



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

**Azienda ospitante:**  
INDUSTRIES SPA

**Relatore:**  
Mauro Conti

**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 effettuare 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 verranno 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

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	L'Azienda . . . . .	1
1.2	Obiettivi di stage . . . . .	1
1.3	Principali problematiche . . . . .	2
<b>2</b>	<b>Analisi e Pianificazione</b>	<b>3</b>
2.1	Analisi del contesto . . . . .	3
2.2	Piano di Lavoro . . . . .	3
<b>3</b>	<b>Studio delle Tecnologie</b>	<b>4</b>
3.1	Stealth 3000 . . . . .	4
3.1.1	Soggetto . . . . .	4
3.1.2	Condizioni di Vendita . . . . .	8
3.1.3	Modelli . . . . .	10
3.1.4	Parti . . . . .	12
3.1.5	Modelli-Parte . . . . .	12
3.1.6	Listini di Vendita . . . . .	12
3.2	Oracle PL/SQL Developer . . . . .	13
3.2.1	Scrittura di un programma PL/SQL . . . . .	13
3.2.2	Debug di un programma . . . . .	14
3.2.3	Piano di esecuzione . . . . .	14
3.3	Oracle Reports . . . . .	14
3.4	Tecnologia per l'aggiornamento . . . . .	15
<b>4</b>	<b>Sviluppo</b>	<b>16</b>
4.1	Sviluppo programma di export . . . . .	16
4.2	Sviluppo programma di import . . . . .	28
4.3	Creazione report di import . . . . .	28
<b>5</b>	<b>Valutazione Retrospettiva</b>	<b>29</b>
5.1	Obiettivi raggiunti . . . . .	29
5.2	Difficoltà incontrate . . . . .	29
5.3	Bilancio Formativo . . . . .	29

## Elenco delle figure

1	Anagrafica Soggetti . . . . .	5
---	-------------------------------	---

## Elenco delle tabelle

1	Soggetti: Dati di testata . . . . .	6
2	Soggetti: Dati Generali . . . . .	8
3	Condizioni di Vendita: Dati Generali . . . . .	10
4	Modello: Dati di Testata . . . . .	11
5	Modello: Dati base . . . . .	12

# 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, 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 era quella di ridurre la dispersività del programma, data dalla creazione di numerosi file, mantenendo ovviamente la consistenza e possibilmente aumentando la velocità 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 pubblica online 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 vastità 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 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 installati nei cari punti vendita, principalmente tablet, grazie ai quali i vari clienti possono visualizzare il campionario ed eseguire gli eventuali 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



## 3 Studio delle Tecnologie

### 3.1 Stealth 3000

La maggior parte delle aziende tessili di larga scala in Italia utilizza un ERP 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'applet 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 la form di anagrafica Soggetto vista da Stealth 3000:

**STEALTH 3000**

Azioni Modifica Ricerca Blocco Record Campo ? Window

**Soggetti**

Codice  ☐ Pers. Fisica Rag. Sociale

Cliente  Fornitore  Agente  ☐ **X**

Dati Generali | Cliente | Fornitore | Agente

Indirizzo  Indirizzo 2

CAP  Località

Stato   Prov./Regione

Fax  N.ri Tel.   E-mail

P. IVA  Contatto  Lingua

P. IVA alt.  Gruppo

Cod.Fisc.  Sogg. Statistico

Indirizzi | Note | Dati Aggiuntivi

Codice  Rag. Sociale

Indirizzo  Indirizzo 2

CAP  Località

Stato   Prov./Regione

Fax  N.ri Tel.   E-mail

Contatto

Figura 1: Anagrafica Soggetti

Dati di testata:

<b>Dato</b>	<b>Descrizione</b>
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 anziché 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 1: 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 particolari processi di trattamento per i rapporti attivi e passivi.

Tabella 2: Soggetti: Dati Generali

### 3.1.2 Condizioni di Vendita

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

Tramite questa form è possibile gestire i dati di vendita del cliente con record 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) 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 default 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 nel campo 'In'.
Banca Cli.	Codice della tabella banche del Soggetto da utilizzare come banca trattata 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 3: Condizioni di Vendita: Dati Generali

### 3.1.3 Modelli

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.

Di seguito si può vedere un'anagrafica di gestione dei Modelli da Stealth 3000:

Dati di Testata:

<b>Dato</b>	<b>Descrizione</b>
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 descrizioni 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 sta 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 4: Modello: Dati di Testata

Dati Base:



<b>Dato</b>	<b>Descrizione</b>
Taglie/Misura/Nullo	Bottone 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'abilitazione stagionale del Modello, che quindi potrà essere comunque utilizzato in più stagioni.
Classe/Sottoclasse	Classificazione statistica del modello, utilizzabile in stampe e selezioni operative nelle più diverse funzioni per richiamare più modelli appartenenti alla stessa Classe/sottoclasse.

Tabella 5: Modello: Dati base

### 3.1.4 Parti

Panoramica Parti

### 3.1.5 Modelli-Parte

Panoramica Modelli-Parte

### 3.1.6 Listini di Vendita

Panoramica Listini

## 3.2 Oracle PL/SQL Developer

PL/SQL Developer è un IDE 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 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*.

Image

refcursor[...]

Image

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.

[...]

Image

### 3.2.2 Debug di un programma

Una funzionalità caratteristica e particolarmente utile di PL/SQL è quella di poter eseguire il *debug* di una funzione o di un intero package.

[...]

### 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.

[...]

Image

## 3.3 Oracle Reports

Uno degli strumenti utilizzati dall'azienda fortemente collegato al database Oracle, è Oracle Reports, 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.

[...]

Breve Panoramica Oracle Reports e studio della piattaforma

### 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. Perché una connessione abbia via dblink abbia successo, è necessario che ogni database abbia un **global database name** all'interno del dominio di rete.

[...]

## 4 Sviluppo

### 4.1 Sviluppo programma di export

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'upgrade 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, utilizzando i vari sistemi di tracciamento adottati, che vedremo in seguito nello specifico, si può 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. 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-Colore, ovvero i dati anagrafici dei Modelli-Parte con l'aggiunta dei colori abilitati agli eventi specificati nella schedulazione del programma.

```
v_query :=
'Select Distinct
t annullato , ' ||
```

```

'substr(t.cd_stagiov,1,4) mod_annorif ',' ||
'Substr(t.cd_stagiov,5,1) mod_codsta ',' ||
't.cd_linea mod_codlin ',' ||
'substr(t.cd_linea,1,2) mod_codmar ',' ||
't.cd_modello mod_codmod ',' ||
't.cd_varia mod_codvar ',' ||
't.cd_ana cd_ana ',' ||
't.cd_artico mod_codart ',' ||
't.carco mod_codcarco ',' ||
't.modparid mod_mod_par_id ',' ||
't.modid mod_mod_id ',' ||
't.parid mod_par_id ',' ||
'view_col.co_cod' ||

'From pindt_ext_modart_crm t,
(
select co.stg_col_ogg_soc_cod co_ogg_soc_cod ',' ||
'co.stg_col_col_cod co_cod ',' ||
'co.stg_col_data_mod co_data_mod ',' ||
'co.stg_col_ogg_id ogg_id' ||

'from s3t_stg_col co' ||

'UNION' ||
' select colc.stg_col_ogg_soc_cod co_ogg_soc_cod ',' ||
'
        colc.stg_col_col_cod co_cod ',' ||
'
        colc.stg_col_data_canc co_data_mod ',' ||
'
        colc.stg_col_ogg_id ogg_id' ||

'from pprrtt_stg_col colc' ||

') view_col' ||

'Where t.modparid = view_col.ogg_id' ||
'AND ((to_char(t.dataupd, 'YYYY/MM/DD HH24:MI:SS') >
'''||to_char(LastExcutionData, 'YYYY/MM/DD HH24:MI:SS')||''' ||
'
        OR' ||
' to_char(view_col.co_data_mod, 'YYYY/MM/DD HH24:MI:SS') >
'''||to_char(LastExcutionData, 'YYYY/MM/DD HH24:MI:SS')||''' ) ||
'
        OR' ||

```

```

' exists (select 1 from s3t_opb' ||
'where opb_mod_id = t.modid' ||
' and nvl(opb_f_annu,'0') = '0'') ||
'and to_char(opb_data_mod,'YYYY/MM/DD HH24:MI:SS') >
'''||to_char(LastExcutionData,'YYYY/MM/DD HH24:MI:SS')||''')' ||
' AND t.carco IS NOT NULL' ||
' And substr(t.cd_stagiov,1,4) || '/' ||
Substr(t.cd_stagiov,5,1) = '''||P_STG_ATT||'''' ||
' and t.societa = '''||s3ksysglobal.Soc_Ute||''''';

IF v_mrc_lis IS NOT NULL THEN
v_query := v_query ||
' And Substr(t.cd_linea,1,2) in ('''||v_mrc_lis||''')';
END IF;

IF v_lnv IS NOT NULL THEN
v_query := v_query ||
' And t.cd_linea in ('''||v_lnv||''')';

END IF;

v_query := v_query ||
' UNION' ||
' select DISTINCT mp.ogg_f_annu,' ||
' substr(mp.ogg_stg_anno,1,4),' ||
' substr(mp.ogg_stg_cod,1,1),' ||
' s.ogg_soc_lin_cod,' ||
' substr(s.ogg_soc_lin_cod,1,2),' ||
' substr(m.ogg_cod,6,5),' ||
' substr(m.ogg_cod,11,2),' ||
' substr(m.ogg_cod,1,5),' ||
' p.ogg_cod,' ||
' substr(mp.dagg_num5,1,3),' ||
' mp.ogg_id,' ||
' m.ogg_id,' ||
' p.ogg_id,' ||
' b.opb_col_cod' ||
'from pindt_ext_modart_crm t, s3t_opb b,
s3t_ogg mp, s3t_ogg m, s3t_ogg p, s3t_ogg_soc s' ||
'where t.modid (+)= b.opb_mod_id' ||
' and t.parid (+)= b.opb_par_id' ||

```

```

'      and t.societa(+) = b.opb_soc_cod' ||
'and to_char(b.opb_data_mod, 'YYYY/MM/DD HH24:MI:SS') >
'''||to_char>LastExcutionData, 'YYYY/MM/DD HH24:MI:SS')||'''||
' and t.cd_modello IS NULL' ||
' and mp.ogg_tipo = '5''' ||
'and mp.ogg_soc_cod = b.opb_ogg_soc_cod' ||
'and mp.ogg_mod_id = b.opb_mod_id' ||
'and mp.ogg_par_id = b.opb_par_id' ||
'and m.ogg_tipo = '1''' ||
'and mp.ogg_soc_cod = m.ogg_soc_cod' ||
'and mp.ogg_mod_id = m.ogg_id' ||
'and p.ogg_tipo = '3''' ||
'and mp.ogg_soc_cod = p.ogg_soc_cod' ||
'and mp.ogg_par_id = p.ogg_id' ||
'and s.ogg_soc_ogg_soc_cod = mp.ogg_soc_cod' ||
'and s.ogg_soc_ogg_id = m.ogg_id' ||
'and mp.dagg_num5 is not null' ||
'and mp.ogg_soc_cod =
s3ksysutils.SocPubPriv(''||s3ksysglobal.soc_ute||'', 'OGG');

IF v_mrc_lis IS NOT NULL THEN
    v_query := v_query ||
        ' And Substr(s.ogg_soc_lin_cod,1,2) in ( ' || v_mrc_lis || ' )';
END IF;

IF v_lnv IS NOT NULL THEN
    v_query := v_query ||
        ' And s.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-parte-colore.

LOOP

```

fetch Cur_modcol into
    f_annu_col, mod_annorif_col,
    mod_codsta_col, mod_codlin_col,
    mod_codmar_col, mod_codmod_col,
    mod_codvar_col, cd_ana_col,
    mod_codart_col, mod_codcarco_col,
    mod_mod_par_id_col, mod_mod_id_col,
    mod_par_id_col, v_col_id;
exit when Cur_modcol\%notfound;

Open CAblCol(mod_mod_par_id_col, mod_par_id_col,
             Mod_Annorif_col, Mod_Codsta_col);

Loop
    Begin
    Fetch CAblCol Into RAblCol;
    Exit When CAblCol\%Notfound;
    If RAblCol.col_cod<>'000' —colori di campionato

    Then —colori in cartella colori
        —ctr esistenza in cartella colori
        v_col_blc :=Get_Blocco_Colore(
                                Mod_Annorif_col,
                                Mod_Codsta_col,
                                Mod_Mod_Par_Id_col,
                                Mod_Mod_Id_col,
                                Mod_Par_Id_col,
                                RAblCol.Col_Cod);
        —Colori Bloccati

    IF v_col_blc IS NULL THEN
        v_col_blc := '0';
    ELSE
        v_col_blc := '1';

```

```

        END IF ;
        v_cct_id:=PPPRKOGGCCT.Get_CCT_ID(
            s3ksysglobal.soc_ute ,
            Mod_Annorif_col,Mod_Codsta_col ,
            Mod_Codmar_col,Mod_Codlin_col ,
            Mod_Par_Id_col,mod_codcarco_col );
        v_esiste:=0;
        If v_esiste=0 AND (v_opb_data_mod>
            LastExcutionData
            OR
            RAblCol.sc_data_mod>
            LastExcutionData)
Then
BEGIN
    v_img2 := null;
    IF Mod_Codcarco_col IS NOT NULL THEN
        v_img2      := Mod_Annorif_col ||
            Mod_Codsta_col ||
            Mod_Codlin_col ||
            Mod_Codart_col ||
            Mod_Codcarco_col ||
            RAblCol.Col_Cod;
    END IF ;
    INSERT INTO PINDT_MODART_CRM(
        cd_modello ,
        cd_varia ,
        tipoana ,
        cd_ana ,
        cd_artico ,
        cd_cart ,
        cd_colore ,
        cd_linea ,
        datains ,
        dataupd ,
        annullato ,
        cd_stagiov ,
        imgname ,
        imgname2 ,
        flagmo ,
        flagspecial ,

```

```

        disattiva ,
        col_annu ,
        societa
    )
VALUES( Mod_Codmod_col ,
        Mod_Codvar_col ,
        'C' ,
        cd_ana_col ,
        Mod_Codart_col ,
        'STD' ,
        RAblCol.Col_Cod ,
        mod_codlin_col ,
        LastExecutionData ,
        v_sysd ,
        nvl(v_col_blc , '0 ' ) ,
        Mod_Annorif_col ||
        Mod_Codsta_col ,
        Mod_Annorif_col ||
        Mod_Codsta_col ||
        substr(Cd_Ana_col , 3 ) ||
        Mod_Codmod_col ||
        Mod_Codvar_col ||
        Mod_Codart_col ||
        RAblCol.Col_Cod ,
        v_img2 ,
        '0 ' ,
        '0 ' ,
        '0 ' ,
        '0 ' ,
        s3ksysglobal.Soc_Ute
    );
exception
when dup_val_on_index then
BEGIN
    UPDATE PINDT_MODART_CRM t
    SET t.dataupd = v_sysd ,
        t annullato = nvl(v_col_blc , '0 ' ) ,
        t.imgname   = Mod_Annorif_col ||
                    Mod_Codsta_col ||
                    substr(Cd_Ana_col , 3 ) ||

```

```

Mod_Codmod_col ||
Mod_Codvar_col ||
Mod_Codart_col ||
RAblCol.Col_Cod,
t.imgname2 = v_img2,
t.col_annu = '0'
WHERE t.cd_modello = Mod_Codmod_col
AND t.cd_varia = Mod_Codvar_col
AND t.tipoana = 'C'
AND t.cd_ana = Cd_Ana_col
AND t.cd_artico = Mod_Codart_col
AND t.cd_cart = 'STD'
AND t.cd_colore = RAblCol.Col_Cod
AND t.cd_stagiov = Mod_Annorif_col ||
Mod_Codsta_col
AND t.cd_linea = Mod_Codlin_col
and societa = s3ksysglobal.Soc_Ute;
exception when others then
v_error := sqlerrm;
s3ksysmess.Batch_Messaggi(60070 ,sysdate ,
'MODELLO-COLORE: ' ||
nvl(Mod_Codmod_col,'codice null') || '/' ||
nvl(Mod_Codart_col,'codice null') || '/' ||
nvl(RAblCol.Col_Cod,'codice null'));

END;
when others then
v_error := sqlerrm;
s3ksysmess.Batch_Messaggi(60070 ,sysdate ,
'MODELLO-COLORE: ' ||
nvl(Mod_Codmod_col,'codice null') || '/' ||
nvl(Mod_Codart_col,'codice null') || '/' ||
nvl(RAblCol.Col_Cod,'codice null'));

END;
End If;
End If;
Exception When Others Then
v_error:=Sqlerrm;
s3ksysmess.Batch_Messaggi(60070 ,sysdate );
End;
End Loop;

```

```
Close CAblCol ;
```

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 utilizzata come testata per un secondo cursore interno, il quale estrae tutti i colori disponibili per quel modello-parte.

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_MODCOL IN (
    SELECT * FROM PINDT_MODART_CRM t
    WHERE t.dataupd = v_sysd
           and t.societa = s3ksysglobal.Soc_Ute
           AND t.cd_stagiov = replace(P_STG_ATT, '/' ) )
  LOOP
    BEGIN
      IF nvl(Cur_modcol.Col_Annu, '0') = '0' THEN
        INSERT INTO modelliarticoli@crm_sydat_eur.industries.com(
          "Cd_Modello",
          "Cd_Varia",
          "TipoAna",
          "Cd_Ana",
          "Cd_Artico",
          "Cd_variat",
          "Cd_Cart",
          "Cd_Colore",
          "CD_Variante",
          "CD_Lava",
          "CD_Drop",
          "CD_Statura",
          "cd_stagProd",
          "cd_stagiov",
          "CD_Flash",
```

```
"CD_Collez" ,
"Cd_Linea" ,
"imgName" ,
"imgName2" ,
"Annullato" ,
"FlagMO" ,
"FlagSpecial" ,
"Disattiva" ,
"DataUpd" ,
"DataIns" ,
"Cd_TemaC" ,
"Desc_TemaC")
VALUES(Cur_modcol.Cd_Modello ,
Cur_modcol.Cd_Varia ,
Cur_modcol.Tipoana ,
Cur_modcol.Cd_Ana ,
Cur_modcol.Cd_Artico ,
Cur_modcol.Cd_Variat ,
Cur_modcol.Cd_Cart ,
Cur_modcol.Cd_Colore ,
Cur_modcol.Cd_Variante ,
Cur_modcol.Cd_Lava ,
Cur_modcol.Cd_Drop ,
Cur_modcol.Cd_Statura ,
Cur_modcol.Cd_Stagprod ,
Cur_modcol.Cd_Stagiov ,
Cur_modcol.Cd_Flash ,
Cur_modcol.Cd_Collez ,
Cur_modcol.Cd_Linea ,
Cur_modcol.Imgname ,
Cur_modcol.Imgname2 ,
Cur_modcol.Annullato ,
Cur_modcol.Flagmo ,
Cur_modcol.Flagspecial ,
Cur_modcol.Disattiva ,
v_date ,
v_date ,
CUr_modcol.Cod_Temac ,
Cur_modcol.Des_Temac
);
```

```

commit;

v_modelli_par_col_ins := v_modelli_par_col_ins + 1;
ELSE
BEGIN
DELETE FROM modelliarticoli@crm_sydat_eur.industries.com
WHERE "Cd_Modello"      = Cur_modcol.Cd_Modello
AND "Cd_Varia"          = Cur_modcol.Cd_Varia
AND "TipoAna"           = Cur_modcol.Tipoana
AND "Cd_Ana"            = Cur_modcol.Cd_Ana
AND "Cd_Artico"         = Cur_modcol.Cd_Artico
AND "Cd_Cart"           = Cur_modcol.Cd_Cart
AND "Cd_Colore"         = Cur_modcol.Cd_Colore;
commit;
if SQL\%rowcount > 0 then
v_modelli_par_col_del := v_modelli_par_col_del + 1;
end if;
exception when others then
rollback;
v_error:= sqlerrm;
s3ksysmess.Batch_Messaggi(60071 ,sysdate);
END;
END IF;
exception
when Dup_Insert then
BEGIN
UPDATE modelliarticoli@crm_sydat_eur.industries.com
SET "Cd_variat"          = Cur_modcol.Cd_Variat ,
"CD_Variante"           = Cur_modcol.Cd_Variante ,
"CD_Lava"               = Cur_modcol.Cd_Lava ,
"CD_Drop"               = Cur_modcol.Cd_Drop ,
"CD_Statura"            = Cur_modcol.Cd_Statura ,
"cd_stagProd"           = Cur_modcol.Cd_Stagprod ,
"cd_stagiov"            = Cur_modcol.Cd_Stagiov ,
"CD_Flash"              = Cur_modcol.Cd_Flash ,
"CD_Collez"             = Cur_modcol.Cd_Collez ,
"Cd_Linea"              = Cur_modcol.Cd_Linea ,
"imgName"               = Cur_modcol.Imgname ,
"imgName2"              = Cur_modcol.Imgname2 ,
"Annullato"             = Cur_modcol.Annullato ,

```

```

        "DataUpd"      = v_date ,
        "Cd_TemaC"     = Cur_modcol.Cod_Temac ,
        "Desc_TemaC"   = Cur_modcol.Des_Temac
WHERE  "Cd_Modello"   = Cur_modcol.Cd_Modello
      AND "Cd_Varia"   = Cur_modcol.Cd_Varia
      AND "TipoAna"    = Cur_modcol.Tipoana
      AND "Cd_Ana"     = Cur_modcol.Cd_Ana
      AND "Cd_Artico"  = Cur_modcol.Cd_Artico
      AND "Cd_Cart"    = Cur_modcol.Cd_Cart
      AND "Cd_Colore"  = Cur_modcol.Cd_Colore ;
commit ;
v_modelli_par_col_upd := v_modelli_par_col_upd + 1 ;
exception when others then
    v_error := sqlerrm ;
    s3ksysmess.Batch_Messaggi(60071 ,sysdate ,
                                'Modelli Colore ' || sqlerrm ) ;
    v_modelli_par_col_err := v_modelli_par_col_err + 1 ;
    rollback ;
END ;
when others then
    v_error := sqlerrm ;
    s3ksysmess.Batch_Messaggi(60071 ,sysdate ,
                                'Modelli Colore ' || sqlerrm ) ;
    v_modelli_par_col_err := v_modelli_par_col_err + 1 ;
    rollback ;
END ;
END LOOP ;
exception when others then
    s3ksysmess.Batch_Messaggi(1583 ,sysdate ,
                                'Modelli Colore: ',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 presen-



za 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

La versione iniziale del programma di import è stata sviluppata nel 2002 da uno dei programmatori attualmente presenti in azienda e si basa su[...]

## 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 [...]

## 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