con le bandiere colorate.
Modello/Classe	Radio Button con il quale indicare se l'anagrafica che si stà inserendo corrisponde ad un Modello effettivo oppure ad una classe di modelli (Ad uso della gestione dei capi formali). Dato fissato per Default come Modello.
NonForm/Formale	Radio Button per la scelta del tipo di gestione del codice corrente. Dato fissato per Default come NonForm.
Stagionalità	Questo Flag facoltativo indica, se alzato, che saranno considerate valide, per questo Modello, SOLO le abilitazioni con l'indicazione esplicita dell'Anno-Stagione. Quelle abilitazioni definite 'continuative' non saranno prese in considerazione per il Modello con questo flag alzato.
Annullo	Flag di annullamento di validità del record corrente. Con questo flag alzato il Modello risponderà ai controlli di relazionalità del database, ma sarà a tutti gli effetti non valido.

Tabella 5: Modello: Dati di Testata

Dati Base:

Dato	Descrizione
Taglie/Misura/Nullo	Pulsante a scelta esclusiva in cui si definisce che il Modello sarà gestito con l'indicazione, rispettivamente: Taglie, Misura oppure nessuna delle due. Le prime due scelte attivano dei blocchi aggiuntivi per la gestione dei dati relativi.
Abbinabile	Il flag Abbinabile indica che il Modello potrà essere utilizzato per generare un Modello Parte. Altrimenti il codice non sarà associabile ad una Parte.
Nr.Pezzi	Indicazione valida per Modelli che descrivono capi formati da più pezzi indipendenti (Tailleur, Giacca e pantaloni da neve, ecc...). (Ad uso della gestione dei capi formali).
Stagione	Stagione di nascita del Modello. E' un dato statistico che non interviene nell'abilitàzione stagionale del Modello, che quindi potrà essere comunque utilizzato in più stagioni.
Classe/Sottoclasse	Classificazione statistica del modello, utilizzabile in stampe e selezioni opertive nelle più diverse funzioni per richiamare più modelli appartenenti alla stessa Classe/sottoclasse.

Tabella 6: Modello: Dati base

3.1.4 Parti

La parte è un codice che può avere molteplici funzioni:

- Descrizione del materiale o dell'aspetto complementare ad un codice modello per la formazione del Modello-Parte.
- Codice corrispondente ad un materiale effettivo che sarà usato anche per la formazione del Modello-Parte
- Codice corrispondente ad un oggetto fisico, normalmente componenti di produzione dei Modelli- Parte (Elementi di DiBa) che non sarà mai usato per la formazione di Codici Modello-Parte

- Codice corrispondente ad un oggetto non fisico utilizzato per l'inserimento in documenti aziendali di voci immateriali quali: rimborso spese di trasporto, addebito bolli, Servizio di catering, ecc...

Essa è un codice univoco di 5 caratteri.
Di seguito in *figura 5* si può vedere una anagrafica di gestione Parti in Stealth 3000:

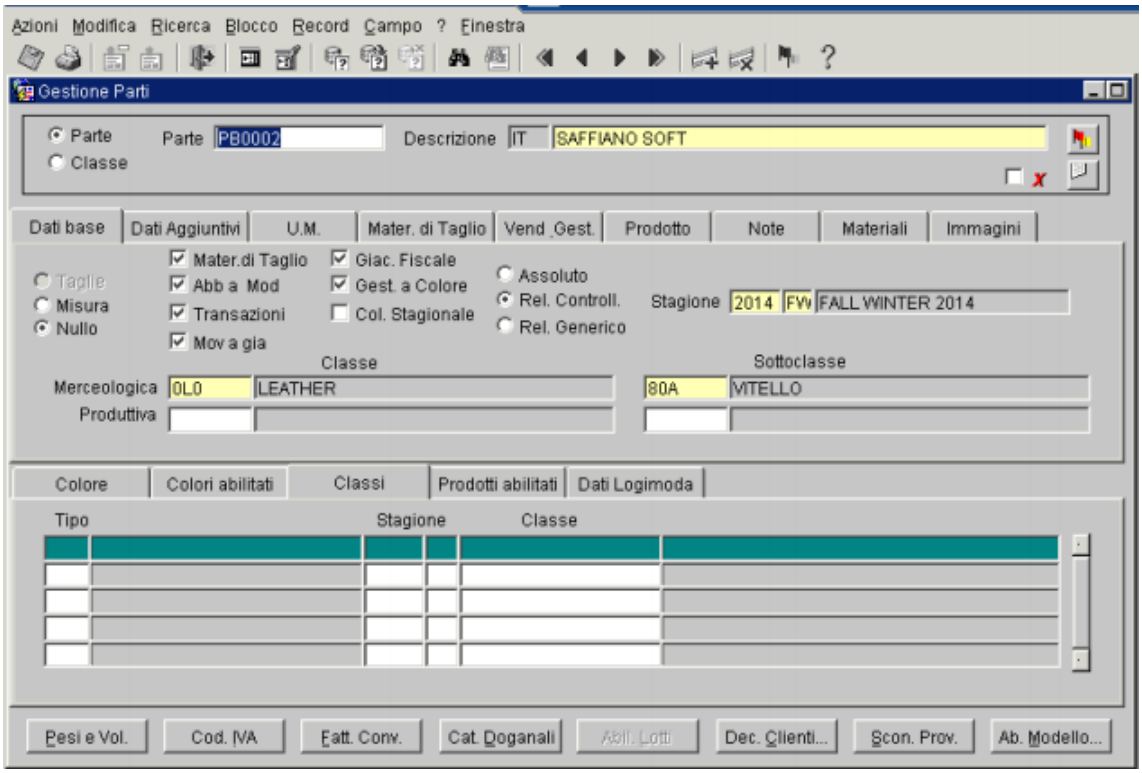

Figura 5: Anagrafica Parte

Dati di Testata:

Dato	Descrizione
Codice	Codice Oggetto: l'utente può caricare un codice a suo piacimento, ma può anche saltare l'inserimento di dati in questo campo. In questo caso il sistema proporrà un codice in automatico da parte di un numeratore da configurare a sistema. In ogni caso all'uscita del campo avverrà anche un controllo di unicità del codice all'interno dell'archivio Parti.
Descrizione	Descrizione della Parte. Sarà proposta la lingua di gestione dell'utente, ma sarà possibile inserire anche decrizioni alternative nelle lingue previste a sistema, tramite il pulsante laterale con le bandiere colorate.
Parte/Classe	Radio Button con il quale indicare se l'anagrafica che si stà inserendo corrisponde ad un Modello effettivo oppure ad una classe di modelli.
Annullo	Flag di annullamento di validità del record corrente. Con questo flag alzato la Parte risponderà ai controlli di relazionalità del database, ma sarà a tutti gli effetti non valida.

Tabella 7: Parte: Dati di Testata

Dati di base:

Dato	Descrizione
Taglie/Misura/Nullo	Pulsante a scelta esclusiva in cui si definisce che la Parte sarà gestita con l'indicazione, rispettivamente: Taglie, Misura oppure nessuna delle due. Le prime due scelte attivano dei blocchi aggiuntivi per la gestione dei dati relativi.
Mater. Di Taglio	Questo flag identifica quei materiali che potranno essere sottoposti ad operazioni di taglio per cui si renderà necessaria l'apertura di un blocco supplementare di dati per la gestione dei dati dimensionali.

Abbinabile	Il flag Abbinabile indica che la Parte potrà essere utilizzata per generare un Modello Parte. Altrimenti il codice non sarà associabile ad un Modello.
Transazioni	Flag che identifica le Parti che si possono inserire in transazioni (Doc. Di trasporto, Doc di Vendita o Acquisto ecc...) indipendentemente dal fatto che corrispondano o meno ad un oggetto fisico.
Mov. a gia.	Questo flag acceso indica che ogni movimento del codice corrente avrà effetti sul conteggio finale della giacenza in maniera algebrica. Il flag spendo indica invece codici di parti per cui non sarà possibile ottenere una interrogazione di giacenza (Materiali di consumo, parti immateriali, codici figurativi, ...).
Giac. Fiscale	Il flag indica che la parte corrente sarà da inserire nei report di giacenza ai fini fiscali.
Gest. A Colore	Questo flag alzato indica che la Parte è gestita a colori e quindi ogni riferimento al codice corrente dovrà essere completato da un riferimento ad un codice colore.
Col. Stagionale	L'indicazione di stagionalità del colore indica che per ogni stagione sarà obbligatorio inserire l'abilitazione a colori. Il non inserimento di un'abilitazione stagionale farà sì che il colore non potrà essere utilizzato al di fuori della stagione abilitata. Il flag spento fa sì che si possano definire abilitazioni di colori generici validi per tutte le stagioni, a patto che non esistano altre abilitazioni stagionali che, quindi, saranno le uniche valide per la stagione.
Stagione	Stagione di nascita della Parte. E' un dato statistico che non interviene nell'abilitàzione stagionale della Parte, che quindi potrà essere comunque utilizzato in più stagioni.

Classe/Sottoclasse	Classificazione statistica del modello, utilizzabile in stampe e selezioni opertive nelle più diverse funzioni per richiamare più modelli appartenenti alla stessa Classe/sottoclasse.

Tabella 8: Parte: Dati base

3.1.5 Modelli-Parte

L'associazione tra Modelli e Parti crea il Modello-Parte, ovvero un prodotto finito generalmente, che eredita Taglie e Misure dal Modello. Il codice è una concatenazione del Modello e della Parte, che quindi generano un codice finale di 17 caratteri.
Di seguito in *figura 6* si può vedere un'anagrafica del Modello-Parte vista da stealth 3000 ed una descrizione dei Dati:

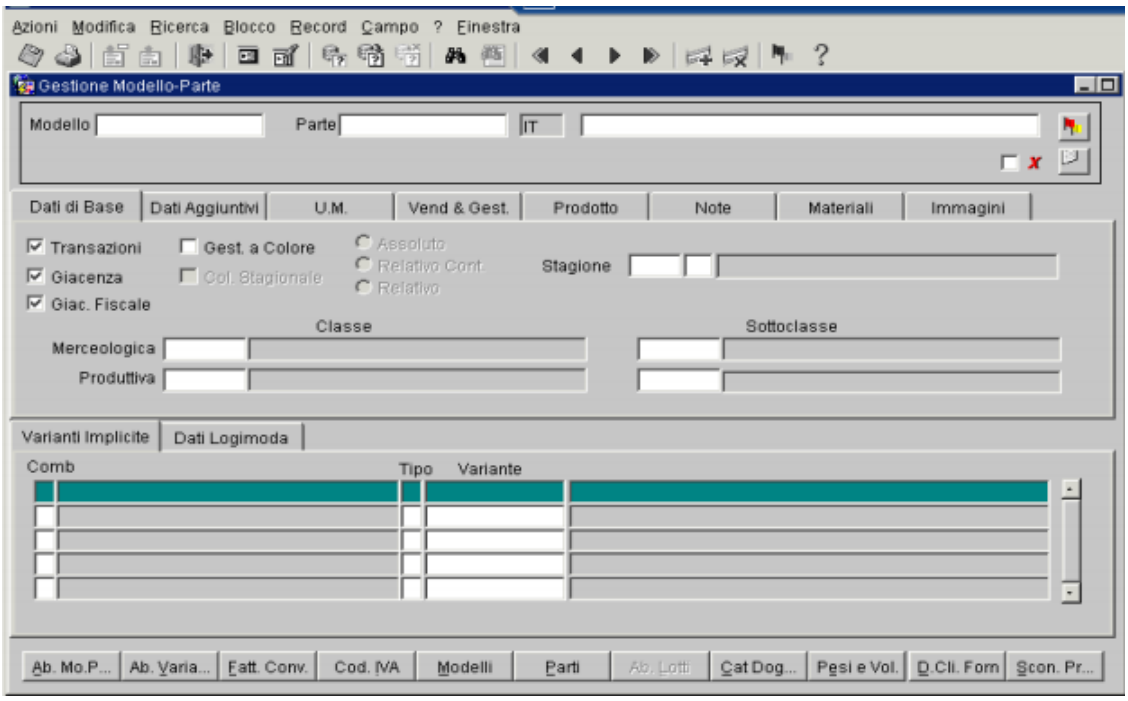

Figura 6: Anagrafica Modello-Parte

Dati di Testata:

Dato	Descrizione
Modello/Parte	Campi per cercare/abbinare Modelli e Parti.
Descrizione	Il campo é valorizzato in automatico con l'unione delle descrizioni della classe e sottoclasse statistica del Modello. Descrizione della Parte. Sarà proposta la lingua di gestione dell'utente, ma sarà possibile inserire anche decrizioni alternative nelle lingue previste a sistema, tramite il pulsante laterale con le bandiere colorate.
Annullo	Flag di annullamento di validità del record corrente. Con questo flag alzato il Modello-Parte risponderà ai controlli di relazionalità del database, ma sarà a tutti gli effetti non valido.

Tabella 9: Modello-Parte: Dati di Testata

Dati di Base:

Dato	Descrizione
Transazioni	Flag che identifica i Modelli-Parte che si possono inserire in transazioni (Doc. Di trasporto, Doc. di Vendita o Acquisto ecc...) indipendentemente dal fatto che corrispondano o meno ad un oggetto fisico.
Mov. a gia.	Questo flag acceso indica che ogni movimento del codice corrente avrà effetti sul conteggio finale della giacenza in maniera algebrica. Il flag spento indica invece codici di Modelli-Parte per cui non sarà possibile ottenere una interrogazione di giacenza (Materiali di consumo, parti immateriali, codici figurativi, ecc...).
Giac. Fiscale	Il flag indica che il Modello-Parte corrente sarà da inserire nei report di giacenza ai fini fiscali.
Gest. A Colore	Questo flag alzato indica che il Modello-Parte è gestito a colori e quindi ogni riferimento al codice corrente dovrà essere completato da un riferimento ad un codice colore.

Col. Stagionale	L'indicazione di stagionalità del colore indica che per ogni stagione sarà obbligatorio inserire l'abilitazione a colori. Il non inserimento di un'abilitazione stagionale farà sì che il colore non potrà essere utilizzato al di fuori della stagione abilitata. Il flag spento fa sì che si possano definire abilitazioni di colori generici validi per tutte le stagioni, a patto che non esistano altre abilitazioni stagionali che, quindi, saranno le uniche valide per la stagione.
Stagione	Stagione di nascita del Modello-Parte. E' un dato statistico che NON interviene nell'abilitàzione stagionale del Modello, che quindi potrà essere comunque utilizzato in più stagioni.
Classe/Sottoclasse	Classificazione statistica del modello, utilizzabile in stampe e selezioni opertive nelle più diverse funzioni per richiamare più modelli appartenenti alla stessa Classe/sottoclasse.

Tabella 10: Parte: Dati base

3.1.6 Listini di Vendita

L'obiettivo della gestione dei Listini di vendita è la definizione dei Prezzi di vendita dei Prodotti e di quant'altro (purchè codificato), oggetto di transazione onerosa nei confronti di Clienti, in funzione di:

- Stagione
- Marchio
- Tipo di listino
- Valuta
- Mercato

ed eventualmente di una specifica

- Linea di vendita **(Business Line)**

nell'ambito del Marchio di appartenenza.
Ci potrebbero essere dei Listini di Vendita importati da sistemi esterni che ne gestiscono il calcolo, la gestione e la sincronizzazione. Tali Listini non possono perciò essere modificati con le procedure qui di seguito descritte. La loro gestione è quindi demandata al sistema che li ha generati.
Di seguito in *figura 7* possiamo vedere un'anagrafica di gestione Stealth 3000 per i Listini di Vendita, con la descrizione dei dati principali.

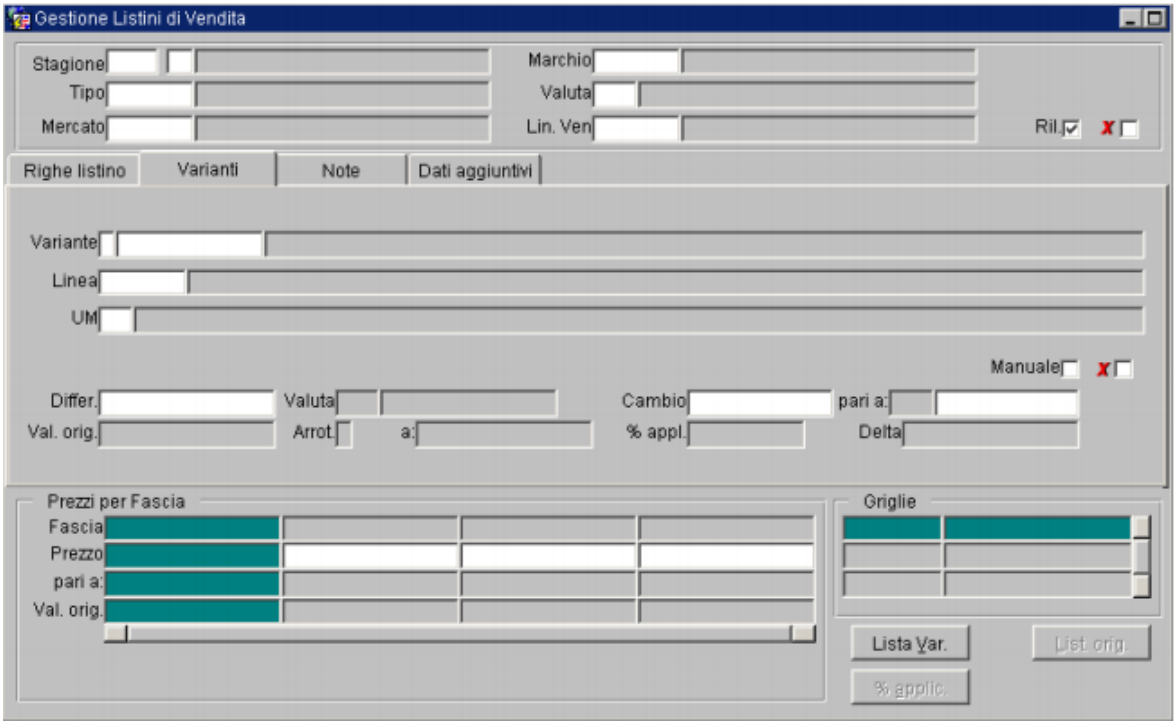

Figura 7: Anagrafica Listino di Vendita

Dati di Testata:

Dato	Descrizione
Marchio	Codice del Marchio di riferimento validato dalla tabella dei Marchi.
Tipo Listino	Codice del Tipo di Listino validato dalla tabella dei Tipi di Listino.
Valuta	Codice della Valuta validata dalla tabella delle Valute.
Mercato	Codice del Mercato di riferimento validato dalla tabella dei Mercati.
Lin.Ven	Non gestita.
Ril.	Flag listino rilasciato. Se attivato indica che il Listino di Vendita è utilizzabile.
X	Flag di annullamento di validità del record corrente. Con questo flag alzato il Listino risponderà ai controlli di relazionalità del database, ma sarà a tutti gli effetti non valido.

Tabella 11: Listino di Vendita: Dati di Testata

Righe di listino

Dato	Descrizione
Mod/Cl.	Codice del modello relativo al prodotto oggetto della riga di listino, oppure codice della classe di modelli, nel caso di gestione di listini per classe e non per singolo prodotto. Nel caso di righe di listino relative a materiali il campo non è valorizzato.
Combinazione	Non utilizzato
Par/Cl	Codice della parte relativa al prodotto oggetto della riga di listino, oppure codice della classe di parti, quando il listino sia gestito per modello/classe di parti o per classe di modelli/classe di parti. Nel caso di righe di listino riferite a materiali, questi possono essere gestiti per singola parte piuttosto che per classe di parti.

Colore	Dato da gestire nel caso di variabili di prezzo per colore. Se immesso, il relativo prezzo vale solo per il colore indicato. Nel caso di un articolo previsto in più colori con prezzo uguale tranne eccezioni, dovranno essere inserite tante righe per colore quante sono le eccezioni più una riga generica (senza colore) per tutte le variabili con prezzo uguale.
Misura	Da utilizzare solo nel caso di materiali gestiti a misura, il cui prezzo vari al variare della misura.
Variante	Tipo Variante, validato dalla tabella delle Varianti.
Drop	Codice Drop, non constituisce variabile di prezzo.
Statura	Codice Statura, non constituisce variabile di prezzo.
Etichetta	Codice Etichetta, non constituisce variabile di prezzo.
Cartellino	Codice Cartellino, non constituisce variabile di prezzo.
UM	Unità di Misura di riferimento dell'oggetto.
Prezzo	Prezzo unitario dell' oggetto riferito alla UM, nella valuta espressa in testata listino.
Valuta	Valuta aziendale.
Cambio	Se il prezzo è espresso in una valuta diversa dall' Euro, il cambio consente di determinare il controvalore in Euro.
Pari a:	Unità di Riferimento del cambio
Val. orig.	Valore originario quando il listino. Viene ottenuto per copia da altro, con applicazione di algoritmi di ricalcolo.
Round	Arrotondamento.
a:	Applicazione dell'arrotondamento.
% appl.	Differenza percentuale rispetto al listino di partenza.
Delta	Delta prezzo in valore assoluto

Tabella 12: Listino di Vendita: Righe di Listino

3.2 Oracle PL/SQL Developer

PL/SQL Developer è un ***IDE*$_g$** creato per sviluppare unità di programma memorizzato in un database Oracle.
SQL nasce come linguaggio per interrogazioni ad un database, che siano di estrazione o modifica dati, ma non permette di manipolare i dati in maniera estensiva, caratteristica invece di un linguaggio procedurale; istruzioni condizionali (IF ELSE) e cicli di iterazione, oltre a creazione di variabili sono le fondamenta alla base di programmi ed algoritmi complessi, ed è questo il vantaggio di PL/SQL.

3.2.1 Scrittura di un programma PL/SQL

PL/SQL permette di creare script principalmente come funzioni, procedure oppure ***package*$_g$** che le contengono. Tutte le unità di programma sono accessibili da altre funzioni all'interno dello stesso database se appartengono allo stesso utente di accesso.
Il codice PL/SQL ha una struttura specifica, organizzata a "blocchi" nel formato:
BEGIN
[content]
exception
[exception handling]
END;
Questo formato viene utilizzato all'interno di procedure e funzioni, e permette di utilizzare i costrutti di base dei linguaggi procedurali, come i cicli (for, while...) e istruzioni condizionali, oltre alla manipolazione delle variabili. Queste sono dichiarabili solo alla definizione di un programma, in una sezione presente appena dopo aver scritto il nome di una funzione o procedura, ma prima del comando **BEGIN**, nel formato
Procedure/Function <Nome procedura o funzione>(parametri) **IS**
[elenco variabili]
BEGIN
[content]
exception
[exception handling]
END Nome procedura o funzione;
Le variabili sono visibili solo all'interno del blocco in cui sono dichiarate, in particolare le variabili dichiarate internamente ad una funzione sono visibili solo all'interno di essa, mentre le variabili definite all'interno del package sono visibili a tutte le funzioni contenute in esso, e vengono definite **globali**.

Oltre alle variabili standard tipiche di altri linguaggi procedurali, una delle caratteristiche sicuramente più utili di PL/SQL è quella di poter creare dei **cursor**; questi sono una dichiarazione di una query di selezione che poi potrà essere eseguita all'interno del programma ed il suo contenuto analizzato, tramite in comando **fetch**, il quale estrae una riga ed in successione le altre ogni volta che viene chiamato. Il contenuto del cursore può essere estratto in n variabili a seconda delle colonne generate dalla query, oppure in una variabile di tipo *nome_ cursore*%**rowtype** dalla quale è possibile estrarre ogni campo della riga del cursore scrivendo *nome_ rowtype.nome_ campo*.
Una particolare forma di cursor sono i **refcursor** che permettono di trasformare una stringa di testo contenente una query, in una query vera e propria. L'utilità principale è quella di poter creare una query parametrica, nel senso che intere condizioni di filtro oppure dati da voler estrarre a seconda della richiesta, possibillità non concessa da una query standard. Una cosa a cui fare attenzione però è che finché si è in *compile-time* non sarà altro che una stringa di test qualsiasi, ed eventuali errori nella stesura della query vengono scoperti soltanto in *Run-time*, ovvero all'esecuzione del programma.
L'accesso alle funzioni di un package avviene tramite una definizione dello stesso chiamata Package Specification, ovvero la parte del package accessibile pubblicamente da qualunque altro package o funzione nel database a cui l'utente abbia accesso.
Ci sono due tipologie di package, dal punto di vista logico:

- **Package di utility**: essi hanno nella loro *package specification* il riferimento a tutte le funzioni implementate nel package, dovendo esse venire richiamate singolarmente a seconda dell'utilità, ad esempio ***MyPackage.GetUserName()*** e ***MyPackage.GetUserAge()***;
- **Package di funzione**: questi hanno solitamente nella *package specification* solo una funzione di 'entrata', la quale richiama al suo interno, nell'implementazione del package, tutte le varie funzioni definite privatamente, ad esempio ***MyPackage.GetUserLog()*** può essere una funzione pubblica, la quale richiama nella definizione le funzioni private ***MyPackage.GetUserAccessId()*** e ***MyPackage.GetUserAction()***, non accessibili dall'esterno per eventuali architetture stabilite.

3.2.2 Debug di un programma

Una funzionalità caratteristica e particolarmente utile di PL/SQL è quella di poter eseguire il $\boldsymbol{debug_g}$ di una funzione o di un intero package.
Questa funzione permette, come la maggior parte dei compilatori nei linguaggi

di programmazione, di analizzare l'esecuzione del programma passo per passo, con l'obiettivo di individuare criticità in determinati punti del codice in **Run-Time**. Diventa particolarmente utile nel caso si abbia a che fare con i refcursor spiegati in precedenza, ma ovviamente la funzionalità è importante in ogni parte del programma ed è parte fondamentale della verifica del codice nella sua interezza. L'entrata in modalità di debug decompila il package, per cui bisogna prestare molta attenzione nell'utilizzo della funzione, in quanto un package che utilizza una funzione che viene decompilata per debug, andrà in errore nella sua esecuzione fino a che non viene ricompilato il sottoprogramma in stato di debug.

3.2.3 Piano di esecuzione

Una funzionalità caratteristica e particolarmente utile di PL/SQL è quella di poter visualizzare il piano di esecuzione di una query per poterne analizzare i punti critici. Nella *figura 8* è rappresentato il piano di esecuzione di una query del programma di export. Il piano di esecuzione è una funzionalità che permette di vedere l'ordine e le modalità di accesso alle tabelle da parte di PL/SQL Developer, in particolare che tipo di indici utilizza per la giunzione delle tabelle. Con riferimento alla (figura)

Optimizer goal: All rows

Description	Object owner	Object name	Cost	Cardinality
SELECT STATEMENT, GOAL = ALL_ROWS			157	2
NESTED LOOPS			157	2
NESTED LOOPS			157	2
NESTED LOOPS			153	2
NESTED LOOPS			149	2
NESTED LOOPS			145	2
NESTED LOOPS			110	1
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_LTS	107	3
INDEX RANGE SCAN	ST3K	S3T_LTS_UK_KEYLIS	105	3
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_LST	1	1
INDEX UNIQUE SCAN	ST3K	S3T_LST_PK	0	1
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_LDE	36	2
INDEX RANGE SCAN	ST3K	S3T_LDE_UK_KEYLIS	15	28
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_OGG	2	1
INDEX RANGE SCAN	ST3K	S3T_OGG_IK_MODPAR	1	1
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_OGG	2	1
INDEX UNIQUE SCAN	ST3K	S3T_OGG_PK	1	1
INDEX UNIQUE SCAN	ST3K	S3T_OGG_PK	1	1
TABLE ACCESS BY INDEX ROWID	ST3K	S3T_OGG	2	1

Figura 8: Piano di esecuzione

possiamo prendere in analisi alcune delle funzionalità del piano di esecizione. Le

modalità di giunzione tra le tabelle vengono specificate dal tipo di indice utilizzato; i vari tipi di indice sono:

- **Index Unique Scan**: Recupera un singolo id di riga (*rowid*) dall'indice;
- **Index Range Scan**: recupera uno o più id di riga, in ordine crescente;
- **Index Full Scan**: Recupera tutti gli id di riga dall'indice, in ordine crescente;
- **Skip Scan**: Recupera gli id di riga da un indice usato come concatenazione di campi, senza usare le colonne effettive

Il piano di esecuzione permette inoltre di vedere come cambiano le prestazioni, in termini di costo computazionale, al cambio di *hint*, ovvero suggerimento all'ottimizzatore, specificato nella query con la dicitura /*+ nome_hint */ subito dopo la keyword Select, nel casi si tratti di una query di selezione. I tipi di hint sono:

- **All Rows**: esplicita richiesta di utilizzare il costo computazionale più basso possibile
- **First Rows(n)**: esplicita richiesta di utilizzare il costo computazionale più basso possibile per il numero n di righe. L'ottimizzatore ignora questo hint nelle query di *Delete*, *Update* e nelle query di *Select* contenenti la seguente sintassi: operatore *Distinct*, funzioni aggregate, clausole *Group By*, *Order By*, *For Update* e operatori di **Set** (*Union*, *Intersect*, *Minus* ecc.)
- **Rule**: esplicita di disabilitare l'uso dell'ottimizzatore. Nella maggior parte dei casi è sconsigliato l'uso.

3.3 Oracle Reports Builder

Uno degli strumenti utilizzati dall'azienda appartenente alla suit Oracle, è Oracle Reports Builder, che permette di disegnare dei report e fare in modo che i dati estratti derivino dal database di riferimento.
Possono essere utilizzate tutte le funzioni sviluppate in quel database da PL/SQL e ciò permette una diretta relazione tra gli strumenti.
Il software permette di inserire una o più query di selezione come modello di partenza *figura 9* e mostra in real time il risultato all'interno del report costruito tramite l'editor apposito *figura 10*

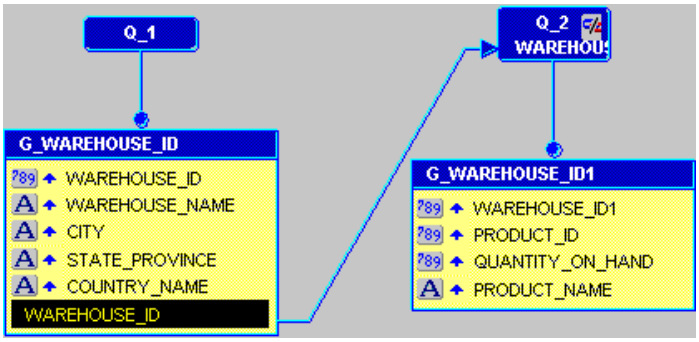

Figura 9: Oracle Reports Builder: Modello da Query

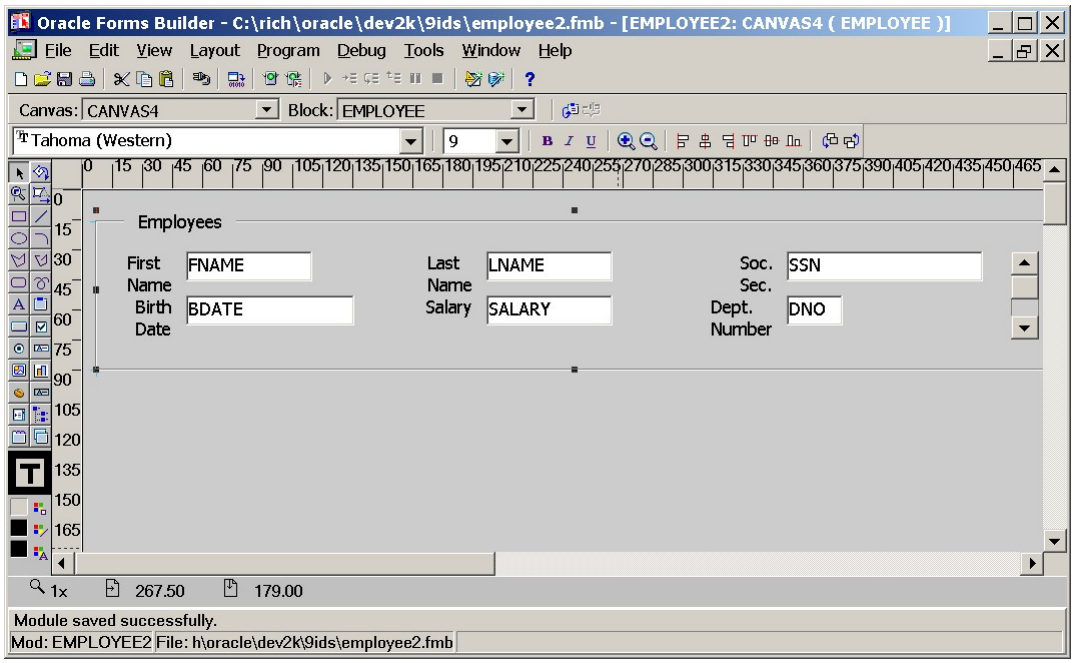

Figura 10: Oracle Reports Builder: Editor

Viene data la possibilità di scrivere funzioni e *trigger* come unità di programma in PL/SQL, richiamabili in ogni momento, il che permette di manipolare i dati estratti dalle query di modello.

3.4 Tecnologia per l'aggiornamento

I Database Link sono uno strumento messo a disposizione da Oracle per la comunicazione unidirezionale verso un altro database server, sotto forma di puntatori salvati come record all'interno di una **Data Dictionary Table**, ovvero tabelle di

sola lettura che forniscono informazioni sul database. La comunicazione è unidirezionale nel senso che se un database Oracle possiede un dblink verso un altro database, non significa che dal database di destinazione si possa accedere al database Oracle.
Il vantaggio principale nell'utilizzo dei Database Links, è che permette l'accesso ad un utente ad un database remoto i cui oggetti hanno privilegi impostati dal proprietario.
Perché una connessione abbia via dblink abbia successo, è necessario che ogni database abbia un **Global Database Name** all'interno del dominio di rete. Tipicamente, un Database Link ha lo stesso nome del Globa Database Name del database remoto a cui fa riferimento.
I tipi di Database Link sono i seguenti:

- **Privato**: Crea un link in uno specifico scgema del database locale. Solo il proprietario del database link può utilizzarlo per accedere al database remoto.
- **Publico**: Crea un link valido per tutti il databse. Tutti gli utenti e programmi PL/SQL del database possono usare il dblink per accedere agli oggetti del database remoto.
- **Globale**: Crea un link valido in tutta la rete, accessibile a tutti gli utenti e programmi PL/SQL di tutti i database

Una volta creato il databse Link si è soliti creare un sinonimo utilizzato per riferirsi a tale link, nascondendo il nome del database.
Tramite Database Link non è permesso:

- pemettere i privilegi agli oggetti remoti (*grant*);
- eseguire operazioni di *Describe*, ovvero una visualizzazione della struttura degli oggetti di riferimento;
- Definire ruoli ad utenti in un database remoto;
- Assegnarsi ruoli in un database remoto;
- eseguire join tra database che utilizzano connessioni server condivise.

4 Sviluppo

4.1 Sviluppo programma di export

4.1.1 Architettura della solizione

La versione iniziale del programma di export è stata sviluppata nel 2007 da uno dei programmatori attualmente presenti in azienda e si basa sulla creazione di file di testo che vengono inviati al fornitore, perché si occupi di caricare i database degli showroom della campagna vendite.
Dato che la decisione per l'aggiornamento è stata quella di utilizzare dei DbLink, sotto consiglio del tutor aziendale il primo passo è stato creare una copia esatta delle tabelle dei database di destinazione, all'interno del database di sviluppo, sul quale è stato sviluppato il prototipo. Il motivo della copia è che al momento della scrittura remota tramite dblink sarebbe stato molto più semplice fare un riversamento del contenuto di una tabella all'interno di un'altra, ed in più in questo modo si ha una versione di backup locale dei dati, in modo che se ci dovessero essere errori nel trasferimento, si possa correggere facilmente ogni problematica.
Inoltre in un sistema Eterogeneo (ovvero collegamento fra database server di tipo diverso, Oracle to Sql Server) non si possono eseguire manipolazioni dei dati all'interno delle query di inserimento, come ad esempio conversioni o formattazioni, che vengono quindi anticipate alla fase di caricamento nella parte locale.
Una volta create tutte le tabelle locali, il passo successivo è stato prendere spunto dalle query del programma esistente ed ottenere tutti i dati necessari a ricreare le nuove query, con alcune modifiche proposte dai service manager, per popolarle.
L'esecuzione del programma prevede anche degli input che talvolta possono essere facoltativi, ma la loro presenza va considerata all'interno delle query e ciò comporta l'utilizzo dei **refcursor** per permettere una parametrizzazione della query.
Durante il popolamento della tabella locale, viene valorizzato il campo di log 'Data_modifica' che verrà in seguito utilizzato per capire quali dati riversare nel database remoto, confrontandola con la data impostata all'inizio di esecuzione del programma.

4.1.2 Codice

Di seguito vediamo la porzione di codice che mostra il percorso di estrazione dati, popolamento della tabella locale ed infine popolamento della tabella remota relativa ai Modelli.

Il primo passo è la definizione di una stringa di testo che in Run-Time sarà query eseguibile, contente l'interrogazione esatta necessara:

```
v_query :=
'Select Distinct
    substr(m.ogg_cod,6,5)   cd_modello,'      || chr(10)
       ||
'   substr(m.ogg_cod,11,2)   cd_varia,'       || chr(10)
   ||
'   ''C''  TipoAna,'                          || chr(10)
   ||
'   substr(m.ogg_cod,1,5)  cd_Ana,'           || chr(10)
   ||
'    m.ogg_des Descrizion,'                   || chr(10)
   ||
'   ''STD''  cd_defmod,'                      || chr(10)
   ||
'   m.ogg_stg_anno||
    Substr(m.ogg_stg_cod,1,1) cd_stagion,'    || chr(10)
       ||
'   Substr(ms.ogg_soc_lin_cod,1,3) cd_lin ,' || chr(10)
   ||
'   tg.tgl_ogg_grt_cod cd_Taglia,'            || chr(10)
   ||
'   null cd_Flash,'                           || chr(10)
   ||
'   null cd_Colle,'                           || chr(10)
   ||
'   0    qtaconf,'                            || chr(10)
   ||
'   m.ogg_cod   cd_modello_ex,'               || chr(10)
   ||
'   nvl(m.ogg_f_annu,''0'') mod_flannu'       || chr(10)
   ||
'  From s3t_ogg_soc ms,
           s3t_tgl_ogg tg,
         s3t_ogg m' || chr(10) ||
' Where ms.ogg_soc_soc_cod(+)=
   s3ksysutils.SocPubPriv(''''||s3ksysglobal.Soc_Ute||''''
      ,
                           ''OGG_SOC'')' || chr(10)
```

```
                                    ||
'And ms.ogg_soc_ogg_soc_cod(+)=
            m.ogg_soc_cod' || chr(10) ||
'And ms.ogg_soc_ogg_id(+)=
            m.ogg_id' || chr(10) ||
'And tg.tgl_ogg_ogg_soc_cod(+)=
            m.ogg_soc_cod' || chr(10) ||
'And tg.tgl_ogg_ogg_id(+)=
            m.ogg_id' || chr(10) ||
'And m.ogg_tipo=''1''' || chr(10) ||
'AND (to_char(m.ogg_data_mod,''YYYY/MM/DD HH24:MI:SS'')
   >
''''||to_char(LastExcutionData,'YYYY/MM/DD HH24:MI:SS')
||''''||
        chr(10) ||
'         OR' || chr(10) ||
'to_char(ms.ogg_soc_data_mod,''YYYY/MM/DD HH24:MI:SS'')
   >
''''||to_char(LastExcutionData,'YYYY/MM/DD HH24:MI:SS')

'         OR' || chr(10) ||
'to_char(tg.tgl_ogg_data_mod,''YYYY/MM/DD HH24:MI:SS'')
   >
''''||to_char(LastExcutionData,'YYYY/MM/DD HH24:MI:SS')
||''')'||
        chr(10) ||
'And m.ogg_stg_anno||''/''||m.ogg_stg_cod =
       ''''||P_STG_ATT||'''';

IF v_mrc_lis IS NOT NULL THEN
  v_query := v_query ||'
             And Substr(ms.ogg_soc_lin_cod,1,2)
                 in (''''||v_mrc_lis||''')  ORDER BY 1';
END IF;
IF v_lnv IS NOT NULL THEN
  v_query := v_query ||
             ' And ms.ogg_soc_lin_cod
                  in (''''||v_lnv||''')';
END IF;
```

La sintassi di PL/SQL prevede che la concatenazione fra stringhe di testo (delimitate da apici) avvenga tramite il 'pipe' due volte in successione, inoltre l'inserimento di una variabile all'interno della stringa va gestito con cautela, in quanto va fatta una distinzione sul tipo della variabile inserita:

- se è di tipo alfanumerico vanno utilizzati 3 apici in chiusura della stringa, seguiti dalla variabile ed a sua volta seguita da altri 3 apici, ogni parte concatenata con l'altra; questo perché ci deve essere una forma di *escape* tra gli apici che distinguono il testo della query in stato di stringa ed il contenuto della variabile, che in Run-Time, quando viene effettivamente eseguita la query, diventa un valore alfanumerico senza significato per un compilatore, non utilizzabile in un confronto o una selezione.

- se è di tpo intero basta una concatenazione senza apici aggiuntivi

La fase successiva all'estrazione dei dati è quella di caricarli nella tabella locale dei Modelli.

```
open Cur_mod for v_query;
  LOOP
    fetch Cur_mod into cd_modello_mod,
                       cd_varia_mod,
                       TipoAna_mod,
                       cd_Ana_mod,
                       Descrizion_mod,
                       cd_defmod_mod,
                       cd_stagion_mod,
                       cd_lin_mod,
                       cd_taglia_mod,
                       cd_flash_mod,
                       cd_Colle_mod,
                       qtaconf_mod,
                       cd_modello_ex_mod,
                       mod_flannu_mod;
    exit when Cur_mod%notfound;
    BEGIN
      INSERT INTO PINDT_MOD_CRM(cd_modello,
                                cd_varia,
                                tipoana,
                                cd_ana,
                                descrizion,
```

```
                                cd_defmod,
                                cd_stagion,
                                cd_linea,
                                cd_taglia,
                                cd_flash,
                                datains,
                                dataupd,
                                annullato,
                                qtaconf,
                                cd_modello_ex,
                                societa
                                )
                           VALUES(Cd_Modello_mod,
                                Cd_Varia_mod,
                                TipoAna_mod,
                                Cd_Ana_mod,
                                Descrizion_mod,
                                cd_defmod_mod,
                                Cd_Stagion_mod,
                                Cd_Lin_mod,
                                Cd_Taglia_mod,
                                Cd_Flash_mod,
                                LastExcutionData,
                                v_sysd,
                                Mod_Flannu_mod,
                                Qtaconf_mod,
                                Cd_Modello_Ex_mod,
                                s3ksysglobal.Soc_Ute
                                );
    exception
      when dup_val_on_index then
        BEGIN
          Update Pindt_Mod_Crm t
             Set t.Descrizion     = Descrizion_mod,
                 t.cd_defmod      = Cd_Defmod_mod,
                 t.cd_stagion     = Cd_Stagion_mod,
                 t.cd_linea       = Cd_Lin_mod,
                 t.cd_flash       = Cd_Flash_mod,
                 t.dataupd        = v_sysd,
                 t.annullato      = Mod_Flannu_mod,
```

```
                t.qtaconf         = Qtaconf_mod,
                t.cd_modello_ex   = Cd_Modello_Ex_mod
          Where t.cd_modello      = Cd_Modello_mod
            and t.cd_varia        = Cd_Varia_mod
            and t.tipoana         = Tipoana_mod
            and t.cd_ana          = Cd_Ana_mod
            and t.societa         = s3ksysglobal.Soc_Ute;
      exception when others then
          v_error := sqlerrm;
          s3ksysmess.Batch_Messaggi(60070,sysdate,
      'MODELLI:'||nvl(Cd_Modello_mod,'codice null'));
      END;
    when others then
      v_error := sqlerrm;
      s3ksysmess.Batch_Messaggi(60070,sysdate,
      'MODELLI:'||nvl(Cd_Modello_mod,'codice null'));
  END;
  commit;
END LOOP;
close Cur_mod;
```

In questo blocco di codice possiamo vedere un'applicazione dei *refcursor*, il cui contenuto viene iterato per ogni record estratto dalla query definita sopra, ed ogni riga viene inserita nella tabella locale dei Mmodelli.
Si può inoltre vedere una gestione degli errori, permessa dalla struttura **BEGIN/exception/END** di PLSQL, in cui un errore di tipo 'chiave logica duplicata' viene gestito con un ulteriore blocco in cui viene aggiornato il valore del record per la chiave estratta. Eventuali errori generici vengono gestiti con un sistema di messaggistica sottoforma di *log*, messo a disposizione da Stealth.
All'inizio della funzione che contiene i due blocchi di codice precedenti, viene assegnato il valore della data attuale alla variabile *v_sysd* grazie alla keyword **sysdate**, e quest'ultima sarà il filtro per decidere quali valori della tabella locale verranno trasferiti nel database remoto, come si vede nel seguente blocco di codice finale:

```
BEGIN
  FOR CUR_MOD IN (
              SELECT *
                FROM PINDT_MOD_CRM t
               WHERE t.dataupd = v_sysd
                 and t.societa = s3ksysglobal.Soc_Ute
                 AND t.cd_stagion =
```

```
                    replace(P_STG_ATT,'/'))
  LOOP
    BEGIN
      v_qtaconf := to_number(Cur_mod.Qtaconf);
      INSERT INTO modelli@crm_sydat_eur.industries.com(
                  "Cd_modello",
                  "Cd_Varia",
                  "TipoAna",
                  "Cd_Ana",
                  "Descrizion",
                  "Cd_Defmod",
                  "Cd_Stagion",
                  "Cd_linea",
                  "Cd_Taglia",
                  "Cd_Flash",
                  "Cd_Colle",
                  "QtaConf",
                  "DataUpd",
                  "DataIns",
                  "Annullato",
                  "cd_modello_ex"
                  )
           VALUES(Cur_mod.Cd_Modello,
                  Cur_mod.Cd_Varia,
                  Cur_mod.Tipoana,
                  Cur_mod.Cd_Ana,
                  Cur_mod.Descrizion,
                  Cur_mod.Cd_Defmod,
                  Cur_mod.Cd_Stagion,
                  Cur_mod.Cd_Linea,
                  Cur_mod.Cd_Taglia,
                  Cur_mod.Cd_Flash,
                  Cur_mod.Cd_Colle,
                  v_qtaconf,
                  v_date,
                  v_date,
                  Cur_mod.Annullato,
                  Cur_mod.Cd_Modello_Ex
                  );
       commit;
```

```
      v_modelli_ins := v_modelli_ins + 1;
   exception
    when Dup_insert then
      BEGIN

       UPDATE modelli@crm_sydat_eur.industries.com
          SET "Descrizion" = Cur_mod.Descrizion,
              "Cd_Defmod" = Cur_mod.Cd_Defmod,
              "Cd_Stagion" = Cur_mod.Cd_Stagion,
              "Cd_linea" = Cur_mod.Cd_Linea,
              "Cd_Taglia" = Cur_mod.Cd_Taglia,
              "Cd_Flash" = Cur_mod.Cd_Flash,
              "Cd_Colle" = Cur_mod.Cd_Colle,
              "QtaConf" = Cur_mod.Qtaconf,
              "DataUpd"       = v_date,
              "Annullato"     = Cur_mod.Annullato,
              "cd_modello_ex" = Cur_mod.Cd_Modello_Ex
        WHERE "Cd_modello" = Cur_mod.Cd_Modello
          AND "Cd_Varia" = Cur_mod.Cd_Varia
          AND "TipoAna" = Cur_mod.Tipoana
          AND "Cd_Ana" = Cur_mod.Cd_Ana;
           commit;

          v_modelli_upd := v_modelli_upd + 1;
       exception when others then
         v_error := sqlerrm;
         s3ksysmess.Batch_Messaggi(60071,sysdate,'
            Modelli'||sqlerrm);
         v_modelli_err := v_modelli_err + 1;
         rollback;
       END;
     when others then
       v_error := sqlerrm;
       s3ksysmess.Batch_Messaggi(60071 ,sysdate,'
          Modelli'||sqlerrm);
       v_modelli_err := v_modelli_err + 1;
       rollback;
  END;
END LOOP;
exception when others then
```

```
        s3ksysmess.Batch_Messaggi(1583,sysdate,'Modelli: '
            ,sqlerrm);
    END;
```

In un sistema Eterogeneo, ovvero in cui i database collegati dal DbLink sono diversi, come in questo caso tra Oracle e Sql Server, il riferimento ai campi dati di una tabella remota vanno specificati utilizzando il doppio apice ad inizio e fine, ed il nome è *case sensitive*.
I dati estratti dal cursore per essere riversati nella tabella remota sono filtrati per la data impostata precedentemente, in fase di caricamento della tabella locale. Quanto emerso da elaborazioni su set ristretti di dati, i DbLink in un sistema eterogeneo sono piuttosto lenti, nel caso specifico vengono trasferiti circa 1200 record al secondo in inserimento, mentre circa la metà in fase di modifica data la presenza di condizioni di filtro (nella clausola WHERE) che necessariamente rallentano l'esecuzione. Inoltre la decisione di eseguire una **commit** ad ogni record, utile per avere dei dati in fase di esecuzione in caso il programma sia molto lungo nella sua esecuzione, rallenta il processo, rispetto ad avere una singola commit alla fine dell'esecuzione, al costo ovviamente di non aver inserito nessuna riga in caso di un qualsiasi errore.
Nel complesso, l'autonomia dell'esecuzione, e soprattutto la decisione di schedulare il programma ogni notte per rendere marginale la questione della velocità di esecuzione, permettono di avere un vantaggio rispetto alla versione attuale del programma.

4.2 Sviluppo programma di import

4.2.1 Architettura della soluzione

Il programma di import tratta gli ordini che vengono creati dai clienti che acquistano gli articoli durante la campagna vendite. Questi vengono poi importati direttamente nelle tabelle del database a cui fa riferimento Stealth, tramite un processo di caricamento di tabelle intermedie create specificatamente per i processi di import/export. Di conseguenza il programma di import in realtà si occupa solo di caricare le tabelle intermedie (che hanno lo stesso nome delle tabelle effettive, ma con l'aggiunta alla fine 'IEX') e di invocare la funzione standard di stealth di import per il caricamento definitivo.
Prima di essere direttamente caricati nelle tabelle IEX, i dati vengono caricati dal database remoto a quello Oracle in alcune tabelle temporanee, ovvero che alla creazione venogno specificate come tali ed il loro contenuto esiste solo per la ses-

sione corrente, quindi al di fuori dell'esecuzione saranno vuote.
Ogni procedura eseguita da stealth ha un identificativo di richiesta, il quale a sua volta crea uno o più identificativi di lancio del programma, in base a quante volte viene lanciata l'esecuzione. Si è quindi deciso di creare un campo 'IdLancio' nelle tabelle degli ordini del database remoto, che avrebbe avuto valore nullo nel momento dell'inserimento dei dati nuovi da parte del programma che crea ordini negli showroom, ed avrebbe permesso quindi di capire quali fossero gli ordini da importare. Vengono quindi marchiati i record con id di lancio nullo per l'esecuzione dell'import, e riannullati in caso di errori, in maniera tale da essere consideratii nel lancio successivo.
Per motivi di documentazione aziendale, viene infine aggiornato ogni record inserito con successo nelle tabelle del database remoto, valorizzando il campo 'NOrdAZ' delle tabelle remote di testate e righe.

4.2.2 Codice

Di seguito si può vedere la funzione principale che gestisce il flusso dei dati, ed a sua volta chiama le varie funzioni necessarie per estrarre e definitivamente inserire i dati nel database di Stealth, oltre a notificare al database remoto i record caricati con successo:

```
procedure elabora is
--Cursori dati dalle tabelle temporanee
cursor ctes is
  select *
    from pindtto_tesord_crm
order by cd_stagion, codage;

cursor crig (
            p_oct_annorif in number,
            p_oct_codsta  in varchar2,
            p_oct_codage  in varchar2,
            p_oct_dtord   in date,
            p_oct_numord  in varchar2) is
  select *
    from pindtto_rigord_crm
   where cd_stagion = p_oct_annorif||
                      p_oct_codsta
     and codage     = p_oct_codage
     and dtord      = p_oct_dtord
     and numord     = p_oct_numord
```

```
order by numrig;

rcode      boolean := true;

BEGIN

--Import dati in tabelle temporanee
  if P_SOC_UTE = '01' then
    b_import := import_crm_tables_eur;
  elsif P_SOC_UTE = '45' then
    b_import := import_crm_tables_jap;
  elsif P_SOC_UTE = '48' then
    b_import := import_crm_tables_usa;
  end if;

  if b_import then

    open ctes;
    loop
      fetch ctes into roct;
      exit when ctes%notfound;

      v_agente   := roct.codage;
      v_stg_anno := to_number(substr(roct.cd_stagion
         ,1,4));
      v_stg_cod  := (substr(roct.cd_stagion,5,1));
      --Identificativo di testata
      begin
        select s3sq_oct_iex.nextval into voct_id from
           dual;
      exception
      when others then
        s3ksysmess.batch_messaggi(11500, sysdate, '
           S3SQ_OCT_IEX');
      end;

      -- Inserimento TESTATA
      rcode := ins_testata(roct, voct_id);

      -- Inserimento RIGHE e TAGLIE
```

```
      vocr_riga:=0;

      open crig(substr(roct.cd_stagion,1,4),
                substr(roct.cd_stagion,5,1),
                v_agente,
                roct.dtord,
                roct.numord);
      loop
        fetch crig into rocr;
        exit when crig%notfound;

        vocr_riga := vocr_riga + 1;
        rcode     := ins_riga(roct, rocr, voct_id, roct.
           dtacons);

      end loop;
      close crig;

    end loop;
    close ctes;

    commit;

    -- Richiamo Import per gli ordini caricati nelle IEX
    if b_importa then
      lancio_import;
    end if;

    if P_SOC_UTE = '01' then
      aggiorna_tab_remote_eur;
    elsif P_SOC_UTE = '45' then
      aggiorna_tab_remote_jap;
    elsif P_SOC_UTE = '48' then
      aggiorna_tab_remote_usa;
    end if;

  end if;

end elabora;
```

4.3 Creazione report di import

I report generati dal programma di import si basano su dati presenti nel database, per cui l'aggiornamento del progetto non ne comporta importanti modifiche; è comunque stato richiesto un minimo intervento in ottica migliorativa, in particolare la query che genera il modello è stata aggiornata in *figura 11* per ottenere i dati dalla tabella temporanea in cui vengono inizialmente caricati i dati. Va precisato che l'esecuzione del programma che genera i Report da Oracle Reports Builder è **sincrono**, per cui la sessione del lancio di Stealth è ancora aperta ed attende la generazione dei documenti; se si fosse deciso di rendere il lancio dei report asincrono, si avrebbe avuto un leggero miglioramento nelle prestazioni, non dovendo far attendere il programma, ma si sarebbe dovuta cambiare l'architettura standard poiché la sessione in cui la tabella temporanea veniva caricata, sarebbe terminata prima che la query che genera il modello di Oracle Reports Builder venisse eseguita.

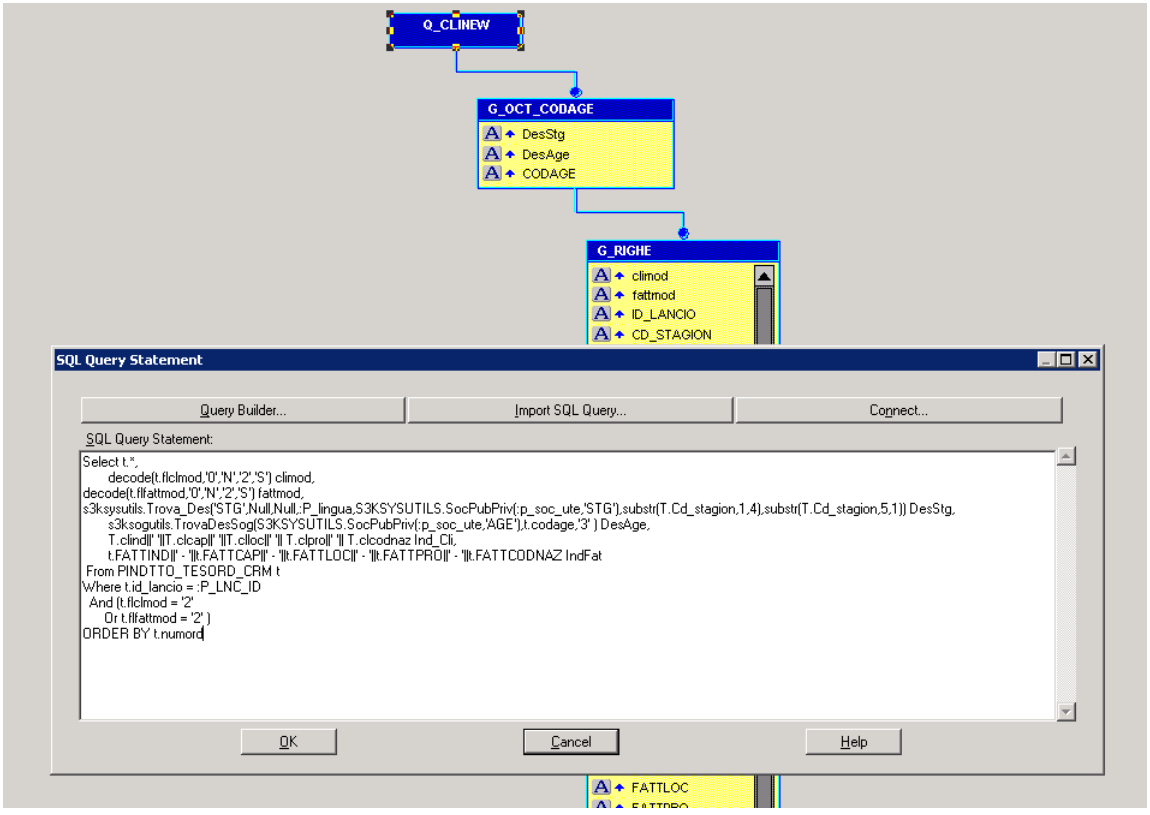

Figura 11: Oracle Reports Builder: Modello da Query

Sono state inoltre modificate alcune delle program unit ed aggiunte altre, visibili in *figura 12* per rispondere ad alcune richieste dei Service Manager.

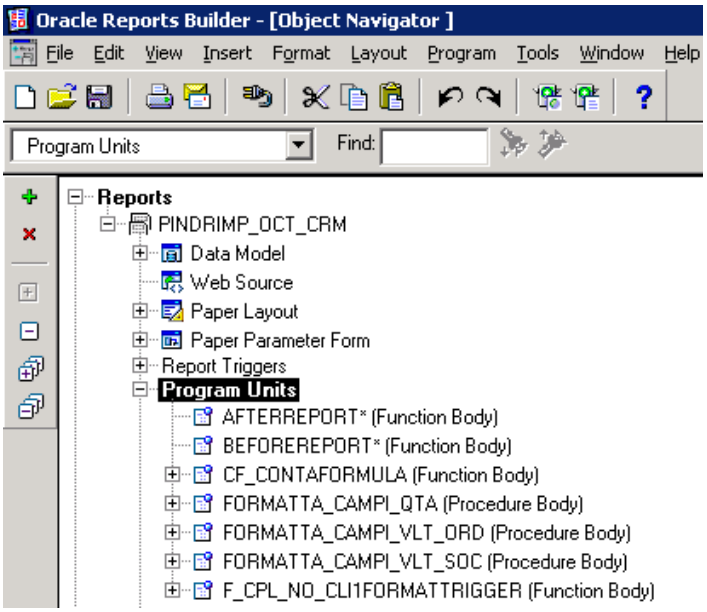

Figura 12: Oracle Reports Builder: Program Unit

In generale l'attività ha richiesto poco tempo avendo il layout in *figura 13* valido dalla versione attuale, così come il contenuto di alcune program unit.

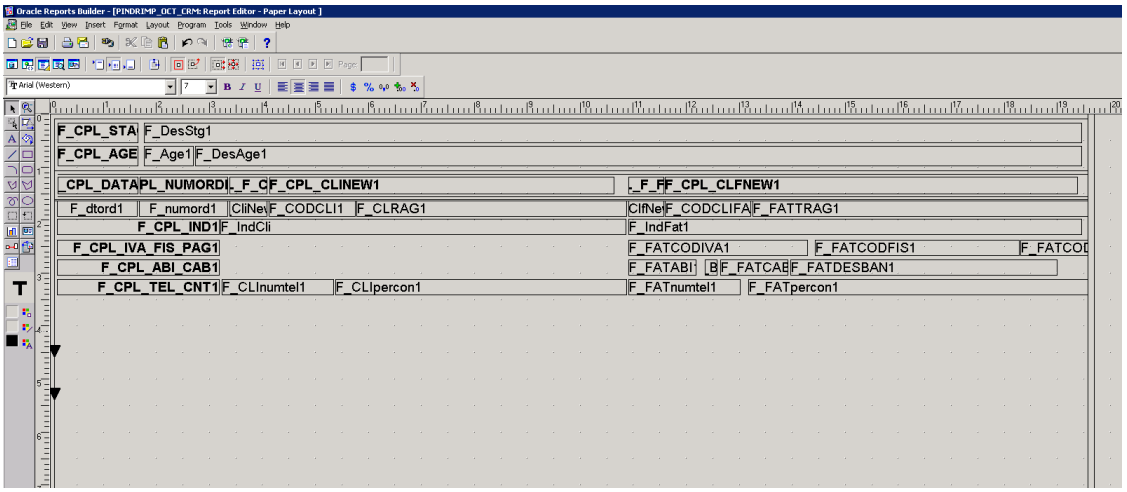

Figura 13: Oracle Reports Builder: Layout Editor

5 Valutazione Retrospettiva

5.1 Obiettivi raggiunti

Il prodotto generato alla fine dello stage è stato tale da poter essere utilizzato con minime modifiche nella prossima campagna vendite, rispettando quindi quanto desiderato dalla proposta iniziale. L'applicazione creata non è comunque particolarmente veloce data la natura lenta dei DbLink, ma nel complesso i Service Manager sono stati soddisfatti del risultato. Sono state inoltre create documentazioni esaustive a supporto del prototipo in caso venga effettivamente deciso di implementare la soluzione anche senza la mia presenza nell'azienda.

5.2 Difficoltà incontrate

La maggior difficoltà incontrata è stata relativa allo studio delle logiche di immagazzinamento dei dati nel database, nello specifico quelli relativi a soggetti ed oggetti, essendo informazioni specifiche del mondo della moda, che di conseguenza non vengono spiegate a livello accademico e ci sono minime informazioni di pubblico dominio a riguardo.
Per quanto riguarda le sfide tecnologiche, ambientarsi con l'ambiente di PL/SQL che non avevo mai visto è stata un esperienza di grosso impatto ma con il tempo si fa velocemente l'abitudine in particolare perché tutto ciò che viene offerto, con cui non avevo familiarità, si rivela essere particolarmente utile.

5.3 Bilancio Formativo

Nel complesso l'esperienza è stata estremamente positiva a livello personale, dato l'ambiente di lavoro a livello umano e professionale. Ho avuto la possibilità di lavorare ad un progetto che potenzialmente ha un'utilità materiale per l'azienda e nel farlo ho potuto vedere il mondo che sta dietro ad un'azienda di prestigio mondiale.
A livello tecnologico le competenze sono molto verticali, essendo richieste quasi esclusivamente nel mercato dell'informatica applicata all'industria della moda, ma lo studio di ciò che viene offerto da PL/SQL, pur essendo un editor che permette di scrivere codice specifico al mondo dei database a differenza di linguaggi di programmazione ad oggetti, è certamente una competenza molto importante che sono soddisfatto di aver sperimentato.

Il supporto degli studi universitari è stato importante grazie ai corsi di Basi di Dati, che mi ha dato competenze tali da permettermi di ambientarmi velocemente ad un ambiente mai visto prima, ed al corso di Ingegneria del Software per aver trasmesso la mentalità necessaria ad approcciare un progetto di grosse dimensioni, che richiede pianificazione e creazione di documentazione.

Glossario

D

database

debug

demo

I

IDE

P

package

R

record

S

Service Manager