Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea Specialistica in Ingegneria Informatica

Database reverse engineering e porting di applicazioni Access su Oracle: il caso Bosch Rexroth Oil Control S.p.A.

Relatore
Prof.ssa Sonia Bergamaschi

Tesi di Laurea di
Giampiero Miccoli

Anno Accademico 2007-2008

Parole chiave:

Reverse engineering
Porting tra RDBMS

Indice

Elenco delle figure

Introduzione

Il presente lavoro di tesi ha lo scopo di introdurre l'azienda dove è stato svolto il periodo di stage, della durata complessiva di sei mesi, presso il polo amministrativo dell'azienda Bosch Rexroth Oil Control S.p.A, gruppo leader mondiale nella progettazione, prototipazione e fabbricazione di valvole oleodinamiche, a cartuccia e con collettore, per il mercato del macchinario mobile e dell'impiantistica industriale.
Il gruppo è composto da diversi plant produttivi: Oil Control, TARP, EDI System, LC Oleodinamica, Oil Sistem, Fimma e sono distribuiti nel territorio emiliano tra Modena e Reggio Emilia.
Lo stage è stato svolto presso il reparto ISY (Information System) presente nella divisione Oil Control a Nonantola (Modena).
Il reparto presente in azienda ricopre diversi ruoli, come l'assistenza agli utenti degli uffici ed alla linea produttiva, attività di supporto alle varie funzioni aziendali sui sistemi gestionali ed attività di Data Security e Privacy dei dati.
Il lavoro principalmente svolto è stato quello di occuparsi della gestione degli ambienti applicativi e dei database, attraverso l'analisi, modifiche e cancellazioni di query e programmi sviluppati, con lo scopo di adeguarli alle recenti integrazioni delle divisioni Bosch Rexroth Oil Control. Oltre al lavoro appena descritto oggetto del tirocinio, è stato ricoperto il ruolo di supporto al personale addetto ad effettuare report/query di estrazione dati richiesti dai diversi uffici (Amministrazione, Logistica, ecc.) e lavoro di supporto al personale di Data Security interno dell'azienda e delle diverse divisioni appartenenti al gruppo.

Nei primi due capitoli verrà presentata l'azienda, soffermandosi sulle fasi che hanno portato alla sua attuale struttura societaria ed organizzativa per poi soffermarsi sull'analisi della struttura dei sistemi informativi gestionali utilizzati all'interno del gruppo e del reparto presso le quali è stato effettuato lo stage, spiegando in dettaglio le attività svolte.
Nel terzo capitolo vi è un'approfondita analisi del sistema gestionale ERP Apache utilizzato all'interno della divisione Oil Control di Nonantola.
Il quarto capitolo è interamente dedicato al lavoro oggetto del tirocinio, ovvero sull'analisi e adeguamento delle applicazioni presenti in azienda, molte delle quali sviluppate nel corso

degli anni attraverso Microsoft Access che interagiscono direttamente con il database del sistema gestionale Apache che si basa sul RDBMS Oracle. Queste applicazioni sono state sviluppate su richiesta dei diversi uffici, soprattutto dall'Ufficio commerciale e dalla Logistica, ma molti di questi risultano da aggiornare alle nuove esigenze aziendali o addirittura parzialmente/completamente obsoleti. Inoltre vi è la volontà di voler impiegare tecnologie più recenti e la necessità di imporre maggiori controlli su quali utenti li eseguono e su quali dati possono accedere. Per poter effettuare questo si sono dovute applicare le tecniche di database reverse engineering sulle applicazioni in Access, effettuare la completa riscrittura delle query ed è stata realizzata un'applicazione in C# che permette di poter eseguire le nuove query direttamente sul database server Oracle.
Il quinto capitolo tratta come argomento principale LINQ (Language-Integrated Query), ovvero una tecnologia realizzata dalla Microsoft che offre un modello unificato di programmazione per interrogare le più disparate sorgenti di dati utilizzando la stessa sintassi. Sono prese in esame le versioni LINQ to SQL (una particolare versione simile a SQL) e LINQ to Entities (legato ad Entity Framework, un'infrastruttura per l'astrazione in entità che può essere interrogata da LINQ). Oltre ad introdurre gli aspetti principali di tale tecnologia si è cercato di applicarla alle query realizzate utilizzando come sorgente dati il RBDMS Oracle.

Capitolo 1

L'organizzazione Aziendale

1.1 Il Gruppo Bosch

Da oltre cento anni il nome **Bosch** [2] viene associato a tecnologia all'avanguardia e invenzioni pionieristiche che hanno fatto la storia. Bosch distribuisce i suoi prodotti in tutto il mondo ed è attiva nei più disparati settori, come la ricerca e la produzione di tecnologia per autoveicoli, tecnologia industriale, beni di consumo e tecnologie costruttive. Le origini dell'azienda si trovano nell'*Officina di meccanica di precisione ed elettrotecnica* fondata a Stoccarda nel 1886 da Robert Bosch (1861-1942).
La particolare struttura societaria assicura al Gruppo Bosch indipendenza finanziaria ed autonomia per il reinvestimento totale degli utili destinati al miglioramento delle relazioni tra i popoli, ad iniziative in campo sociale, sanitario, educativo ed alla ricerca. Bosch è l'unica fra le aziende delle sue dimensioni a non dipendere dalle aspettative della Borsa, che cerca inevitabilmente risultati di breve termine, ma ai principi dettati dal suo fondatore. Soci e strutture di governance che fanno parte della *Robert Bosch GmbH* sono:

- ***Fondazione Robert Bosch***
- ***Famiglia Bosch***
- ***Società fiduciaria Robert Bosch Industrietreuhand KG***

Il gruppo Bosch non è quotato ed è partecipato al 92% dalla Fondazione Robert Bosch che non ha diritto di voto, al 7% dalla famiglia Bosch (azioni con diritto di voto) e all'1% dalla Robert Bosch GmbH senza diritto di voto. La società fiduciaria Robert Bosch Industrietreuhand KG (Comitato di Direzione) detiene, invece, il 93% dei diritti di voto, ma nessuna quota nel capitale. Questa istituzione rappresenta la volontà del fondatore di operare per il benessere pubblico in coerenza con i tempi e la società moderna.

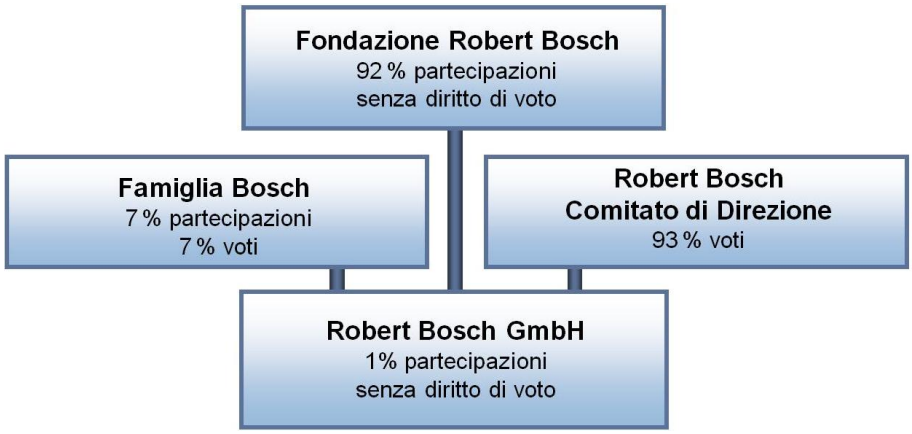

Figura 1.1: La costituzione della Casa Bosch

Robert Bosch GmbH è un fornitore globale leader nei settori *Tecnica per autoveicoli*, *Tecnologia industriale* nonché nei *Beni di consumo e Tecnologie costruttive*. Più in dettaglio:

- Il settore **Tecnica per Autoveicoli** è la divisione più grande del gruppo Bosch e si articola in quattro macroaree di attività: tecnica di iniezione per motori a benzina e diesel, sistemi per sicurezza attiva e passiva del guidatore e dei passeggeri (impianti di frenata idraulici e componenti, sistemi antibloccaggio ABS, antislittamento in accelerazione ASR, elettronico di stabilità ESP, regolazione Airbag), dispositivi elettrici (starter, alternatori e motorini) e, infine, i prodotti per la comunicazione mobile (autoradio, sistemi per la navigazione, sistemi informativi di bordo e telematica, sistemi multimedia - marchio Blaupunkt). In questo settore sono presenti aziende che offrono attrezzature per il controllo e la regolazione dell'assetto, di banchi prova freno e di smontagomme ed equilibratrici per il service dei pneumatici (marchi Beissbarth e Sicam).

- Il settore **Tecnologia industriale** (nelle Divisioni Tecnica di automazione, Tecnologia di imballaggio e Tecnologia Metallurgica) è ben rappresentato dalla controllata ***Bosch Rexroth***, uno dei principali fornitori sul mercato mondiale. Quest'ultima è poi uno dei produttori leader di Idraulica compatta, grazie anche all'acquisizione della ***Oil Control Group S.p.A.***

- Nell'ambito del **Settore Beni di Consumo e Tecnologie Costruttive** Bosch è presente sul mercato con le Divisioni Elettroutensili, degli accessori e del giardinaggio (marchi Bosch, Skil e Dremel), Termotecnica (marchi Buderus, e.l.m. leblanc, Junkers) e Sistemi di sicurezza, oltre che con i prodotti Elettrodomestici della Società a partecipazione BSH Bosch&Siemens Hausgeraete GmbH (marchi Bosch, Siemens, Gaggenau e Neff).

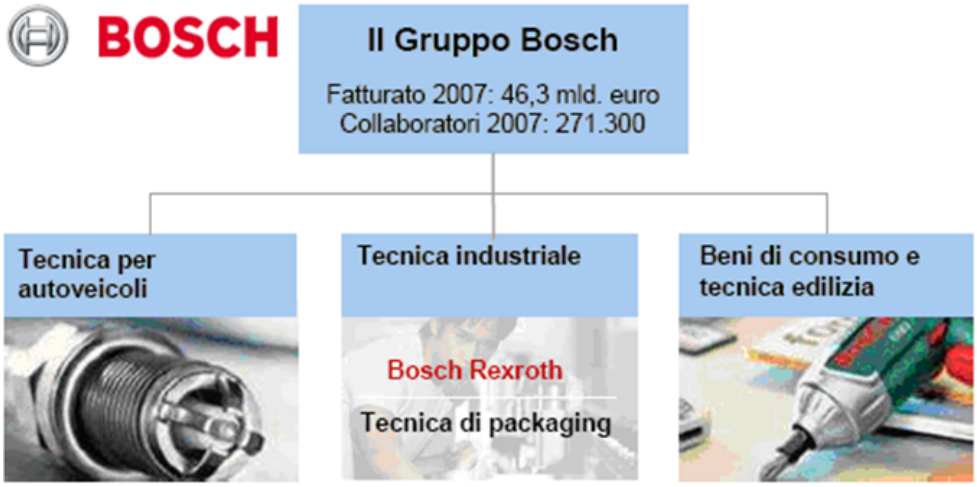

Figura 1.2: Le principali divisioni del Gruppo Bosch

Figura 1.3: Alcuni produttori appartenenti al gruppo

Bosch è già da tempo attiva nel campo delle energie rinnovabili con la voglia di contribuire attivamente alla difesa dell'ambiente e alla salvaguardia delle risorse naturali. Infatti recentemente il gruppo Bosch ha istituito una nuova divisione data da alcune acquisizioni con l'intenzione di potenziare il business legato alle energie rinnovabili. L'azienda ha acquisito la maggioranza della ersol Solar Energy AG di Eufurt (Germania) che si occupa dello sviluppo, della produzione e della vendita di celle solari e pannelli fotovoltaici e dispone di tre sedi, rispettivamente in Germania, negli U.S.A. e in Cina. Bosch Rexroth, ad esempio, fornisce ingranaggi per turbine eoliche, sviluppa componenti per il settore, peraltro ancora emergente, dell'energia che sfrutta le correnti marine e produce sistemi idraulici per le stazioni solari termiche, mentre la divisione Termotecnica si è affermata come leader nel mercato in rapida crescita delle pompe di calore ed è uno dei produttori principali di pannelli solari per la produzione di acqua calda. Inoltre, nel giugno 2007, Bosch ha iniziato una collaborazione con la BASF e la Heliatek GmbH (entrambe in Germania) nel promettente settore del fotovoltaico organico.

Il Gruppo Bosch [3] opera in Italia dal 1904, quando venne inaugurato il primo ufficio di rappresentanza a Milano. Oggi l'Italia costituisce per il Gruppo Bosch uno dei

mercati esteri più importanti, il quarto nel Mondo, dopo Stati Uniti, Francia e Giappone. Il Gruppo Bosch è presente in Italia con le seguenti aziende:

- **Robert Bosch S.p.A.** Milano - Attività di vendita e assistenza clienti: primo equipaggiamento veicoli, ricambi per veicoli e attrezzatura d'officina, autoradio e sistemi di navigazione Blaupunkt, elettroutensili Bosch, Dremel e Skil, sistemi di riscaldamento e produzione di acqua calda sanitaria e sistemi di regolazione e controllo di caldaie e scaldabagni a gas Junkers ed e.l.m. leblanc.
- **Tecnologie Diesel e Sistemi Frenanti S.p.A.** Modugno (BA) - Attività di produzione: pompe ad alta pressione per il sistema ad iniezione diretta Diesel Common Rail e componenti per sistemi frenanti.
- **Centro Studi Componenti per Veicoli S.p.A.** Modugno (BA) - Attività di sviluppo di progetti di ricerca relativi a componenti del sistema Common Rail. Centro di competenza per le pompe ad alta pressione.
- **VHIT S.p.A Vacuum & Hydraulic Products Italy** Offanengo (CR) - Attività di produzione: pompe del vuoto e idraulica per veicoli industriali.
- **Bosch Rexroth S.p.A.** Cernusco sul Naviglio (Ml) - Attività commerciali: apparecchiature idrauliche, pneumatiche e automazione industriale.
- **Aresi S.p.A.** Brembate (BG) - Attività di produzione: accessori per elettroutensili.
- **Bosch Security Systems S.p.A.** Milano - Attività commerciali: sistemi di sicurezza, TVCC, Audio e Congress.
- **Buderus Italia S.r.l.** Assago (Ml) - Attività commerciali: sistemi di riscaldamento e produzione di acqua calda sanitaria e sistemi di regolazione e controllo.
- **BSH Elettrodomestici S.p.A.** Milano - Attività commerciali: frigoriferi, congelatori, lavatrici, lavastoviglie, forni e piccoli elettrodomestici.
- **Sicam S.r.l.** Correggio (RE) - Attività di produzione: attrezzature per garage ed autofficine.

Nel gruppo è presente anche **TEC** (Training/Esperienze/Competenze), la scuola di formazione manageriale e tecnologica del Gruppo Bosch in Italia.

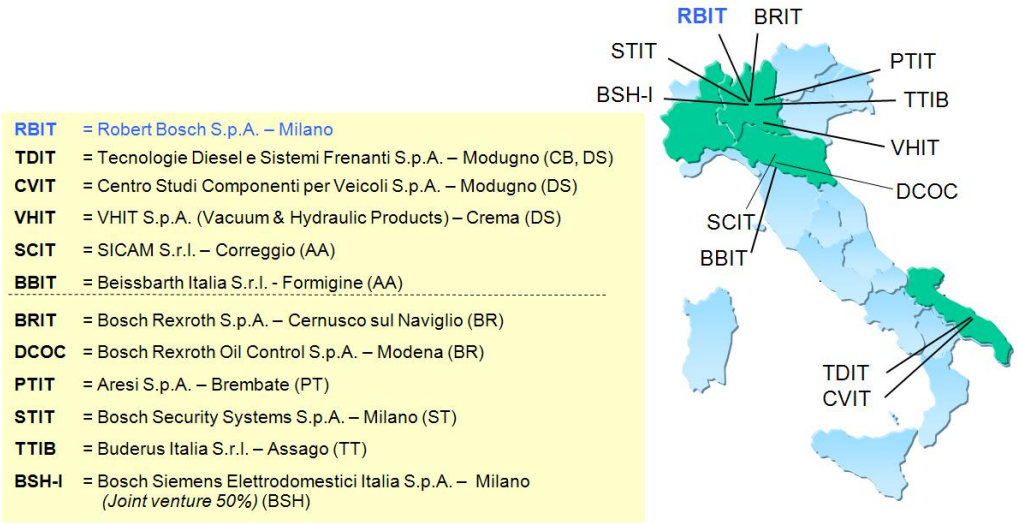

Figura 1.4: Presenza del Gruppo Bosch in Italia

1.2 Bosch Rexroth

Il gruppo internazionale ***Bosch Rexroth AG*** [4], nato nel maggio del 2001 dalla fusione della divisione di *Tecnica di Automazione e Oleodinamica Mobile* (Bosch Automationstechnik) di Robert Bosch S.p.A., *Mannesmann Rexroth AG* e *Star Mikron S.p.A.*, produce e commercializza la gamma di prodotti in precedenza appartenenti ai marchi *Bosch Automation*, *Brueninghaus Hydromatik*, *Indramat*, *Lohmann & Stolterfoht*, *Mecman*, *Refu*, *Rexroth Hydraulics* e *Star*.

L'azienda offre componenti e soluzioni complete per i sistemi d'azionamento e il controllo del movimento per tutti i settori industriali: dalle presse alle macchine utensili e per la plastica, dalle applicazioni per la lavorazione del marmo al food & packaging, printing & paper, automation, handling, movimento terra e macchine da lavoro (dall'oleodinamica e la pneumatica, alla tecnica lineare e di montaggio), ingegneria civile, fino agli azionamenti e ai controlli elettrici. Ne consegue che Bosch Rexroth è l'unica azienda ad offrire una soluzione per ogni differente campo di applicazione.

La strategia di Rexroth, orientata alla fornitura di soluzioni di sistema per specifiche applicazioni trova compimento nei progetti di Systems & Engineering. La società si è specializzata nell'integrazione di sistemi di azionamento e controllo completi nelle macchine e negli impianti del cliente, anche con formula "chiavi in mano". Rexroth Systems & Engineering utilizza l'oleodinamica come tecnica d'azionamento ideale per le applicazioni più grandi e impegnative. La multinazionale tedesca fornisce soluzioni per progetti di macchinari di scena teatrali, parchi giochi, simulatori di movimento, ricerca marina e ingegneria civile. In questi casi dove le funzioni di azionamento e controllo si fanno molto complesse, Rexroth si occupa interamente di questo compito, lasciando che il cliente si

possa concentrare solo sui problemi fondamentali della progettazione.
La rete di vendita diretta è affiancata da una capillare rete di distributori autorizzati, partner commerciali che integrano e supportano l'offerta Rexroth garantendo un servizio di customizing e logistico immediato e flessibile, capace di soddisfare le richieste dei clienti in tempi brevissimi.
Bosch Rexroth S.p.A. [5] è la sede italiana della Bosch Rexroth AG ed è controllata al 100% dalla Robert Bosch GmbH.
Questa presenza in più settori fa di Bosch Rexroth un'azienda unica nel panorama industriale internazionale, con oltre 80 sedi in tutto il mondo e con cinque centri regionali in Italia: Milano, Torino, Padova, Bologna e Napoli.

Figura 1.5: Diffusione della Bosch Rexroth su scala mondiale

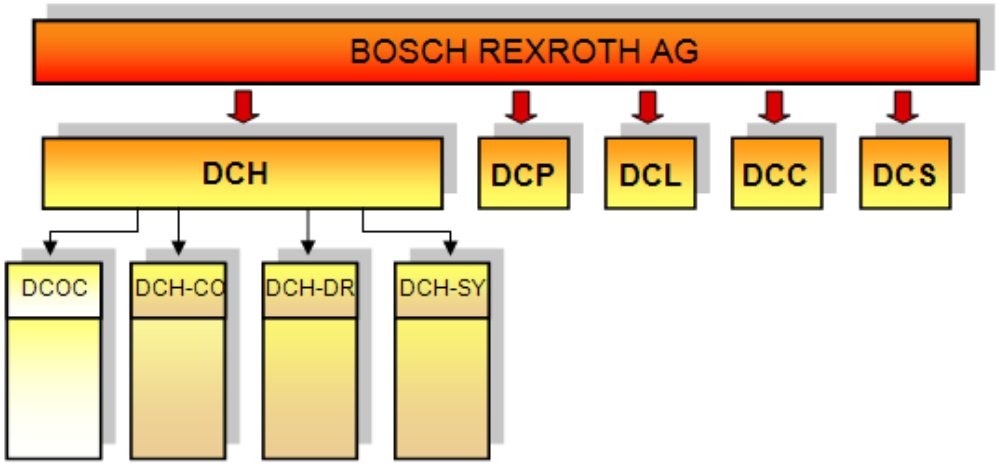

Figura 1.6: Le Business Unit di Bosch Rexroth AG

Così come avviene per le Business Unit (BU) a livello internazionale, la struttura commerciale di Bosch Rexroth S.p.A., per essere più vicina al cliente, si ispira a un'organizzazione a "matrice".
Tra le diverse divisioni presenti nel gruppo Rexroth citiamo:

- **DCH - *Oleodinamica Industriale e Applicazioni Mobili*** *(Industrial and mobile hydraulics)*;
- **DCP - *Pneumatica*** *(Pneumatic components and systems)*;
- **DCL - *Tecnica Lineare e di Montaggio*** *(Linear and assembly equipment)*;
- **DCC - *Controlli e Azionamenti Elettrici*** *(Industrial Automation and Control Systems)*;
- **DCS - *Service*** *(Service industry and manufacturing automation).*

dove l'acronimo DC viene utilizzato per indicare *Drive and Control Technology Division*.
I prodotti di ciascuna Business Unit sono:

- **DCH - Hydraulics**: componenti idraulici ed elettronici per il comando, il controllo, la regolazione e la movimentazione di organi, di macchine e/o impianti adatti per differenti settori applicativi, tra cui le applicazioni mobili.
- **DCP - Pneumatics**: attuatori, valvole con azionamento manuale, meccanico o pneumatico, valvole elettromagnetiche, valvole regolatrici di pressione, batterie di valvole a comando BUS, filtri, unità trattamento aria, sistemi di controllo, sistemi personalizzati, prodotti per l'automazione.
- **DCL - Linear Motion and Assembly Technologies**: guide a sfere e a rulli, guide lineari con manicotti a sfere, unità viti a sfere e cuscinetti d'estremità, sistemi lineari, elementi meccanici di base, sistemi manuali, tecnica per il flusso di informazioni e materiali, robot e sistemi di gestione, software di progettazione e istruzioni.
- **DCC - Electric Drives and Controls**: servomotori brushless e asincroni, motori lineari, azionamenti digitali, PLC, controlli numerici, sistemi di saldatura e di avvitatura.

A differenza delle altre divisioni, la **DCS Service Automation** è stata sviluppata allo scopo di garantire un supporto ottimale in ogni fase del ciclo di vita dell'impianto, al fine di evitare periodi di inattività dovuti a fermo macchina. Questo servizio di consulenza qualificato riunisce in un'unica organizzazione tutte le prestazioni di assistenza per l'intera gamma di prodotti dell'automazione industriale, dall'help desk al servizio parti di

ricambio, al servizio sul campo e di riparazione, fino al retrofit/modernizzazione e alla formazione.
Con queste divisioni Bosch Rexroth si propone come integratore di sistemi per tutti i principali settori industriali, in grado di offrire per ogni esigenza applicativa la soluzione d'azionamento più adatta, sia idraulica, pneumatica, meccanica sia elettrica. Negli anni, infatti, Bosch Rexroth ha acquisito un vastissimo know-how in termini di progettazione e realizzazione di applicazioni a elevato contenuto tecnologico. I numerosi centri di competenza dislocati nel mondo sviluppano e mettono a disposizione dell'intero gruppo tutte le esperienze maturate nei contesti specifici ed i funzionari commerciali sono affiancati da tecnici specializzati nei principali settori industriali (dalle macchine utensili alle presse, dalle macchine per l'industria alimentare a quelle per la stampa, dai centri di lavoro alle centrali eoliche). I clienti possono quindi disporre di componenti "best in class" e contare su un interlocutore in grado di offrire una consulenza nei diversi ambiti applicativi.

1.3 Bosch Rexroth Oil Control

Bosch Rexroth Oil Control S.p.A. [6] [7] è uno dei più importanti costruttori nell'ambito dell'idraulica compatta ed offre, con le sue valvole idrauliche/oleodinamiche, blocchi di comando integrati e minicentraline, una gamma di prodotti altamente tecnologici (ad esempio componenti a supporto di scavatori, ruspe, carrelli elevatori, gru, carrelli aerei ed altro macchinario mobile). L'azienda è presente sia nel *settore mobile* (*mobile hydraulics*), dove la gamma delle valvole copre tutte le applicazioni possibili, dal movimento terra al sollevamento, e sia nel *settore industriale* (*industrial hydraulics*), grazie alla continua innovazione in campo oleoidraulico con soluzioni compatte ed integrate (*compact hydraulics*). Bosch Rexroth Oil Control è una Business Unit appartenente al gruppo *Bosch Rexroth AG*, affiliata al 100% a Robert Bosch GmbH, ed è il frutto dell'acquisizione avvenuta nel 2005, rafforzandone così le attività nel campo dell'idraulica compatta. L'azienda utilizza per le proprie attività prodotti internamente sviluppati e realizzati come Business Unit di Bosch Rexroth AG. Il gruppo in Italia è principalmente presente nell'area di Modena e Reggio Emilia con 8 stabilimenti di produzione, circa 1.400 collaboratori e società distributrici in Europa e negli Stati Uniti. La struttura organizzativa aziendale è composta dalla sede amministrativa a Nonantola (Modena) dove sono dislocate le *Funzioni Direzionali, Tecniche e Amministrative* e le *Divisioni di Prodotto* che sviluppano e realizzano i prodotti. Le più grandi aziende che compongono il gruppo sono: ***Oil Control*** con sede a Nonantola (Modena), ***TARP*** con sede a Pavullo nel Frignano (Modena), ***EDI System*** con sede a Modena, ***Oil Sistem*** e ***Fimma*** con sede a Reggio Emilia e ***LC Oleodinamica*** con sede a Vezzano sul Crostolo (Reggio Emilia), facendo diventare la Oil Control una holding mista (in quanto la società capogruppo svolge anche l'attività

di produzione e di scambio). Queste aziende costituiscono il "nocciolo duro" del gruppo e si propongono con un marchio proprio, ma ne sono arrivate altre, alcune create direttamente, mentre altre sono state acquisite: tutte sono al servizio del gruppo pur avendo peculiarità fra loro diverse. L'azienda è presente nella figura 1.6 dove sono indicate le Business Unit del gruppo Rexroth attraverso l'acronimo ***DCOC*** (***Drive Control Oil Control***).

1.3.1 L'organizzazione aziendale

[8] La Oil Control nasce nel 1969 come ***Fratelli Storci S.n.c.***, fondata da Andrea e Orlando Storci, e si occupa di lavorazioni meccaniche per conto terzi. È una delle primissime aziende in Europa ad avviare la produzione delle valvole idrauliche, fino ad allora importate solo dagli Stati Uniti. La scelta risulta vincente tanto da far nascere nel 1974 la ***Oil Control*** che, da subito, trova nei costruttori di gru i principali clienti. Per tutti gli anni Settanta e per la prima parte degli Ottanta l'azienda si concentra sul mercato italiano acquisendo come clienti le principali case del settore del sollevamento, facendosi largo in un comparto fino a quel momento monopolio di aziende straniere. Nella seconda metà degli anni Ottanta inizia ad aprire filiali all'estero, iniziando dall'Inghilterra, seguita, negli anni successivi, dalle filiali in Germania, Danimarca, Olanda e Francia, fino ad arrivare con un ufficio di rappresentanza in Cina ed un'importante filiale negli Stati Uniti. Il successo commerciale è sempre stato supportato da un'evoluzione organizzativa e da un piano industriale concepito per ridurre i costi produttivi e teso alla ricerca tecnologica. Per essere competitiva sul mercato, nel corso degli anni Oil Control ha ampliato la propria struttura societaria, anche attraverso la creazione di un vero e proprio network, costituito da aziende che si occupano di produrre la componentistica utilizzata dall'azienda per la realizzazione del prodotto finito, e da imprese costruttrici di "prodotti complementari", capaci cioè di fare "sistema" con la casa madre, regalando una gamma più ampia e competitiva.
La grande esperienza accumulata negli anni viene "capitalizzata" nel 1998, quando i soci decidono di dare una struttura più solida al network aziendale, cercando di creare un gruppo. Le azioni vengono così conferite in un'unica società, la ***Oil Control Group S.p.A.*** [9], una holding finanziaria senza funzioni operative. Un contenitore di azioni che dà corpo a una realtà industriale presente in tutto il mondo. Da qui parte un'importante processo di integrazione, che va oltre le interdipendenze di tipo produttivo e commerciale, tra le varie realtà del gruppo. L'azienda, potendo disporre di una buona liquidità, decide di procedere ad alcune importanti acquisizioni. Tra le diverse aziende citiamo la *TARP*, la *Edi System*, la *Oil Sistem* e la *LC Oleodinamica*. Nel 2003 Oil Control Group è una realtà consolidata, capace di esprimere competenza e professionalità, che ha nel proprio

portafoglio aziende molto importanti del calibro di Catepillar, JBC e il gruppo CNH.

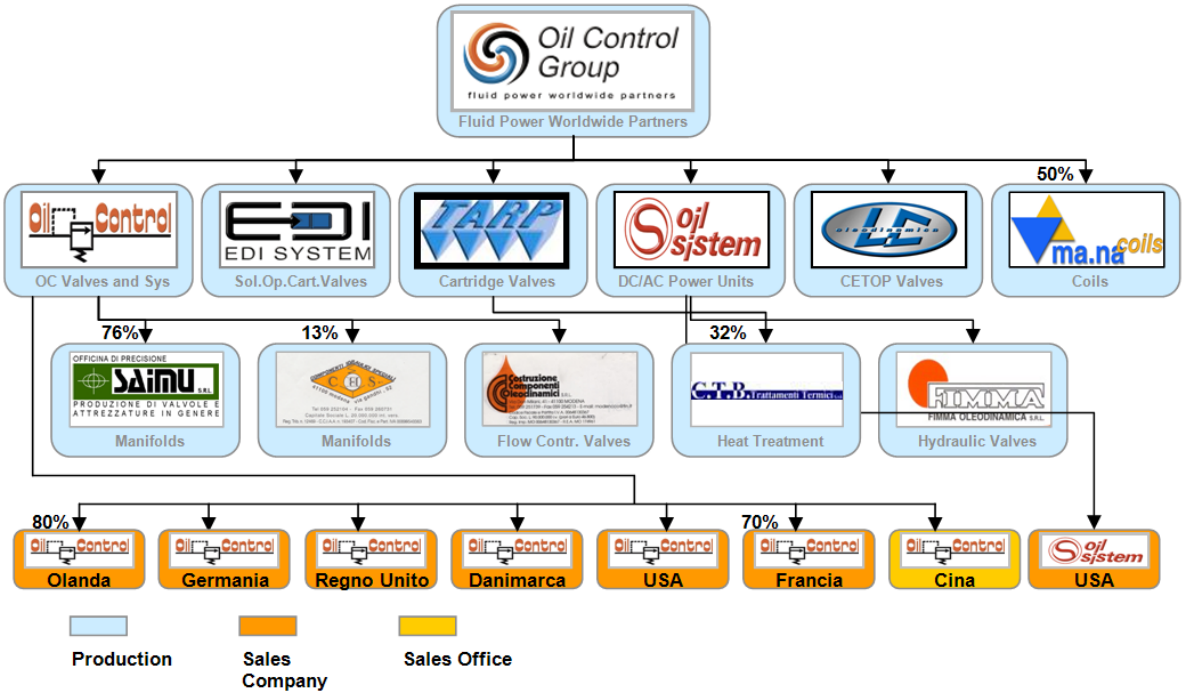

Figura 1.7: Oil Control Group: la vecchia struttura aziendale

1.3.2 L'acquisizione

Nel 2003 arriva la proposta di acquisizione del gruppo da parte della Bosch Rexroth, decisa ad acquisire il gruppo modenese alla luce del suo posizionamento strategico nel settore delle valvole a cartuccia. Il management deve prendere una decisione vitale per il futuro dell'azienda: Oil Control Group è una realtà sana e dinamica, capace di andare avanti con le proprie gambe ancora per lungo tempo, ma dall'altra parte c'è un'azienda come Rexroth che può amplificare, forte di una rete commerciale senza paragoni, il successo dei prodotti Oil Control.

Il dilemma si risolve il 21 dicembre 2004: Oil Control Group e Bosch Rexroth si danno appuntamento davanti al notaio, dove viene sottoscritto il passaggio del 67,3% delle azioni Oil Control Group S.p.A. nelle mani di Bosch Rexroth. Percentuale destinata a salire, nel corso degli anni, fino al 100%.

Dal 2005 il gruppo ha cambiato il nome in ***Bosch Rexroth Oil Control S.p.A.*** come frutto dell'acquisizione, dove sono serviti due anni per poter mettere a punto il progetto, non tanto per problemi legati a questioni finanziarie, ma piuttosto per poter verificare nei minimi particolari il nuovo piano industriale. L'acquisizione della maggioranza delle quote della società Oil Control Group è avvenuta con riserva dell'autorizzazione delle Autorità Antitrust.

Figura 1.8: I diversi plant appartenenti al gruppo

Logo	Nome divisione / Luogo	Breve identificativo
	Divisione Oil Control Nonantola / Modena	(Nnt e Mda2)
	Divisione Edi System Modena	(Mda1 e Mda3)
	Divisione TARP Pavullo nel Frignano	(Pvo 1, 2, 3, 4, 5, 6)
	Divisione LC Oleodinamica Vezzano sul Crostolo	(Vzz)
	Divisione Oil Sistem Reggio Emilia	(Rge1)
	Divisione Fimma Reggio Emilia	(Rge2)

Mappa

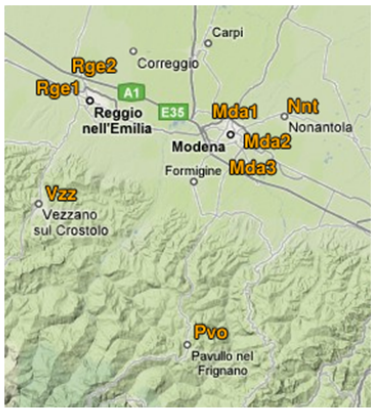

Figura 1.9: La locazione geografica dei plant

Dopo il nullaosta delle autorità antitrust, arrivato dopo quattro mesi dalla firma del contratto, si è avviata la procedura di integrazione post-acquisizione. Così tutte le aziende gravitanti nell'orbita Oil Control Group S.p.A. sono convogliate in un'unica entità, la Bosch Rexroth Oil Control S.p.A.; al contempo Oil Control si è dovuta integrare in Bosch Rexroth e in tutte le realtà locali di proprietà del gruppo tedesco.
L'esperienze acquisite negli anni hanno permesso all'azienda di poter avere un'organizzazione molto efficiente, nonostante le dimensioni, e la possibilità di poter soddisfare le richieste dei clienti/aziende molto più grandi e strutturate meglio; la partnership con la Rexroth ha portato in più la possibilità di poter ampliare i processi di internazionalizzazione e la dimostrazione che la sede di Nonantola sia ritenuta affidabile sul piano organizzativo è stata la scelta di farla diventare la *sede della divisione Rexroth per il settore dell'idraulica compatta*, divenendo così la prima azienda che è stata scelta al di fuori dal territorio tedesco.

L'ingresso di Rexroth non ha comportato una rivoluzione culturale all'interno delle varie aziende che componevano la Oil Control Group, giacché i valori di base (qualità, servizio, legalità) erano i medesimi, e sotto questo punto di vista non c'è stata molta difficoltà per procedere verso l'integrazione. Per quanto riguarda l'organizzazione interna, viceversa, molto è cambiato e molto resterà da fare nei prossimi anni per attivare le enormi sinergie possibili tra le due aziende. Sinergie di mercato, naturalmente, vista la grandissima diffusione della rete commerciale Bosch Rexroth e la sua presenza che virtualmente si estende ad ogni OEM (Original equipment manufacturer) esistente sul pianeta, ma anche sinergie di prodotto, di marketing, di ricerca e di logistica.

1.3.3 L'integrazione

Ovviamente, Bosch ha un altro livello dimensionale ed attraverso il marchio Rexroth gestisce oltre 500mila clienti in più di 80 Paesi, offrendo tutte le tecnologie-chiave per l'azionamento, il controllo e la movimentazione, passando per l'idraulica e la pneumatica fino all'elettronica così come il relativo servizio di assistenza. Con l'acquisizione la Oil Control Group ha potuto rafforzare la posizione nell'idraulica compatta, un mercato in forte crescita. Infatti il settore dell'idraulica compatta sta vivendo anni di ottime performance, e stando alle previsioni il mercato sarà caratterizzato da tassi di crescita elevati e superiori alla media anche nei prossimi anni. Questo spiega l'interesse di Bosch per il gruppo modenese che negli ultimi anni ha potuto fornire i propri prodotti alle maggiori aziende nel campo della meccanica.
L'assoluta centralità del reparto R&S (ricerca e sviluppo) e la dinamicità della gestione commerciale, consentono presto all'azienda di guadagnare quella posizione di leadership sul mercato domestico che le permetterà di dedicarsi allo sviluppo dei mercati esteri, primi fra tutti quelli europei. Negli ultimi anni il gruppo ha esteso il suo know-how tecnico, spostandosi oltre le tradizionali valvole di bilanciamento per gru e piattaforme aeree, per fornire anche soluzioni con valvole capaci di rivolgersi a tutto il mercato dell'idraulica mobile.
Forte della partnership tecnologica che andava consolidandosi con grossi costruttori di macchinari, Oil Control cresce al punto di rendere presto necessaria la creazione di filiali estere dedicate al servizio ed allo sviluppo dei rispettivi mercati. Nello stesso tempo si espandono le aree di interesse dell'azienda e, oltre che alle valvole overcenter applicate alle gru ed alle navicelle aeree, si dedica con successo alle innumerevoli applicazioni relative al controllo dei carichi sul macchinario mobile quali: escavatori, argani, carrelli sollevatori, ecc.

1.3.4 La situazione attuale

Il 2008 entrerà nella storia economica come uno degli anni più turbolenti. Nei primi mesi l'azienda è riuscita ad aumentare notevolmente il fatturato, ma dalla primavera, tuttavia, la crisi finanziaria ha toccato l'economia reale con tutta la sua forza, colpendo Bosch, Bosch Rexroth e Bosch Rexroth Oil Control. Proprio l'industria automobilistica e quella edile, importanti clienti finali dei prodotti fabbricati in azienda, hanno registrato un crollo repentino che ha significato una riduzione notevole delle ordinazioni e questo ha avuto un impatto anche sui livelli occupazionali.
In questa situazione l'azienda ha reagito in modo rapido e consistente per abbassare i costi e garantirne il futuro stesso. Indipendentemente dalle oscillazioni a breve termine, i settori di attività hanno la possibilità di crescita e potenziali a medio e lungo termine poiché Rexroth ha una buona posizione ed è parte di un gruppo forte con strategie pensate per il lungo termine.

1.3.5 I prodotti

La scelta di effettuare la fusione con Oil Control Group è stata fatta con l'obiettivo di poter presentare una gamma completa in grado di soddisfare tutte le esigenze nel campo delle valvole idrauliche: compatte, complete, economiche divenendo così l'unica azienda in grado di offrire una gamma di prodotti ampia e completa.
La famiglia dei Compact Hydraulics è composta da:

- **Cartridge Valves**;
- **Integrated Circuits**;
- **Load Holding / Motion Control Valves**;
- **Power Modules - Mini Power Packs**;
- **Compact Directional Valves**.

Rexroth vanta di una gamma completa di prodotti per l'idraulica compatta: valvole proporzionali, distributrici e limitatrici della pressione, valvole smorzatrici e numerosi altri componenti per l'impiego economico di circuiti idraulici in spazi ristretti dove sono ampiamente impiegate nei veicoli commerciali e nelle macchine operatrici mobili.
Rexroth non solo copre le attuali esigenze standard del mercato, ma con la serie *High Performance* si apre a nuove sfide. Il know-how combinato di Rexroth e Oil Control si riflette anche nella progettazione e montaggio di blocchi di comando compatti ed economici.
Le tendenze di sviluppo nell'idraulica compatta sono caratterizzate dall'esigenza di ottenere pressioni di lavoro sempre più elevate con ingombri sempre più ridotti.

Figura 1.10: I diversi prodotti offerti dal gruppo

Figura 1.11: Uffici e linee produttive della divisione Oil Control di Nonantola

La gamma dei prodotti Rexroth e Oil Control sono integrabili fino a formare un sistema modulare completo. Per i requisiti standard attuali (pressione di esercizio di 210 bar e due milioni di cambi di carico garantiti) il leader di mercato offre valvole distributrici, di limitazione e riduzione della pressione, sequenziali e smorzatrici, come anche numerosi tipi di valvole speciali. Tutte le valvole Cartridge sono sviluppate secondo lo standard industriale UNF. Le valvole smorzatrici, inoltre, sono disponibili anche per altri tipi comuni di fori filettati.
Con l'acquisizione della Oil Control Group, Rexroth ha nettamente ampliato le capacità per la progettazione "su misura" di blocchi di comando compatti ed economici. Rexroth fornisce sia singoli componenti sia blocchi di comando, moduli e gruppi completi e pronti per il collegamento. La rete mondiale di distribuzione e assistenza di Global Player assicura un'elevata disponibilità dei prodotti e del know-how per le applicazioni.
Dalla mappa in figura 1.9 si può notare come la società si è potuta "ingrandire" acquisendo molte aziende del territorio, aziende fortemente specializzate nel settore e non a caso sono presenti tutte in Emilia Romagna poiché, soprattutto le provincie di Reggio, Modena e Bologna, è considerata la terra dei motori in Italia con molte aziende meccaniche, anche di fama a livello internazionale.
Attualmente le principali divisioni produttive del gruppo Bosch Rexroth Oil Control sono raffigurate in figura 1.13.

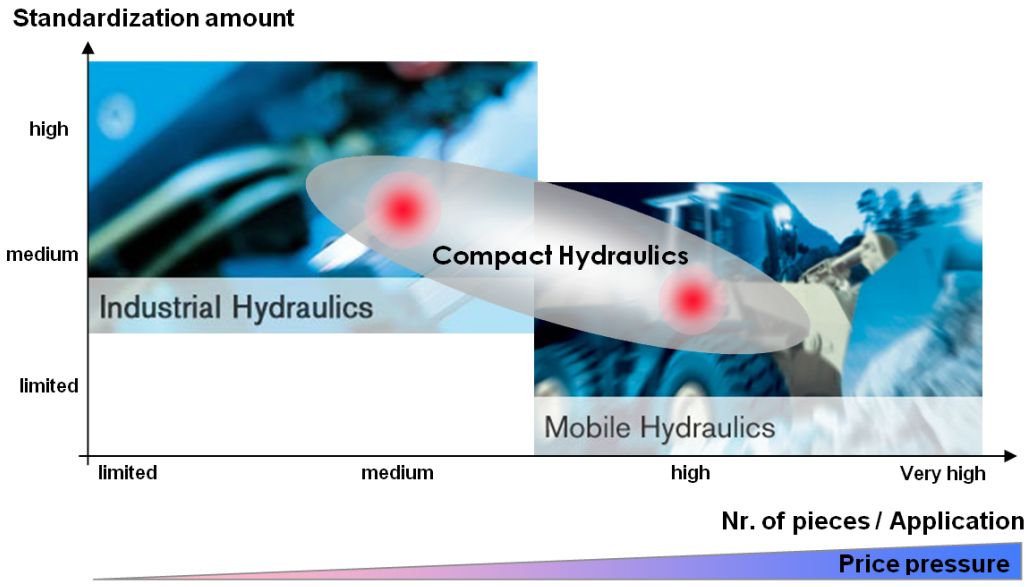

Figura 1.12: Settore dei prodotti realizzati

SEDE CENTRALE - NONANTOLA			
Divisioni	**Linee di Prodotto**	**Sedi territoriali delle Divisioni**	
Divisione Oil Control	Motion Control	Nnt, Mda2	Nonantola, Modena
Divisione Edi System	Electric Cartridge Valves Integrated Circuits	Mda1 Mda3	Modena
Divisione Oil Sistem/Fimma	Compact Power Packs Inline Valves	Rge1, Rge2	Reggio Emilia
Divisione TARP	Mechanical Cartridge Valves	Pvo 1/6	Pavullo nel Frignano
Divisione Oleodinamica LC	Compact Directional Valves	Vzz	Vezzano sul Crostolo

Figura 1.13: Le linee di prodotto dei diversi plant

1.3.6 I reparti aziendali: il reparto ISY

Il progetto di fusione attuato da Bosch Rexroth, che in quasi due anni ha portato otto aziende a costituire un unico gruppo, ha richiesto molto lavoro da parte del management aziendale, soprattutto per quanto riguarda la struttura organizzativa. Allo scopo di rendere più flessibile il gruppo si è optato per una forma organizzativa divisionale, conferendo ad ogni divisione la responsabilità e specializzazione per un prodotto visti in precedenza. La decisione di adottare una forma di tipo divisionale, mantenendo la struttura di ogni singola società, permette al Gruppo di recuperare i vantaggi della piccola dimensione d'impresa, mantenendo quelli tipici della grande impresa: *solidità*, *economia di scala*, *economia di scopo*, ecc.
Allo scopo di ridurre le differenze tra le divisioni e favorire il processo di integrazione, si è deciso quindi di adottare una struttura divisionale accentrata, caratterizzata da:

- presenza di interdipendenze sequenziali tra le divisioni;

- grado di decentramento divisionale ridotto: la direzione centrale non solo pianifica la strategia e lo sviluppo delle singole divisioni ma orienta e coordina in modo direttivo i comportamenti. Si è optato infatti per una centralizzazione della Direzione e delle varie funzioni, che servono tutte le divisioni, delocalizzando solo le attività di produzione, contabilità clienti e fornitori e gli uffici commerciali. Le funzioni centrali hanno un ruolo pervasivo, dato che spetta loro il compito di fornire linee guida delle politiche funzionali. Dal polo amministrativo di Nonantola, presso le sedi della divisione Oil Control, dipendono tutte le divisioni della Bosch Rexroth Oil Control S.p.A.

L'azienda è composta da diversi reparti aziendali che collaborano insieme per il conseguo della *mission*. Tutte le divisioni hanno l'organigramma in comune, dove ogni funzione è descritta da un'abbreviazione in inglese, necessaria per l'identificazione delle persone all'interno del gruppo Bosch. La struttura attuale dell'organigramma non è definitiva, ma è soggetta a modifiche visto che l'azienda non ha ancora raggiunto il suo assetto definitivo.
I diversi reparti sono visibili nella figura 1.14 con il relativo acronimo. Qui si possono notare le tre funzioni principali direttamente dipendenti dal comitato esecutivo aziendale:

- ***General Manager*** (***GM***);
- ***Vice President Engineering and Manufacturing*** (***TE***);
- ***Vice President Finance and Controlling*** (***FC***).

GM, TE e FC formano il Board aziendale e sono responsabili di ogni attività svolta; in particolare hanno la responsabilità primaria di coordinare lo sviluppo dei processi aziendali, della divulgazione della "Politica della Qualità e dell'Ambiente & Sicurezza" e del riesame del Sistema di Gestione Qualità e Ambiente (S.G.Q.A) adottato, verificandone l'adeguatezza ed efficacia. Inoltre il Board ha competenze organizzative e decisionali per tutto ciò che riguarda le strategie di sviluppo aziendale, le vendite, la produzione, gli acquisti, i miglioramenti, la Qualità, l'Ambiente e la Sicurezza.
Tale struttura organizzativa ha permesso di rendere il processo decisionale e il trasferimento delle informazioni più veloce, permettendo una chiara definizione degli obiettivi del management, che grazie ad un adeguato supporto di controlli, ha rinforzato i comportamenti coerenti, riducendo la conflittualità ed aumentando l'integrazione tra le divisioni. Tale processo è stato avvantaggiato dalla stesura delle procedure aziendali interne, volte a descrivere e spiegare ciascuna attività svolta. Le procedure, aggiornate continuamente, permettono alla Direzione di controllare al meglio qualsiasi funzione, assicurandosi così l'applicazione e rispetto delle strategie decise a monte. Le procedure interne vengono

predisposte sulla base delle direttive centrali emesse dalla casa madre. Per le materie più significative e di comune interesse si applicano direttamente le direttive centrali. Tali procedure sono sottoscritte da un amministratore e disponibili, in versione completa e aggiornata, sulla rete intranet dell'azienda. Tutti i dipendenti sono tenuti a leggerle ed applicarle, nello svolgimento delle proprie attività.
Il rispetto delle procedure è ulteriormente garantito da una fitta rete di costanti controlli interni, attivi ad ogni livello della struttura organizzativa, volti ad escludere i rischi che possono nascere da errori umani e/o interventi non corretti, che potrebbero così portare danno al patrimonio dell'azienda.

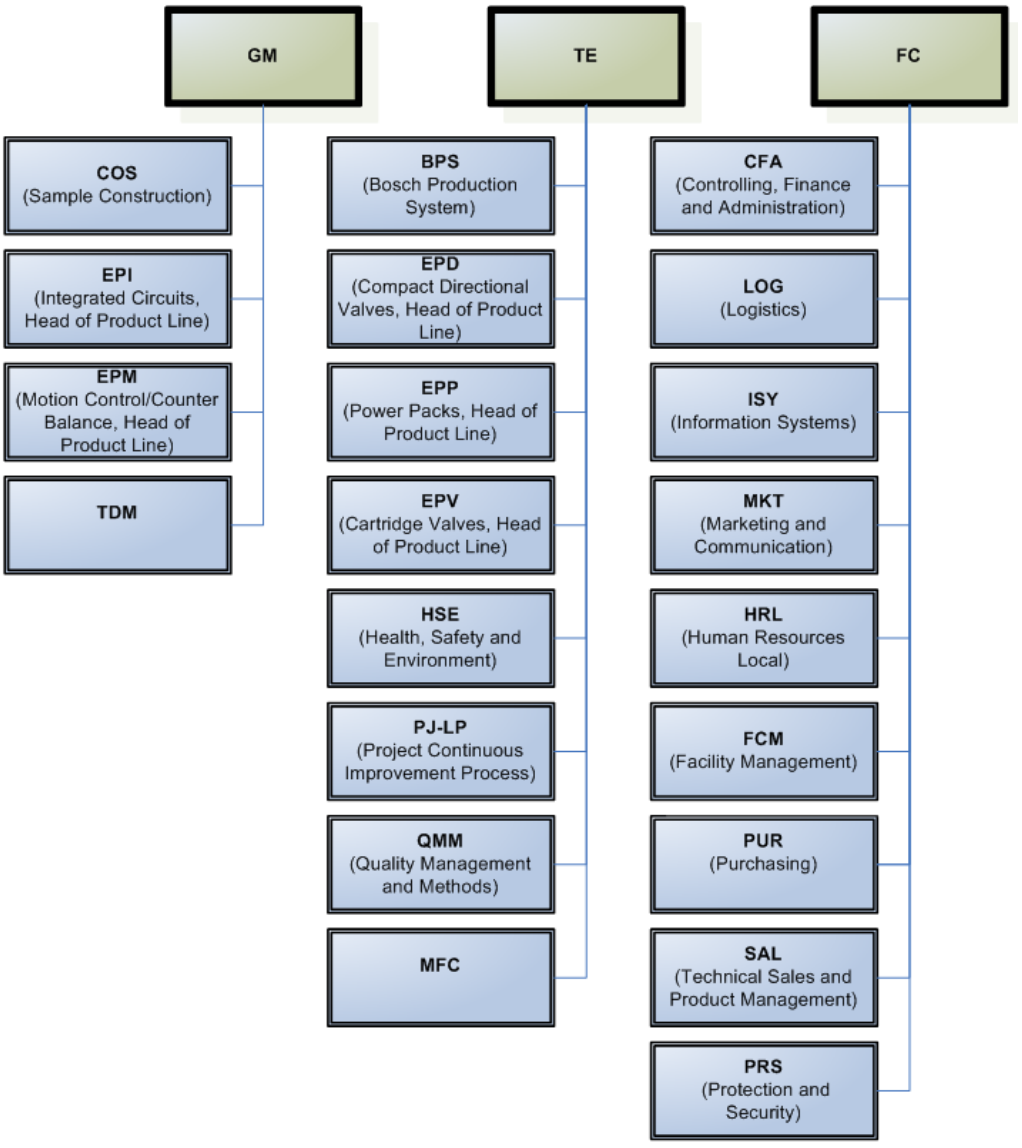

Figura 1.14: Organigramma aziendale (Marzo 2009)

Dai vari reparti presenti in azienda ci concentreremo sul reparto *ISY* (reparto dove è stata svolta l'esperienza lavorativa), in particolare sui sistemi software utilizzati principalmente

dalla produzione (come ad esempio l'inserimento dei dati relativi al magazzino e l'avanzamento della produzione) e la gestione dei dati sull'anagrafica articoli, sul ciclo passivo, sul ciclo attivo.
L'ente ***ISY*** (***Information System***) si occupa principalmente di:

- **Server/Help desk**:
  - Assistenza agli utenti per problematiche legate ad hardware/software;
  - Gestione hardware/software;
  - Installazione e configurazione di client/server;
- **Networking**:
  - Gestione degli apparati di rete e dei sistemi fonia;
- **Applicazioni Web**:
  - Pubblicazione e gestione intranet e sito internet aziendale;
- **Applicazioni gestionali**:
  - Attività di supporto alle varie funzioni aziendali sui sistemi gestionali;
- **Coordinamento generale progetti e investimenti**;
- **Data Security e Privacy**:
  - Formazione degli incaricati del trattamento sui rischi che possono compromettere la sicurezza e la privacy dei dati;
  - Descrizione delle misure di sicurezza disponibili per prevenire eventi dannosi;
  - Pianificazione e implementazione misure di sicurezza.

L'ente è composto da tre uffici collocati nei stabilimenti di Oil Control (Nonantola e Modena) e di Edi System (Modena). L'ufficio principale, dove risiede la maggior parte del personale, è nella sede di Nonantola, mentre gli altri uffici sono solo di "appoggio" quando si interviene nelle rispettive divisioni.
Il reparto ISY a sua volta è composto da due sottogruppi, ISY1 e ISY2, dividendo così il lavoro per poter meglio svolgere le attività richieste: ISY1 è rivolto alla gestione delle infrastrutture Hardware/Software, delle telecomunicazioni ed help desk agli utenti, mentre ISY2 è rivolto alla gestione ed assistenza delle applicazioni.
I responsabili ISY ed i loro collaboratori dispongono dell'accesso "amministrativo" al sistema gestionale per poter supportare le utenze in real time durante la giornata lavorativa. Tutti gli altri utenti hanno accesso limitato in base alle loro funzioni.

In azienda si seguono diversi principi per evitare che un singolo utente possa svolgere attività che potrebbero essere potenzialmente critiche per l'azienda, come il:

- **principio della doppia firma**: qualunque atto impegnativo della società deve essere firmato da due procuratori, aventi adeguati poteri conferiti dal Consiglio di Amministrazione. I procuratori possono scrivere atti vincolanti per l'azienda solo a firma congiunta con un altro procuratore, in base alle aree di attività e per determinati limiti d'importo.
- **principio dei quattro occhi**: la funzione controllante deve essere diversa dalla funzione che predispone il documento o che svolge l'attività per verificarne la correttezza. Tale principio è volto ad evitare che decisioni vincolanti per la società vengano prese da un solo soggetto, riducendo così il rischio di errori e abusi.

Il reparto ISY sarà integrato con il reparto informatico presente a Milano e il nome sarà cambiato in *CI* (*Corporate Sector Information Systems and Services*), reparto riconosciuto direttamente dalla casa madre Bosch.

1.4 Il settore idraulico oleodinamico

Viene adesso introdotta una piccola definizione per poter individuare in modo chiaro il settore e le sue possibili applicazioni.
L'oleodinamica è branca dell'ingegneria meccanica che si occupa dello studio della trasmissione dell'energia tramite fluidi in pressione, in particolare l'olio idraulico.
La portata d'olio generata da una pompa all'interno di un circuito oleodinamico viene utilizzata per muovere un martinetto o un motore idraulico a seconda che l'effetto meccanico desiderato (forza o movimento) sia lineare o rotatorio. Un classico attuatore lineare oleodinamico è il cilindro, costituito da una camicia in cui scorre un pistone, il quale spinge uno stelo che esplica il moto. Per il moto rotatorio basti pensare alle ruote delle macchine di movimento su terra come gli escavatori o grandi trattori agricoli, oppure pensare agli argani per issare le reti dei pescherecci dove servono coppie elevate e solitamente velocità angolari modeste.
Il settore oleodinamico è in forte espansione a livello mondiale grazie alla sua grande capacità di gestire grandi potenze tramite componentistica di dimensioni e pesi ridotti rispetto a tecnologie alternative. L'Italia occupa un ruolo di punta nel mercato europeo ed è tra i primi cinque produttori mondiali di componenti oleodinamici.
In oleodinamica l'olio è un componente che interagisce con tutti i componenti del circuito. Il suo ruolo principale è quello di trasportare l'energia dal generatore all'utilizzatore, ma non va dimenticata la sua importante funzione lubrificante e di asportare calore, che evita

l'usura e l'installazione di ingombranti sistemi di raffreddamento per i componenti del circuito.
Sul mercato esistono diversi tipi di fluidi di lavoro che vengono scelti seguendo le caratteristiche: viscosità, capacità lubrificante, resistenza all'invecchiamento, igroscopicità, elevato punto di fiamma, bassa nocività.
Il comparto oleodinamico ha un ruolo fondamentale quale fornitore di componenti per importanti settori della meccanica strumentale regionale, come quelli delle macchine agricole, delle macchine movimento terra, delle macchine per il sollevamento e la movimentazione e delle macchine operatrici per l'industria estrattiva, edile e manifatturiera.
Alcune possibili applicazioni sono utilizzate nei seguenti settori:

- ***Aeronautica***: flap, carrelli, timone, freni;
- ***Marina***: sterzatura, verricelli, pinne stabilizzatrici, calettatura elica;
- ***Veicoli***: freni, sterzo, sospensioni attive;
- ***Macchine utensili***: presse oleodinamiche;
- ***Macchine movimento terra***: bracci di gru, ruspe, trattori;
- ***Sistemi ferroviari***: deviatori per linee ad alta velocità.

Il principale svantaggio dell'oleodinamica rispetto alla pneumatica (trasferimento di forze mediante l'utilizzo di gas in pressione, molto spesso aria compressa) è l'utilizzo di un fluido di lavoro potenzialmente inquinante in caso di perdite delle guarnizioni o di errato smaltimento, ed è per questo motivo che si stanno diffondendo sistemi oleodinamici di nuova generazione basati su fluidi diversi, come acqua o oli speciali.
Il comparto oleodinamico nazionale si è sviluppato solo a partire dall'inizio degli anni '50. Il settore nazionale è caratterizzato da concentrazioni territoriali di imprese, specializzate in settori maturi, con prezzi competitivi e dotate di elevata flessibilità produttiva e capacità di soddisfare la domanda di applicazioni speciali e di alta qualità. Le imprese nazionali hanno minore dimensione rispetto a quelle dei maggiori produttori mondiali (Germania, U.S.A. e Regno Unito). Questi paesi sono i principali competitori sui mercati europeo e americano, mentre il Giappone lo è sui mercati asiatici.
L'industria oleoidraulica e pneumatica italiana mostra una buona apertura verso i mercati esteri e il mercato interno fa ampio ricorso alle importazioni. L'ampia e diversa specializzazione delle produzioni nazionali, quella italiana è orientata verso serie corte e applicazioni speciali, spiega questo notevole commercio internazionale di prodotti dell'industria oleoidraulica e pneumatica. Le esportazioni forniscono comunque una buona copertura delle importazioni del settore. Le aziende italiane del settore sono in prevalenza

localizzate in Lombardia e in Emilia-Romagna.
Il settore oleoidraulico costituisce la componente principale dell'industria; il settore ha una buona propensione all'esportazione, dove i mercati europei costituiscono lo sbocco principale, anche se una buona quota delle esportazioni raggiunge l'Asia e l'America. Gli U.S.A. compaiono tra i principali paesi di sbocco dopo i maggiori paesi europei.
Il settore pneumatico ha una minore propensione all'esportazione rispetto all'oleoidraulico. I mercati europei costituiscono anche per esso la principale area di sbocco, ma rispetto al settore oleoidraulico, l'Asia e l'America assorbono quote superiori delle esportazioni, come è evidenziato dall'analisi dei principali paesi di destinazione delle esportazioni del settore.

L'industria meccanica ha da sempre un ruolo di primo piano in Emilia-Romagna: attraversando in maniera trasversale tutte le filiere produttive della regione, sviluppa innovazioni di processo e di prodotto essenziali a campi industriali specifici. Legato a doppio filo a una tradizione di carattere artigianale che trova sbocco nelle vocazioni produttive territoriali, ma anche alla presenza di grandi imprese storiche, il settore meccanico ha tratto forza anche dall'esistenza nel territorio di una rete di scuole tecniche, spesso in stretto contatto con le imprese che ha contribuito ad alimentare le competenze e il loro aggiornamento tecnologico attraverso la formazione dei giovani tecnici diplomati.
La quasi totalità delle imprese emiliano romagnole è localizzata nelle provincie di Reggio Emilia, Modena e Bologna. Il principale settore di sbocco della produzione regionale (valvole, distributori, pompe e motori a ingranaggi e cilindri) è dato dalle macchine mobili dove il settore è caratterizzato dal ruolo dei rivenditori, che progettano, assemblano e personalizzano i prodotti secondo le esigenze della clientela.
L'industria meccanica ha una presenza rilevante anche in termini di occupazione: nel settore si possono contare circa 340mila addetti, che incidono sull'economia regionale molto più di quanto succede nel resto del Paese.

Capitolo 2

System Landscape

2.1 Le aree di interesse dei sistemi trattati

Verrà ora fatta una breve panoramica dei sistemi informativi utilizzati all'interno delle divisioni DCOC (Drive Control Oil Control), sia per quanto riguarda sistemi in aiuto alla produzione ed al magazzino, sia per quanto riguarda i classici sistemi gestionali.
Ovviamente i programmi utilizzati, o alcune funzioni degli stessi, sono prettamente dipendenti dai diversi reparti DCOC, ed all'interno dello stesso reparto ci possono essere diversi diritti/accessi ai software.
In azienda non è necessario che ogni utente acceda a tutte le informazioni, condivise o no, che sono utili per la gestione dell'azienda e la produzione. Infatti ad ogni utente è applicato un profilo che definisce le autorizzazioni ai dati che può visualizzare e/o modificare. Oltre ad eventuali problemi di sicurezza sui dati, nessun utente ha bisogno di accedere a tutti i dati in quanto il molteplice accesso può confondere l'utente e ridurre il business value della condivisione delle informazioni.
Ecco perché ogni utente per poter accedere ai diversi sistemi ha bisogno di essere autorizzato dal responsabile dell'ente in cui lavora e dell'attivazione dell'accesso al sistema da parte del reparto ISY attraverso la creazione del profilo utente e l'associazione dello stesso in un gruppo di appartenenza che ne definisce i diritti. In base al gruppo di appartenenza ogni utente sarà abilitato a poter svolgere determinate funzioni, mentre altre no. Ciò non toglie la possibilità che si possono aggiungere/negare funzioni in base a specifiche esigenze.

Prima di introdurre le caratteristiche di ogni sistema informativo utilizzato, viene ora data una breve descrizione del campo in cui lavorano i suddetti sistemi.
Le attività della divisione Oil Control riguardano la progettazione, produzione, vendita e assistenza, di valvole e gruppi integrati multifunzionali per impieghi oleodinamici: quindi durante tutto il ciclo produttivo di ogni singolo pezzo intervengono molti reparti del grup-

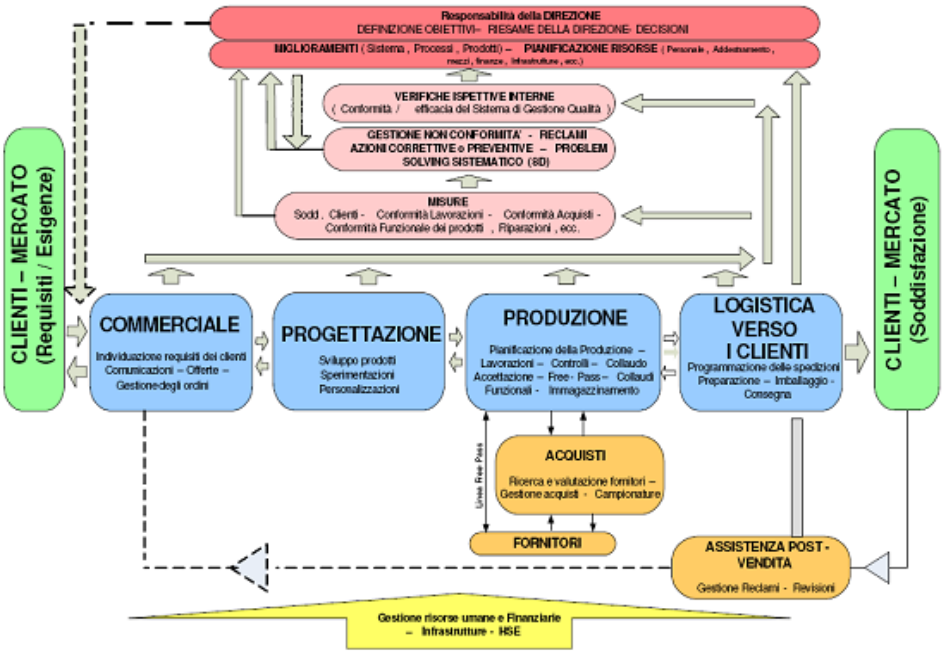

Figura 2.1: Mappatura dei processi DCOC

po. In figura 2.2 si può notare come sono correlati tra loro i diversi reparti della divisione di Nonantola, e appare subito come i reparti lavorano in stretto contatto e come ci sia uno scambio di informazioni abbastanza importante.

Nel sistema produttivo rientrano in gioco diversi concetti rilegati all'ambito industriale che si sono sviluppati nel corso degli anni, e qui ne viene data una breve descrizione partendo dai sistemi PLM fino alla pianificazione effettuata dai sistemi ERP. Ovviamente non si intende dare una spiegazione esauriente visto che non è il tema principale trattato in questa tesi.

La gestione di avanzamento degli ordini di produzione è gestita del tutto attraverso in sistema informativo dove, sia il personale addetto alla realizzazione dei componenti, sia le macchine automatiche, aggiornano in modo costante tutte le attività svolte, dal prelievo del materiale necessario alla realizzazione fino alla fine della loro attività e/o del lavoro sul componente. Il personale utilizza diversi computer ed attraverso il sistema informativo riescono a gestire le informazioni; tale sistema non è indipendente, ma si interfaccia anche con gli altri sistemi presenti in azienda, ma lo si vedrà in seguito. Invece i macchinari automatici aggiornano direttamente i dati attraverso l'apparecchiatura idonea: sulle macchine sono installati diversi componenti ed inviano i dati attraverso schede di rete sul sistema appropriato, rendendo così l'operazione completamente trasparente escludendo il lavoro manuale.

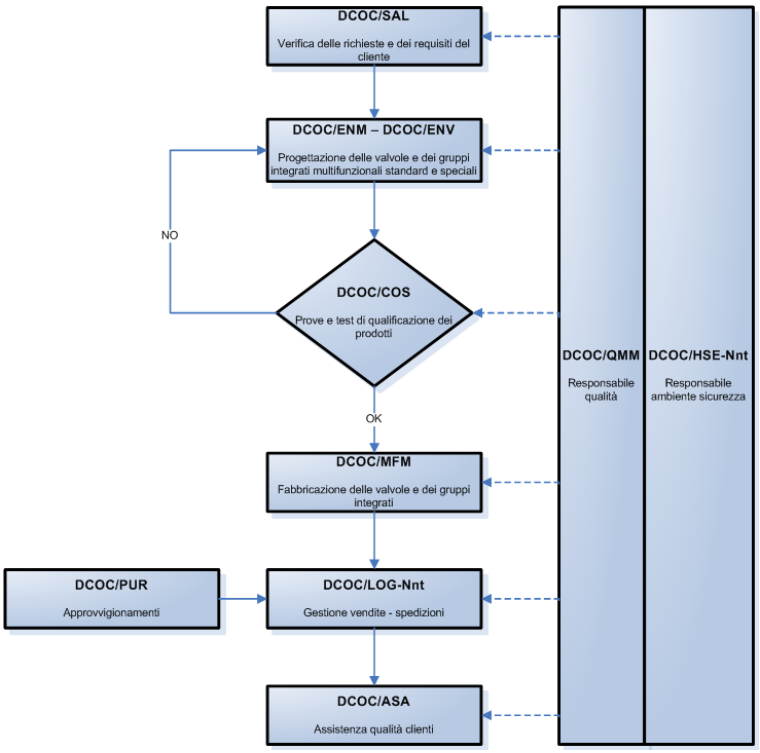

Figura 2.2: Attività svolte dalla divisione Oil Control

2.1.1 PLM (Product Lifecycle Management)

L'acronimo ***PLM*** [11] sta per ***Product Lifecycle Management*** (gestione del ciclo di vita del prodotto) ed è un approccio strategico alla gestione delle informazioni, dei processi e delle risorse a supporto del ciclo di vita di prodotti e servizi, dalla loro ideazione, allo sviluppo, al lancio sul mercato, al ritiro. Il PLM non è solo una tecnologia informatica, ma piuttosto un approccio integrato, basato su un insieme di tecnologie, su metodologie di organizzazione del lavoro collaborativo e sulla definizione di processi.
Il PLM si basa sull'accesso condiviso a una fonte comune da cui attingere informazioni e processi relativi al prodotto. È una strategia di affari che consente all'impresa estesa di apportare innovazione di prodotto durante tutto il ciclo di vita del prodotto, dalla fase di sviluppo all'obsolescenza, come se a operare fosse un'unica entità e creando un archivio di prezioso capitale intellettuale riutilizzabile in qualsiasi momento. In breve, il PLM è un supporto all'innovazione.
Il PLM viene frequentemente messo sullo stesso piano di altri approcci di business con cui è complementare (e per alcuni aspetti in sovrapposizione) come l'*Enterprise Resource Planning* (*ERP*), il *Customer Relationship Management* (*CRM*) e il *Supplier Relationship Management* (*SRM*).
Il PLM è composto da una serie di moduli che concorrono e collaborano allo sviluppo del prodotto; questi possono essere categorizzati come segue:

- ***Document Management***: gestione della documentazione tecnica (CAD/CAM/-CAE) e di progetto;
- ***Product Structure Management***: gestione della configurazione di prodotto (Struttura, BOM);
- ***Configuration management***: gestione delle varianti e dei lotti di produzione;
- ***Change management***: gestione dei cambiamenti di una o più entità che descrivono il prodotto;
- ***Workflow management***: strumento di gestione del flusso aziendale dei dati;
- ***Catalog Library***: gestioni dei componenti normalizzati e delle parti standard (viti bulloni, resistenze, ecc.);
- ***Supply Chain Management***: gestione dello scambio dati con i subfornitori.

L'implementazione di uno o più moduli in un sistema PLM dipende dal grado di integrazione che si vuole dare al processo produttivo.

2.1.2 MRP (Manufacturing Resource Planning)

La sigla ***MRP*** [12] identifica una tecnica per la pianificazione e il controllo della produzione: la sigla sta per ***Material Requirements Planning*** nella prima versione (***MRP I***) e per ***Manufacturing Resource Planning*** nella seconda (***MRP II***). Infatti il sistema è nato come metodo per la gestione dei materiali ed è stato poi esteso alla gestione delle risorse produttive. L'idea di fondo dei sistemi MRP, nati negli USA nei tardi anni '70, è quella di programmare l'approvvigionamento dei materiali su *fabbisogno*, ovvero sulla base degli ordini clienti certi o stimati. In precedenza le tecniche più diffuse, fin dal dopoguerra, erano di programmazione a *scorta*. Esse consentivano la costituzione di scorte ingenti di materie prime, semilavorati e prodotti finiti per fronteggiare variazioni impreviste della domanda e per garantire la massima efficienza e saturazione delle risorse. Negli anni '70 le aziende si trovarono a vivere in un mercato più competitivo, in cui assumevano via via maggiore importanza i tempi di consegna, la varietà e la qualità dei prodotti. D'altra parte a fattori di costo difficilmente contrastabili quali quelli di energia e di manodopera si aggiungevano gli oneri finanziari connessi all'esistenza di scorte elevate. Si rese allora sempre più necessario adottare una nuova organizzazione della produzione che evitasse di produrre in anticipo o in eccesso rispetto ai reali fabbisogni del momento, anche a scapito della saturazione delle risorse. Acquistare e produrre le quantità strettamente necessarie facendole giungere a destinazione solo nel momento in cui vengono utilizzate significa minimizzare l'entità delle scorte in lavorazione.

I sistemi MRP nascono e si diffondono solo nel momento in cui diventano disponibili degli strumenti informatici di supporto all'archiviazione di grandi moli di dati (sui prodotti, sulle distinte base e sulle situazioni di magazzino e di ordini) e al calcolo. Se è vero che i sistemi MRP consentono di adattare la produzione all'andamento sostanzialmente imprevedibile della domanda, è anche vero che consentono la generazione di scorte in anticipo o in eccesso, ottenute raggruppando tutti i fabbisogni di un periodo all'inizio del periodo stesso o dimensionando i lotti di ordine in funzione di quantità predefinite (lotti minimi e multipli).
Questo è alla base della definizione di MRP come sistema *push*, in quanto ciascuna attività è spinta a rispettare gli appuntamenti previste con le attività a valle: nulla impedisce però che l'attività a monte produca più pezzi di quanto impiegati in quella a valle, che verranno consumati solo in successivi piani. I sistemi MRP sono abbastanza diffusi, in forme più o meno semplificate.
In sintesi un sistema MRP è formato da diversi applicativi software (alcune centinaia) che supportano produzione ed approvvigionamenti, dalla programmazione operativa sino alla gestione dei flussi fisici dei materiali (oltre che, limitatamente, le operazioni fisiche delle officine). Non integra (o integra limitatamente) i processi di: vendita e distribuzione, processi amministrativi legati alla contabilizzazione degli acquisti e del magazzino materie prime.

2.1.3 ERP (Enterprise Resource Planning)

L'acronimo ***ERP*** significa ***Enterprise Resource Planning*** (letteralmente "pianificazione delle risorse d'impresa"). Si tratta di un sistema di gestione (sistema informativo) che integra tutti gli aspetti del business e i suoi cicli, inclusa la pianificazione, la realizzazione del prodotto (*manufacturing*), le vendite, gli approvvigionamenti, gli acquisti, la logistica di magazzino ed il marketing.
Con l'aumento della popolarità dell'ERP e la riduzione dei costi per l'ICT (Information and Communication Technology), si sono sviluppate applicazioni che aiutano i business manager a implementare questa metodologia nelle attività di business come controllo di inventari, tracciamento degli ordini, servizi per i clienti, finanza e risorse umane.
La prima versione dell'ERP metteva in collegamento diretto le aree di gestione contabile con l'area di gestione logistica (magazzini ed approvvigionamento); successivamente si sono iniziate ad implementare le relazioni interne anche con le aree di vendita, distribuzione, produzione, manutenzione impianti, gestione dei progetti, ecc.
Di grande importanza è il sistema MRP e la sua evoluzione MRP II (integrati nel sistema ERP) che permettono di programmare logiche di ordini automatici ai fornitori veramente sofisticate, tanto da tener conto dei tempi di consegna e di messa in produzione del prodot-

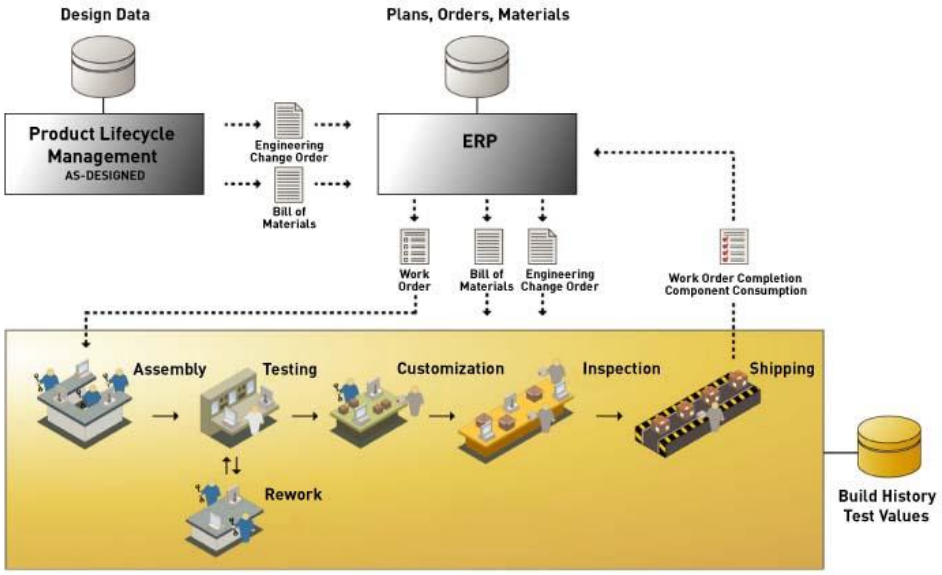

Figura 2.3: Schema semplificato di correlazione tra PLM e ERP [13]

to; questa metodologia permette di ottimizzare la rotazione dei materiali nei magazzini e la minimizzazione delle giacenze che impattano a livello contabile e fiscale.
Mentre gli MRP si preoccupano prevalentemente di supportare i processi primari di produzione (attività in linea di produzione, gestione degli approvvigionamenti dei materiali, ecc.), gli ERP affiancano a questo la gestione dei processi di vendita e di processi non direttamente legati alla produzione (contabilità, gestione del personale, ecc.). Sono comunque tutti sistemi formati da centinaia di software che insieme si occupano di gestire la mole di dati necessaria alla gestione dei processi aziendali. Una differenza sostanziale è che negli MRP i programmi utilizzano diverse porzioni di una complessiva base di dati, mentre gli ERP utilizzano un'unica base condivisa realizzando una più completa integrazione in grado di ridurre tempi e costi di gestione dei dati, di sviluppare una visione d'insieme più completa al fine di supportare meglio i processi decisionali delle funzioni aziendali preposte a questo.
Molti sistemi ERP includono moduli per la pianificazione di prodotto (MRP II), l'acquisto di parti e materiali, controllo dell'inventario, distribuzione dei prodotti, tracking degli ordini, finanza, contabilità, marketing e gestione del personale.

2.2 I sistemi informativi utilizzati in azienda

In un ambiente sempre più dinamico come quello odierno, le imprese si trovano in una situazione di grande complessità gestionale e nell'esigenza di dover gestire quantità sempre maggiori di informazioni in modo sempre più efficace, efficiente e tempestivo per poter così

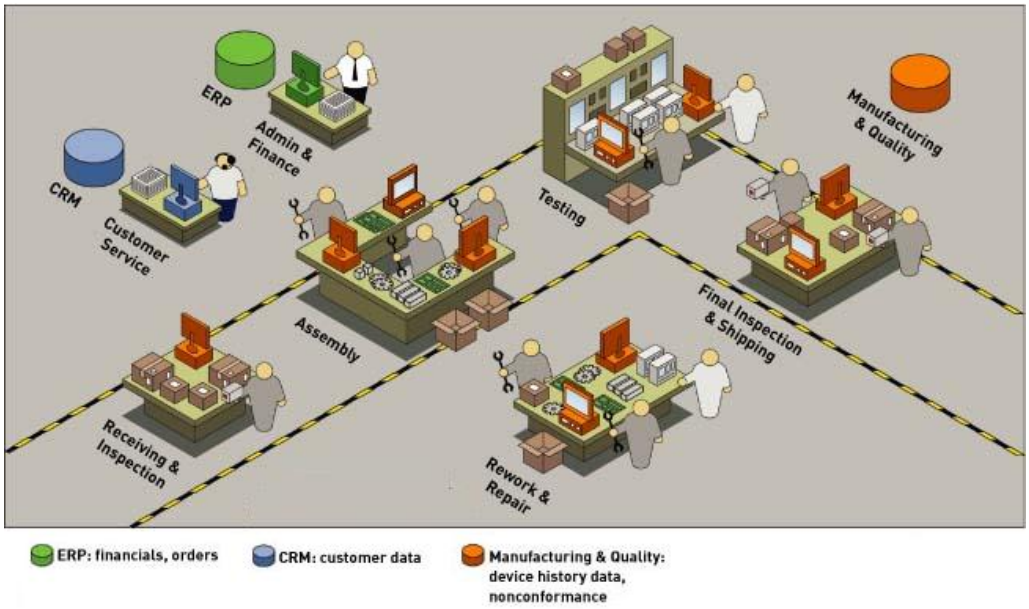

Figura 2.4: Supporto dei sistemi informativi lungo il processo produttivo [13]

rispondere ai continui cambiamenti del mercato e delle sue esigenze: prendere decisioni velocemente richiede la possibilità di disporre di tutte le informazioni necessarie in tempi rapidi, il che è possibile solo se l'impresa è dotata di un sistema informativo in grado di rendere disponibili le informazioni in tempo reale.
Possiamo dunque comparare il sistema informativo aziendale ad un vero e proprio sistema nervoso dell'azienda stessa dove ha il compito di raccogliere i dati, conservare i dati raccolti archiviandoli, elaborare i dati trasformandoli in informazioni e distribuire l'informazione agli organi aziendali utilizzatori.
Le tecnologie informatiche offrono oggi grandi potenzialità (informatizzazione del sistema informativo aziendale):

- consentono alle aziende di controllare, pianificare e gestire in modo integrato tutte le attività;
- consentono di elaborare velocemente una maggiore quantità di dati ed informazioni di quanto fosse possibile in passato.

Per fare questo il sistema informativo si può avvalere di tecnologie informatiche: la parte del sistema informativo aziendale che se ne avvale prende in nome di sistema informatico. Oggi, con il diffondersi delle tecnologie informatiche, il sistema informatico finisce per rappresentare la quasi totalità del sistema informativo, ma, almeno a livello concettuale, il sistema informativo non implica di per sé l'uso dell'informatica; del resto prima che fossero introdotte le tecnologie informatiche già esistevano sistemi informativi.

Le informazioni fornite dal sistema informativo agli organi aziendali sono necessarie agli stessi per assumere le decisioni e sono caratterizzate da:

- il contenuto (ossia la rilevanza per il destinatario e la correttezza intrinseca);
- il tempo nel quale sono rese disponibili;
- il luogo ove sono rese disponibili;
- la forma con la quale sono presentate.

I sistemi informativi utilizzati nelle divisioni Oil Control (Nonantola e Modena) sono:

- **Apache/GPS/MPS** (della MBM [14]) - *Produzione*, *Ciclo Attivo*, *Ciclo Passivo*, *Magazzino*, *Controllo di Gestione*, *Pianificazione*;
- **Nicim** (della Atomos [15]) - *Rilevamento Presenze*, *Raccolta Dati*, *Schedulazione*;
- **WHMS** (del gruppo Siemens [16]) - *Gestione Magazzino Automatico*;
- **SAP R/3 RAINBOW** (della SAP AG [17]) - *Gestione dell'attività e dei processi aziendali.*

Verso la fine del 2008 si è completata l'integrazione dei sistemi informativi di Nonantola e Modena2 (la ex Saimu), che fino a poco prima era presente fra le aziende il classico rapporto cliente-fornitore. Infatti Modena2 si comportava come un terzista che forniva la maggior parte dei collettori di alluminio ed una vasta gamma di valvole finite.
Lo scambio di informazioni tra i due stabilimenti era basato essenzialmente su ordini e bolle di consegna ed i sistemi, benché in parte comuni, non dialogavano tra di loro. Oggi invece Modena2 viene gestito in tutto e per tutto come se fosse un reparto della struttura di Nonantola superando i tanti problemi di molteplice natura che si sono incontrati durante tale integrazione, quali informatici, logistici, organizzativi e di risorse. La programmazione integrata della produzione dei due stabilimenti rende in primo luogo le aziende più flessibili, adatti a rispondere in fretta alle variazioni delle richieste dei clienti. Tra gli altri effetti importanti c'è la riduzione delle scorte di componenti, la fusione dei know-how produttivi dei due plants e l'eliminazione della necessità di mantenere dati duplicati.
Questa integrazione è un passo del processo di migrazione di tutti gli stabilimenti verso un sistema unico nel 2011.
Nella figura 2.5 si può facilmente notare come le varie aree funzionali sono raggruppate tra loro attraverso i diversi software. Questi software vengono utilizzati come sostegno ai diversi reparti, dalla produzione fino ai reparti prettamente finanziari.

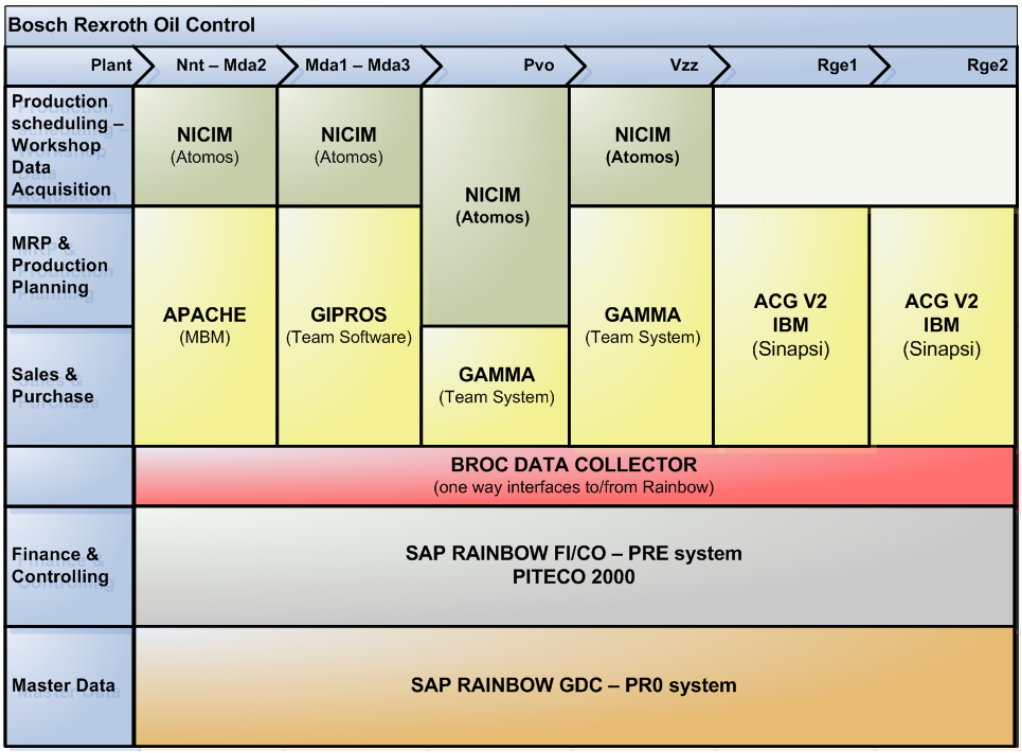

Figura 2.5: I sistemi utilizzati dalle diverse divisioni

Ovviamente sono stati indicati i sistemi più importanti ed utilizzati in azienda nell'ambito produttivo, trascurando tutto il reparto software che riguarda l'utilizzo quotidiano per la gestione di documenti (lettere, fogli elettronici, database, ecc.), di comunicazioni e telecomunicazioni, di gestione, ecc. In ambito produttivo sono importanti programmi come **SolidWorks** (utilizzato per la progettazione 3D CAD e costruzione dei componenti con una drastica riduzione dei tempi di progettazione e degli errori) e **Quarta** (soluzione informatica per il conseguimento del Total Quality Management della Blulink [18]), ma non trattati in questa esperienza lavorativa.

Poiché le varie aziende del gruppo sono nate tutte in modo indipendente, ognuna ha scelto i sistemi software da utilizzare in base alle proprie esigenze. Ma date le diverse acquisizioni si ha avuto il bisogno di avere dei sistemi omogenei tra loro, se non totalmente almeno parzialmente, andando a sostituire completamente i sistemi software utilizzati oppure utilizzando sistemi intermedi appositamente realizzati per poter interfacciare le diverse piattaforme (data middleware interface), come il **BROC Data Collector**. Inoltre, la migrazione da un sistema ad un altro è stato imposto anche dall'azienda casa madre Bosch, con l'obiettivo finale di avere sistemi collegati direttamente con la sede centrale in Germania.

Tuttora sono in corso diversi progetti di integrazione delle diverse piattaforme software verso i sistemi gestionali gestititi dalla casa madre Bosch Rexroth, come ad esempio il progetto **Rainbow** (versione SAP R/3 implementata appositamente per le esigenze del mondo Bosch), che andrà a sostituire il sistema ERP gestionale Apache, attualmente utilizzato solo dalla divisione Oil Control, per poter utilizzare in pieno i moduli del sistema SAP (previsto per il 2010/2011): attualmente sono utilizzati i moduli FI/CO (Finance & Controlling).
Questi tipi di progetti hanno una durata che può variare da pochi mesi fino ad anni interi, poiché è strettamente dipendente dalla priorità, dal tempo necessario per la completa riuscita e dai fondi destinati alla realizzazione.
L'integrazione non comprende solo il parco software, ma può essere effettuata anche tra i reparti; infatti il reparto ISY sarà integrato con il reparto informatico presente a Milano e il nome sarà cambiato in *CI* (*Corporate Sector Information Systems and Services*), reparto riconosciuto direttamente dalla casa madre Bosch.

2.2.1 Apache

Il sistema ***Apache*** (***Applications for Planning And Control of High Enterprises***) [14] è un'applicazione ERP della MBM Management Systems per le aziende di produzione. È un software gestionale integrato e modulare che offre copertura a tutte le esigenze delle aziende industriali e dispone di verticalizzazioni specifiche per vari settori merceologici (moda, alimentare, elettronica, meccanica). È basato su un'ipotesi di organizzazione aziendale conforme agli standard ISO.
L'applicazione copre tutte le aree gestionali:

- ciclo attivo;
- pianificazione/programmazione;
- ciclo passivo/produzione;
- magazzino;
- contabilità;
- controllo di gestione.

In figura alcuni dettagli delle aree appena descritte riguardo la produzione manifatturiera, il controllo di gestione, il ciclo passivo e il ciclo attivo.
Tra le principali funzionalità si possono evidenziare:

- Conformità agli standard ISO 9000 (definizione dei profili utenti, eventi gestiti per stato di avanzamento, storicizzazione delle variazioni);

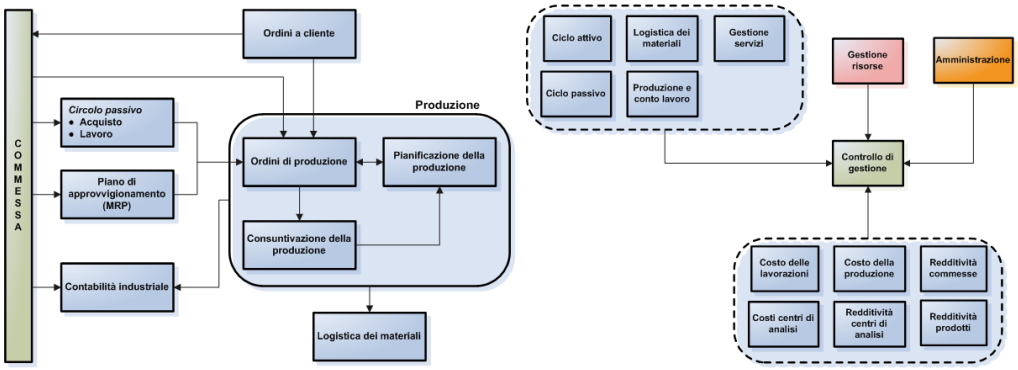

Figura 2.6: Produzione manifatturiera e Controllo di gestione

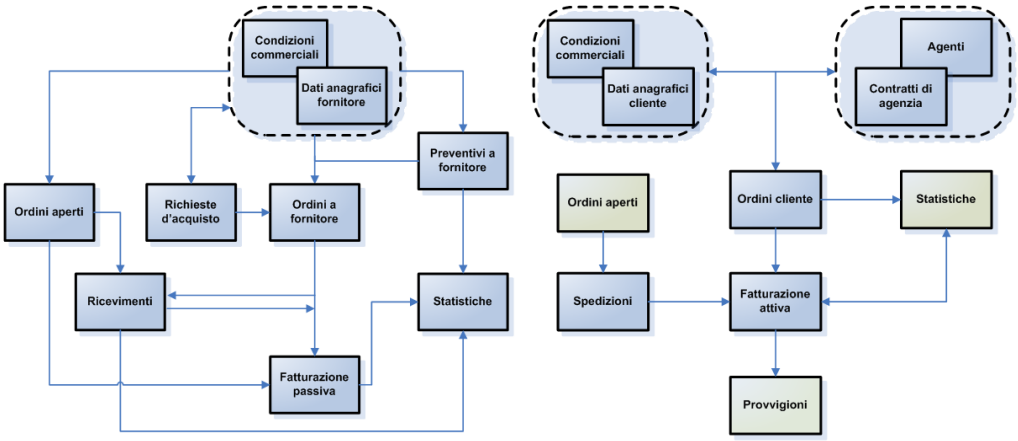

Figura 2.7: Ciclo passivo e ciclo attivo

- Tracciabilità e rintracciabilità dei lotti;
- Gestione dei dati della qualità;
- Disponibilità di strumenti di simulazione per la maggior parte delle aree applicative;
- Distinta base configurabile a qualsiasi livello di struttura;
- Disponibilità di un modulo per la pianificazione dei materiali integrato con la pianificazione delle risorse a capacità finita;
- Completa integrazione tra le lavorazioni interne e quelle esterne;
- SCM: integrazione via web con i fornitori di materiali e lavorazioni;
- Gestione avanzamento produzione con riferimento alle singole fasi;
- Possibilità di aggregare fasi di lavoro di ordini diversi per avanzarle contemporaneamente;

- Calcolo e controllo costi, calcolo automatico delle varianze;
- Gestione multi-lingua.

Oltre ad Apache, sono utilizzati i moduli GPS e MPS: sono moduli che prendono in esame alcuni dati presenti in Apache per poterli analizzare e poter dare possibili soluzioni di pianificazione. Sono moduli che gestiscono le autorizzazioni di accesso separate da quelle di Apache.

2.2.1.1 GPS (Global Planning System)

Il modulo *GPS* (*Global Planning System*) è un sistema di pianificazione dei materiali e delle risorse produttive.
Si basa sull'evoluzione delle metodologie MRP II integrate con un sistema PERT che consente di riposizionare nel tempo (pianificazione in avanti) gli eventi critici e tutti quelli da essi vincolati con logiche di priorità. Si è così in grado di evidenziare sia gli obiettivi non raggiungibili per le date richieste, sia l'entità del ritardo, evidenziando tutti gli eventi che concorrono a generare il problema (materiali e/o risorse).
Il sistema è costituito da un certo numero di maschere e grafici, realizzati con strumenti software adeguati, con i quali risulta molto semplice e veloce eseguire filtri e ricerche sui dati rappresentati, nonché attivare collegamenti dinamici tra finestre.
I moduli che costituiscono il sistema sono:

- **MPS** - *Master Production Schedule*;
- **MRP** - *Material Requirements Planning*;
- **PRM** - *Pegging Relationship Management*;
- **RCP** - *Raw Capacity Planning*;
- **CRP** - *Capacity Requirements Planning*;
- **FCP** - *Finite Capacity Planning*;
- **PDP** - *Purchasing Delivery Planning*;
- **Analyzer e Simulator**.

MPS (*Master Production Schedule*) è il piano principale di produzione che indica gli obiettivi aziendali in termini di quantità di prodotto finito da rendere disponibili in determinate date.
Consente, partendo dalle previsioni di vendita, la formazione di un piano produttivo compatibile con la disponibilità delle risorse e dei materiali.

Opera su più periodi temporali la cui ampiezza viene impostata parametricamente e sono possibili due livelli di dettaglio: per famiglia di prodotti e per articolo.
È disponibile un'interfaccia con l'ambiente commerciale per confrontare i piani in essere con il portafoglio ordini e per passare automaticamente da piani aggregati a piani per articolo.
Le principali funzionalità disponibili sono:

- creazione e gestione delle previsioni di vendita;
- formulazione e gestione del piano aggregato di produzione (per famiglia);
- generazione e gestione del piano dettagliato di produzione (per articolo);
- collegamento con il portafoglio ordini di vendita.

GPS Analyzer è un'applicazione sviluppata in ambiente Windows per analizzare i dati prodotti dal modulo GPS. Consente un'elevata operatività e fornisce all'utente un efficace strumento per visualizzare, analizzare e confrontare i dati tramite diagrammi e tabelle in sostituzione delle tradizionali stampe su carta.

2.2.2 WHMS

Il magazzino, con le movimentazioni e la gestione integrata di merci, scorte ed ordini, rappresenta l'anello fondamentale del processo di fornitura al cliente finale. Quindi serve un sistema che permetta di controllare tutti i flussi materiali che si instaurano nel magazzino, sostituendo in tutto o in parte la discrezionalità degli operatori, attraverso una regia centrale che genera le attività da fare (missioni), le ordina per priorità, le sincronizza nel rispetto delle regole organizzative individuate dai responsabili.
WHMS (***Warehouse Handling Management System***) [16] è sviluppato dal gruppo Siemens basandosi su prodotti standard di mercato (Data Base, SCADA, Interfaccia Operatore) e costituisce un semplice ed efficace strumento per la gestione del magazzino automatico, adattabile a diverse configurazioni di impianto e di modalità operative richieste, per incrementare le prestazioni e ridurre il costo totale lungo tutto il ciclo di vita degli impianti.
Il sistema, inteso come integrazione di hardware e software, è un sistema informatico dipartimentale per la gestione, in tempo reale, del magazzino fisico, con dispositivi operanti in radiofrequenza e si integra con il sistema informativo gestionale aziendale (ERP) e ne fanno uno strumento ideale per le soluzioni integrate di stoccaggio e movimentazione.
Principali funzionalità:

- depositi e prelievi, picking e collocazione fisica dei materiali ottimizzati ed assistiti via RF (frequency);

- classificazione ABC;
- rintracciabilità dell'unità di carico (unità di carico - vettore - cliente);
- spedizioni controllate;
- inventario, reportistica, integrazione con il sistema ERP.

Si caratterizza quindi come sistema esecutivo e di coordinamento delle attività di magazzino che si interfaccia con il campo (automazione, PLC) e/o direttamente con gli operatori (PC o sistemi RF). Con il termine picking intendiamo tutta l'attività di formazione ordini che possiamo garantirvi in tempo reale, limitando al minimo le possibilità di errore e l'allungamento dei tempi di evasione ordini.
Il sistema permette l'integrazione con applicativo Apache per il passaggio di:

- prebolle per generazione piani di prelievo per spedizioni;
- ordini di produzione per generazione piani di prelievo;
- anagrafiche articoli;
- anagrafiche fornitori.

La gestione informatica del magazzino offre vantaggi sul fronte dell'efficacia e dell'efficienza organizzativa; in particolare per l'efficacia delle operazioni:

- Riduzione degli errori di stoccaggio, prelievo e spedizione della merce ai clienti;
- Migliore utilizzo delle risorse di stoccaggio (allocazione ottima dei lotti);
- Migliore utilizzo delle risorse di movimentazione (operatori e mezzi);
- Maggior precisione nella programmazione delle attività (orari di preparazione rispettati);
- Aumento della qualità delle informazioni disponibili sui processi;
- Aumento della qualità delle informazioni disponibili sulla mappa di magazzino.

Mentre per quanto concerne l'efficienza:

- Aumento della velocità di preparazione degli ordini (attività guidate fino al punto di prelievo);
- Riduzione delle "rilavorazioni" per merce mancante o resi;
- Riduzione delle attività per condurre l'inventario fisico;

- Riduzione delle scorte minime dovuto alla buona qualità delle informazioni;
- Riduzione di tutti i tempi di inattività (per ricerca merce, attesa attività da fare, ecc.).

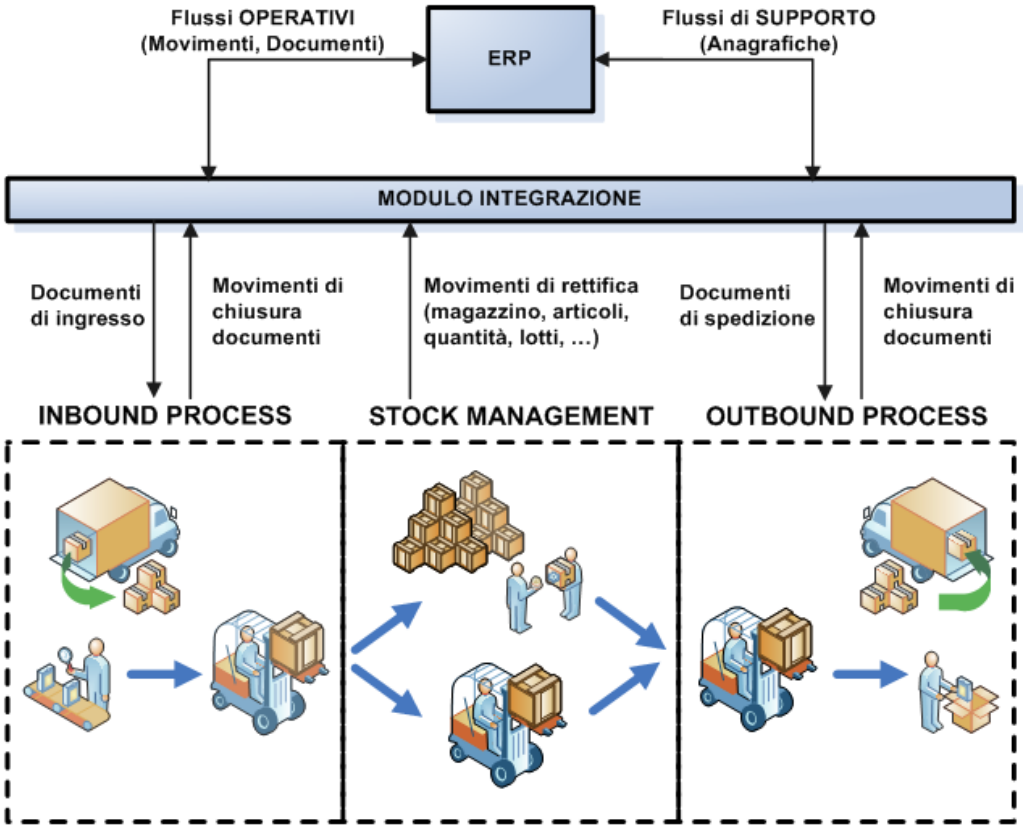

Figura 2.8: Movimentazione dei materiali di un magazzino

2.2.3 NICIM

Nicim [15] è la soluzione della Atomos per l'e-manufacturing di Programmazione della Domanda, Schedulazione ed Ottimizzazione, Avanzamento Produzione e Monitoraggio. L'applicazione è composta da più moduli che permettono di controllare in tempo reale lo stato di avanzamento delle lavorazioni, mentre la storicizzazione dei dati consente di elaborare statistiche di vario tipo, ad esempio su fermi, scarti, rendimenti. Tale sistema è composto da:

- *Nicim Schedule*: esegue la schedulazione a capacità finita delle attività produttive avendo come obiettivo la riduzione del lead time di trasformazione del prodotto, nel rispetto della data di consegna, dell'ottimizzazione dei tempi di attrezzaggio, della riduzione del valore di magazzino.

- *Nicim Monitor*: rileva in tempo reale i dati di produzione, manutenzione, movimentazione materiali, controllo qualità, aggiorna e controlla l'avanzamento lavori, esegue la consuntivazione dei tempi e delle quantità prodotte, permette il monitoraggio automatico degli impianti. Gli eventi critici sono immediatamente segnalati agli enti di competenza.

- *Nicim Sequence*: permette di effettuare forzature manuali sui piani di lavoro spostando le attività nel tempo o su altre risorse compatibili, modificando la sequenza degli ordini di lavoro, allungando o frazionando le lavorazioni stesse.

Le funzioni attualmente supportate all'interno della divisione Oil Control sono: Produzione (raccolta dati), Presenze, Schedulazione e supporta le integrazioni verso:

- *Apache*: anagrafica articoli, ordini di produzione, ciclo ordine, distinta ordine, giacenze, avanzamento fasi di conto lavoro;

- *WHMS*: importazione dichiarazioni quantità per alcune tipologie di ordine.

2.2.4 SAP R/3 Rainbow

SAP R/3 [17], sviluppato dalla SAP AG, è un software di classe Enterprise costituito da un nucleo di componenti di base (Basic Components) per l'integrazione ed il funzionamento dei moduli applicativi.
Il Basic Component, oltre ad un ambiente di programmazione workbench posto a disposizione dei clienti, eroga funzioni essenziali e trasversali, quali tra le altre: l'accredito al sistema (login), la navigabilità (menù), personalizzazione parametrica dei componenti (customizing), ampliamenti (enhancement), gestione della base di dati (un classico RDBMS "Open SQL" con astrazione dal database nativo). I moduli applicativo/gestionali sono integrati e coprono le tematiche proprie di una grande impresa: dall'amministrazione finanziaria al controllo di gestione, la tesoreria, dalla pianificazione alla gestione della logistica di merci e servizi, fino la vendita e la distribuzione sino all'integrazione dei cicli di produzione.
Le sigle storiche dei moduli applicativi sono:

- **BC**: Basic Component

- **FI**: Financial

- **CO**: Controlling

- **MM**: Material Management

- **SD**: Sales and Distribution

- **LE**: Logistic Execution
- **PP**: Production and Planning
- **PS**: Project System
- **PM**: Plant Maintenence
- **QM**: Quality Management
- **HR**: Human Resource

Tramite la dettagliata possibilità di personalizzazione di ogni componente SAP col cliente è consentita una notevole modularità di funzionamento, in grado di rispondere a esigenze complesse come quelle delle multinazionali, senza vincoli del settore/i d'affari o distribuzione geografica delle società, siti produttivi e di distribuzione. Il prodotto soddisfa le esigenze e normative legali/fiscali delle principali nazioni, occidentali e orientali.
La parte forse più impegnativa nell'introduzione e avviamento di SAP R/3 in azienda è la fase di studio e personalizzazione. L'uso di software applicativi integrati comporta una notevole e ampia revisione critica dei processi organizzativi. Ogni componente dovrà vedere e integrarsi con le altre.
L'introduzione di un software di tipo ERP richiede una progettazione congiunta di tecnologia ed organizzazione. Questi software sono prescrittivi per i processi, nel senso che impongono una trasformazione dei processi in funzione dello strumento informatico.
Il progetto ***Rainbow*** ha avuto inizio nel 2006 con la fase 1 già conclusa riguardante prevalentemente l'Area Contabile con i moduli di Finanza e Controllo (FI/CO), mentre la fase 2 è un progetto tuttora in corso riguardante l'integrazione degli altri moduli in tutte le altre aree aziendali (Vendite, Acquisti, Logistica, Produzione, Qualità) con l'obiettivo di avere un sistema SAP integrato.
Rainbow è l'acronimo di ***Rexroth Application INtegration for Business Optimization Worldwide*** ed è un'iniziativa globale che coinvolge tutte le Country Unit Bosch Rexroth, con l'obiettivo di implementare un sistema gestionale unico: SAP R/3, Versione 4.6c.
Rainbow si basa su modelli di processo comuni (template) da estendere a tutte le Country Unit per ottenere l'uniformità nella gestione dei processi a livello informatico, pur consentendo opportuni e necessari adattamenti alle realtà locali del mondo Bosch Rexroth. Questo approccio permette, attraverso il concetto di usabilità, di affrontare e risolvere in maniera efficiente problemi complessi che richiederebbero altrimenti soluzioni informatiche onerose ed articolate.

BRIT (Bosch Rexroth Italia) adotta già SAP R/3 ma attraverso Rainbow avrà l'opportunità di ottimizzare quanto supportato dall'attuale versione del sistema e di adeguarsi ai cambiamenti organizzativi e di mercato.

2.3 System Landscape: i legami tra i diversi sistemi

I sistemi produttivi e gestionali menzionati in precedenza sono collegati tra loro attraverso lo scambio di flussi informativi che possono scorrere in un senso o in entrambi. Tali informazioni sono strettamente legate al tipo di informazioni gestite in locale e al tipo di informazioni che il sistema ricevente ha bisogno. Quindi risulta utile comporre il System Landscape indicante lo scorrere di tutti i flussi informativi in modo tale da poter meglio comprendere come interagiscono tra loro i diversi sistemi e come una modifica strutturale impatta sull'intero sistema.

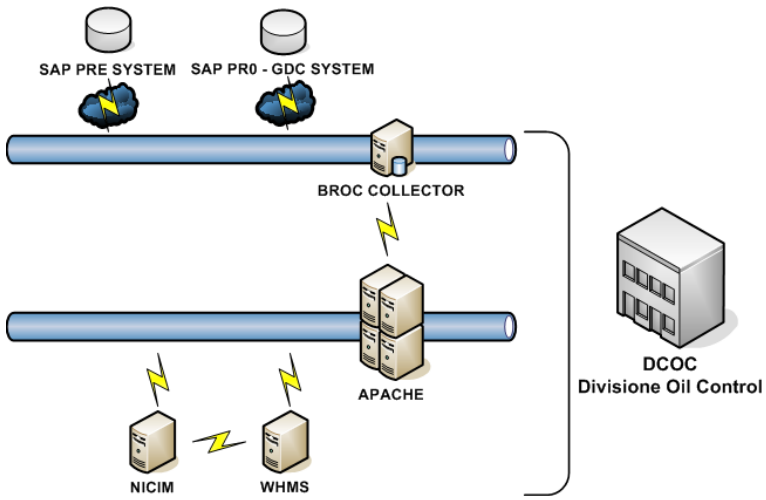

Figura 2.9: System Landscape della divisione Oil Control

Il termine System Landscape è utilizzato da SAP per indicare l'infrastruttura di sistema, ma nel nostro caso viene utilizzato per indicare l'infrastruttura software aziendale in un determinato ambito.
Dalla figura si può notare quali sono i sistemi soggetti a scambio di informazioni notando come alcuni sistemi, sia hardware che software, sono gestiti interamente all'interno della divisione, come Apache, Nicim e WHMS.
Ognuno dei sistemi citati, oltre a risiedere su server, gestisce un database per la memorizzazione dei dati, ma il passaggio delle informazioni tra i diversi sistemi avviene tramite scrittura di file temporanei su file system del sistema destinatario in determinate cartelle; poi sarà compito di qualche programma batch sviluppato appositamente per carpire la

presenza delle nuove informazioni, farle elaborarle dal sistema in questione e cancellare tali file temporanei.

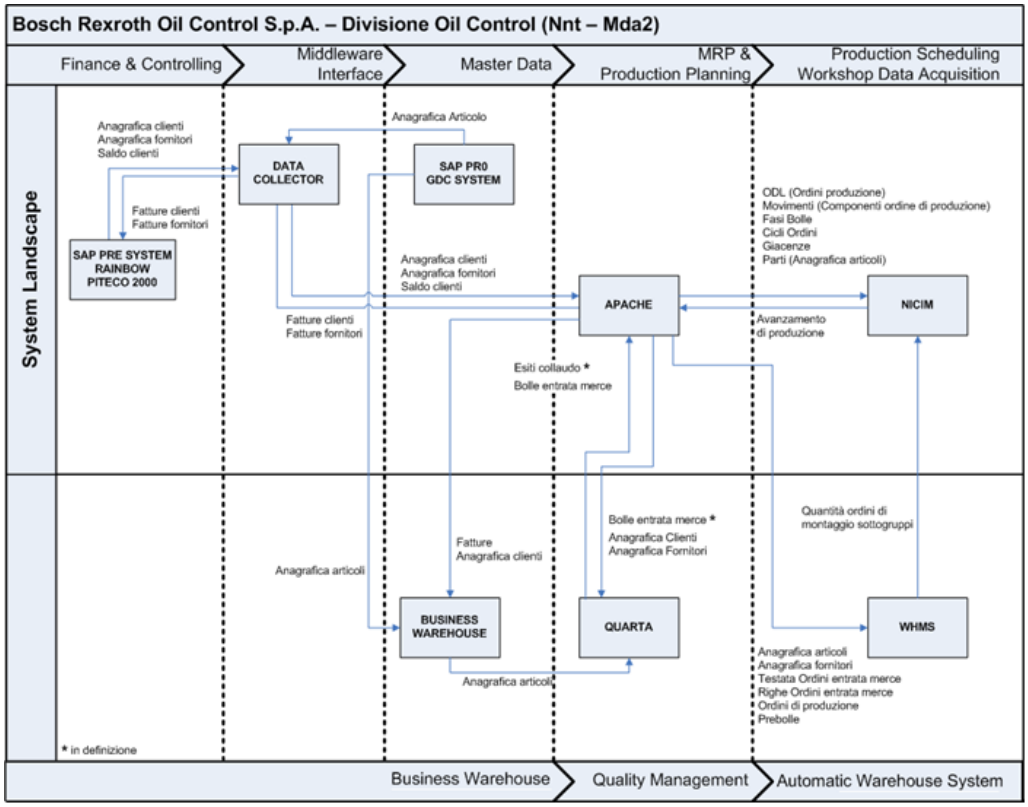

Figura 2.10: Dettaglio del System Landscape della divisione Oil Control

Nella figura 2.10 sono presenti altri sistemi interessati nello scambio di flusso dati ma non menzionati nel paragrafo precedente:

- **Piteco 2000**: è una soluzione applicativa aziendale per la gestione della tesoreria che si interfaccia con SAP R/3 Rainbow.
- **DCOC Data Collector**: è un data middleware interface comune per tutte le divisioni che permette di connettere SAP R/3 Rainbow verso tutte gli altri sistemi.
- **SAP PR0 GDC System** e **SAP PRE System**: sono dei database presenti in Germania contenenti l'anagrafica ed i codici di tutti gli articoli dei prodotti Bosch in ambito globale da permetterne una facile identificazione per acquisti all'interno dello stesso gruppo. Quindi qualsiasi inserimento o modifica effettuata dalla casa madre (Material Master Change/Creation) viene a conoscenza di tutte le divisioni.
- **Quarta** [18]: è una soluzione informatica per il conseguimento del TQM (Total Quality Management) e permette la gestione di strumenti di misura e resi/reclami dei clienti.

- **Business warehouse**: è una soluzione che sarà utilizzata per l'analisi sui dati.

I sistemi Apache, Nicim, WHMS e DCOC Data Collector sono tutti sistemi gestiti localmente, dove ogni divisione ha una sala server con diversi nodi cluster in base alle proprie esigenze. Per alcune applicazioni sono gestiti anche nodi ridondanti che subentrano appena si dovesse verificare un *crash system*. Ovviamente ogni sede implementa regole di accesso e misure di sicurezza opportune (ad esempio accesso tramite chiave/codice, riconoscimento tramite digital fingerprint/badge).
I sistemi SAP PRE System e SAP PR0-GDC System sono gestiti direttamente in Germania mentre il Business Warehouse (BW) e Quarta sono sistemi ancora in fase di sviluppo e risiederanno, molto probabilmente, nella sede di Nonantola.

Capitolo 3

L'interazione con il sistema gestionale

3.1 Il lavoro effettuato

Nel corso degli anni sono state sviluppate diverse applicazioni su Microsoft Access, molte richieste dall'Ufficio commerciale e dalla Logistica, ed utilizzati come programmi di analisi dei dati presenti sul database del sistema ERP aziendale (Apache). Queste applicazioni sono state realizzate attraverso tutti i punti chiave che offre Access, come *tabelle*, *query*, *maschere*, *macro* e *report*, ed integrando particolari funzioni in codice *VBA* (*Visual Basic for Applications*).
La scelta di utilizzare Access è stata effettuata perché permette la manipolazione dei dati senza avere approfondite conoscenze di SQL, e quindi adatto a tutti gli uffici, sia in fase di creazione applicazioni, ma più che altro per l'utilizzo degli stessi.
Molti di questi programmi sono stati realizzati per soddisfare le esigenze legate a quel momento lavorativo, ma che sono venute a mancare successivamente: quindi molti programmi risultano da aggiornare alle nuove esigenze o addirittura parzialmente/completamente obsoleti.
Il lavoro principale è stato quello di occuparsi della gestione degli ambienti applicativi e dei database, attraverso l'analisi, modifiche e cancellazioni di query e programmi sviluppati, con lo scopo di adeguarli alle recenti integrazioni delle divisioni Bosch Rexroth Oil Control.
Oltre al lavoro appena descritto oggetto del tirocinio, è stato ricoperto il ruolo di supporto al personale addetto ad effettuare report/query di estrazione dati richiesti dai diversi uffici (Amministrazione, Logistica, ecc.) e lavoro di supporto al personale di *Data Security* interno dell'azienda e delle diverse divisioni appartenenti al gruppo. Qui si è ricoperto il ruolo di analisi delle combinazioni critiche che ci sono tra le diverse applicazioni (ad esempio dati Apache da/verso SAP) e di supporto ai diversi uffici per la gestione delle combinazioni e transazioni critiche e la riduzione delle stesse.

Come è stato descritto nel capitolo precedente, i diversi programmi presenti in azienda comunicano tra loro attraverso la scambio di dati, ed è interessante avere una visione globale e dettagliata degli utenti che accedono ai diversi sistemi.
Quindi sono stati realizzati dei report appositi che permettono di analizzare gli accessi di un utente sul sistema SAP e su un altro sistema gestionale per verificarne eventuali criticità per tali sistemi, specialmente se ha permessi che possono creare transazioni o combinazioni potenzialmente dannose ai dati: questo serve per seguire la regola dei *(quattro occhi.*
Per SAP esiste già un modulo integrato adatto a questo scopo, mentre per gli altri sistemi (Apache, Gamma, AS400, AS400-Gipros) si utilizzano dei moduli appositi per svolgere ciò.
Dall'analisi effettuata si può notare come le transazioni critiche sono collegate anche al ruolo ricoperto dalla persona all'interno dell'azienda, più è alta la responsabilità, più è alta la possibilità di transazioni/combinazioni critiche. Quindi qui la scelta di far effettuare l'analisi ai rispettivi responsabili delle diverse aree interessate per poter meglio distribuire i compiti tra gli utenti.
Dopo aver effettuato questo, il personale ISY che gestisce i permessi di accesso ai dati (visualizza, inserisci, modifica, scorri, visualizza tutto, ecc.) abilita, tramite un'opportuna maschera, i diritti agli utenti.

3.2 Interagire con il sistema gestionale ERP Apache

Prima di poter svolgere il lavoro è stata effettuata un'attenta analisi dei sistemi su cui operare per poi poter meglio interagire e realizzare ciò che veniva richiesto. Dopo aver accertato come funzionano i diversi sistemi, specialmente il sistema gestionale Apache, le tabelle e come interagire su di esse, si è proceduto alla riscrittura delle query ed al porting dei programmi su RDBMS Oracle, con indubbi vantaggi che offre tale soluzione, sia in termini di prestazioni che in termini di portabilità.
In termini di prestazioni il guadagno che ne viene è abbastanza per poter giustificare il lavoro svolto in quanto la revisione dei programmi snellisce il lavoro di carico sul server dati per il prelievo (lettura dati effettuata solo dal server applicativo) e ne utilizza appieno la potenza di calcolo, il tutto favorendone la velocità di prelievo dati e di elaborazione, quindi una maggior velocità di risposta da parte del programma. Mentre la portabilità non è un'esigenza molto sentita in azienda poiché tutti i programmi sono stati realizzati in Access e la maggior parte dei sistemi operativi utilizzati dagli utenti sono Microsoft Windows, quindi non esiste una vera problematica legata alla portabilità delle applicazioni poiché sono tutte legate allo stesso parco software, ma ciò non toglie che in un prossimo futuro, se l'azienda vorrà adottare una linea diversa da quella attuale, il lavoro svolto

sulla portabilità non sarà del tutto vano.
Ovviamente non si può spostare tutto il lavoro sul database server senza considerare politiche di ottimizzazione degli accessi, in quanto tutti gli accessi introducono lavoro sul server e le invocazioni remote via middleware vanno utilizzate solo se strettamente necessario poiché introducono un overhead sulle prestazioni dovuto alla sola connessione.
Nel caso in cui le operazioni sui dati sono prevalentemente letture, si può ridurre notevolmente il numero di accessi al RDBMS in quanto si possono utilizzare soluzioni di caching dati:

- *locale*: risparmia la latenza di rete memorizzando i dati su una replica del DBMS in locale;
- in *RAM*: massima velocità di accesso ma necessaria una tecnologia specifica per la memorizzazione e l'interrogazione.

3.2.1 Le caratteristiche di Apache

L'applicazione Apache è stata realizzata a partire dagli anni '80 in Cobol ed è disponibile per le principali piattaforme tecnologiche (Windows, Linux, IBM mainframe e AS/400) e l'utilizzo con vari database relazionali (DB2, Oracle, Informix, Sybase, SQL Server).
Il software, nel corso degli anni, ha avuto costanti aggiornamenti con l'aggiunta di nuovi moduli, sia per implementare nuove funzionalità, sia soddisfare specifiche esigenze richieste dalle aziende clienti. Ora molti moduli non vengono sviluppati più in Cobol, ma direttamente in Java (con tutti i vantaggi che ha comportato questa scelta) ed interfacciando i moduli con il programma base.
Nel reparto ISY, specialmente coloro che sono responsabili delle applicazioni gestionali, utilizzano vari programmi di accesso diretto al database non utilizzando la classica interfaccia grafica messa a disposizione da Apache per poter effettuare operazioni sui dati come inserimenti/modifiche/cancellazioni e report: tipicamente queste operazioni vengono effettuati attraverso query in *SQL*, *viste*, *stored procedure* (procedure eseguite dal RDBMS su esplicita richiesta delle applicazioni o degli utenti) e *trigger* (procedure attivate automaticamente dal RDBMS al verificarsi di determinate condizioni). Per poter effettuare ciò si utilizzano dei tool appositi, ma ciò viene descritto in seguito.
Le operazioni che vanno a modificare direttamente i dati sul database escludendo l'utilizzo della GUI non sono frequenti, ma talvolta è necessario per poter riportare alcuni dati in uno stato corretto; comunque sono azioni molto pericolose se si effettuano senza una conoscenza approfondita dell'intera logica di "aggancio" dei dati da parte del gestionale.
La gestione di tutti i dati avviene mediante "livelli di revisione" e per stati di avanzamento. Ciò consente la storicizzazione delle modifiche e la loro preparazione in anticipo,

definendone la data di inizio validità. Ad esempio, l'aggiornamento dell'indirizzo di un cliente o le coordinate bancarie di un fornitore vengono gestiti dal sistema che ne effettua le modifiche sul database ma mantiene le informazioni precedenti, dando così sempre la possibilità di consultarli (storico dei dati).
L'interattività dell'impianto consente di sviluppare qualsiasi funzione attraverso una workstation collegata al server centrale. Se operativamente questo rappresenta un pregio, potrebbe significare accessi e manipolazioni di dati da persone non autorizzate. Per questo motivo, senza escludere il vantaggio procedurale, è presente un modulo in grado di definire gli utenti abilitati e le transazioni con le relative funzioni.

Lo sviluppo di particolari funzioni che si vogliono implementare in Apache vengono direttamente eseguite su commessa dalla MBM, attraverso un gruppo di specialisti in grado di offrire la necessaria assistenza per il corretto sviluppo di un progetto informatico:

- consulenza e analisi sulle problematiche aziendali;
- assistenza alle definizioni organizzative;
- progettazione di sistemi informativi integrati;
- selezione, dimensionamento degli impianti hardware;
- realizzazione delle soluzioni ideate;
- assistenza all'avvio operativo delle procedure.

Uno specifico modulo permette di definire gli utenti abilitati e le transazioni, con le relative funzioni che essi possono utilizzare. Ogni operatore, prima di iniziare una sessione di lavoro, deve identificarsi dichiarando il suo codice e la sua password. Da quel momento, fino alla richiesta esplicita di disconnessione, il terminale è abilitato ad eseguire tutte le attività previste dal profilo dell'utente che si è collegato.
Le abilitazioni sono date dal reparto ISY, più in dettaglio dai responsabili che, attraverso un'opportuna area amministrativa presente in Apache, abilitano o disabilitano i permessi sulle maschere legate all'utente. Tali permessi possono essere di *visualizzazione*, *inserimenti*, *modifica*, *cancellazioni*, azioni come *avvia*, *rilascia*, *scorri*, *stampa*, ecc.
In azienda, per politiche di *"Authorization Concept"*, le mansioni sono suddivise tra i diversi responsabili così da non poter avere una situazione con un responsabile avente diritti sia di creazione utenti e sia di assegnazione dei privilegi. Così ci sarà un responsabile abilitato per la creazione di nuovi utenti, della modifica e cancellazione degli stessi, mentre un altro per l'assegnazione di profili.
Il sistema conserva le informazioni del responsabile che ha creato l'utente e dell'ultimo responsabile che ne ha effettuato eventuali modifiche.

3.2.2 L'architettura

L'architettura hardware utilizzata in azienda può essere ricondotta al modello three-tier (tre strati). Con tale espressione indichiamo una particolare architettura software che prevede la suddivisione del sistema in tre diversi moduli dedicati, rispettivamente all'*interfaccia utente*, alla *logica funzionale* (*business logic*) e alla *gestione dei dati persistenti*. Tali moduli interagiscono tra loro secondo le linee generali del paradigma client-server utilizzando interfacce ben definite e nessun livello fa assunzioni sulla struttura o implementazione degli altri. In questo modo, ciascuno dei tre moduli può essere modificato o sostituito indipendentemente dagli altri.

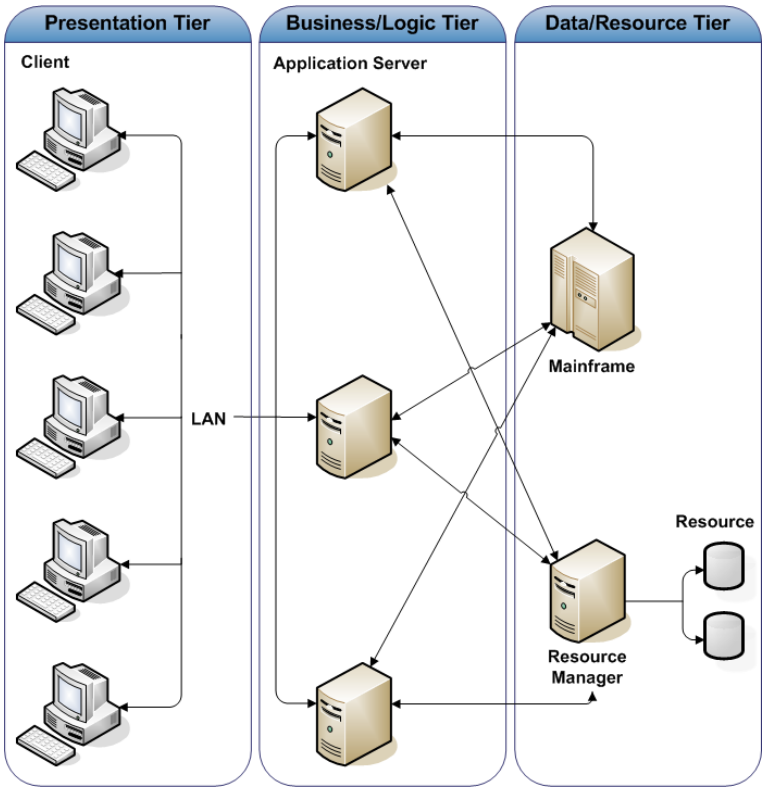

Figura 3.1: Schema semplificato di accesso ai dati

Analizziamo l'architettura più in dettaglio:

- **Presentation tier**: questo livello riguarda l'interfaccia utente per poter far interagire lo stesso con il sistema sottostante. La principale funzione dell'interfaccia è quella di poter tradurre i compiti (task) e mostrare i risultati in un formato comprensibile all'utente.

- **Business Logic tier**: con il termine Business Logic ci si riferisce a tutta quella logica applicativa che rende operativa un'applicazione. Il business logic racchiude in sé regole cosiddette di "business", piuttosto che regole ed elementi legati alla visualizzazione delle informazioni o alla memorizzazione dei dati.

- **Data tier**: in questo livello le informazioni sono memorizzate e recuperate dal database o dal file system dove poi vengono ritornate indietro verso il livello business logic per poterne eseguirne l'elaborazione, ed eventualmente più indietro ancora verso l'utente.

Una soluzione three-tier tipica prevede, per esempio, un PC dedicato all'interfaccia utente grafica, una workstation o un application server per la business logic e un database server o un mainframe per la gestione dei dati.
Questo schema generale è piuttosto diffuso e costituisce un'architettura di riferimento per molte tecnologie moderne. Può anche essere esteso ipotizzando che lo strato intermedio sia a sua volta "stratificato"; in questo caso si giungerebbe ad una architettura multi-tier (n-tier).
In azienda, Apache ricopre i primi due strati dell'architettura, presentation tier (attraverso l'interfaccia utente) e business logic tier, mentre il livello data tier è ricoperto dal RDBMS Oracle 10g.

3.2.3 L'interfaccia

L'interfaccia grafica (graphical user interface, abbreviato con GUI) utilizzata in azienda è ancora quella originaria ed obsoleta rispetto agli standard di mercato attuali, ma ancora funzionale per i suoi scopi.
Dall'interfaccia si può notare come si può accedere direttamente alla maschera interessata attraverso il menu oppure cliccando i bottoni presenti nella maschera.
Per ogni maschera è associata una o più tabelle sottostanti che ne contengono i dati. In linea generale, le maschere sono identificate con delle sigle (ad esempio la maschera "Elenco cicli di articoli modificati" presente in figura 3.2 è denominata GRDL) e ad ogni maschera corrisponde una o più tabelle. Nella maggior parte dei casi il nome della maschera coincide con il nome fisico della corrispondente tabella presente nel database, ma non sempre vale ciò.
Nel corso del 2008 è stata rilasciata la nuova veste grafica sviluppata in Java, Apache.J, con nuove funzionalità applicative e cambiamenti tecnologici:

- nuova veste grafica, migliorati i menu funzionali, l'utilizzo del sistema è ancor più "user friendly" ed intuitivo;

- introdotta la navigazione "ad albero" che permette una maggior velocità nel passaggio da una schermata all'altra.

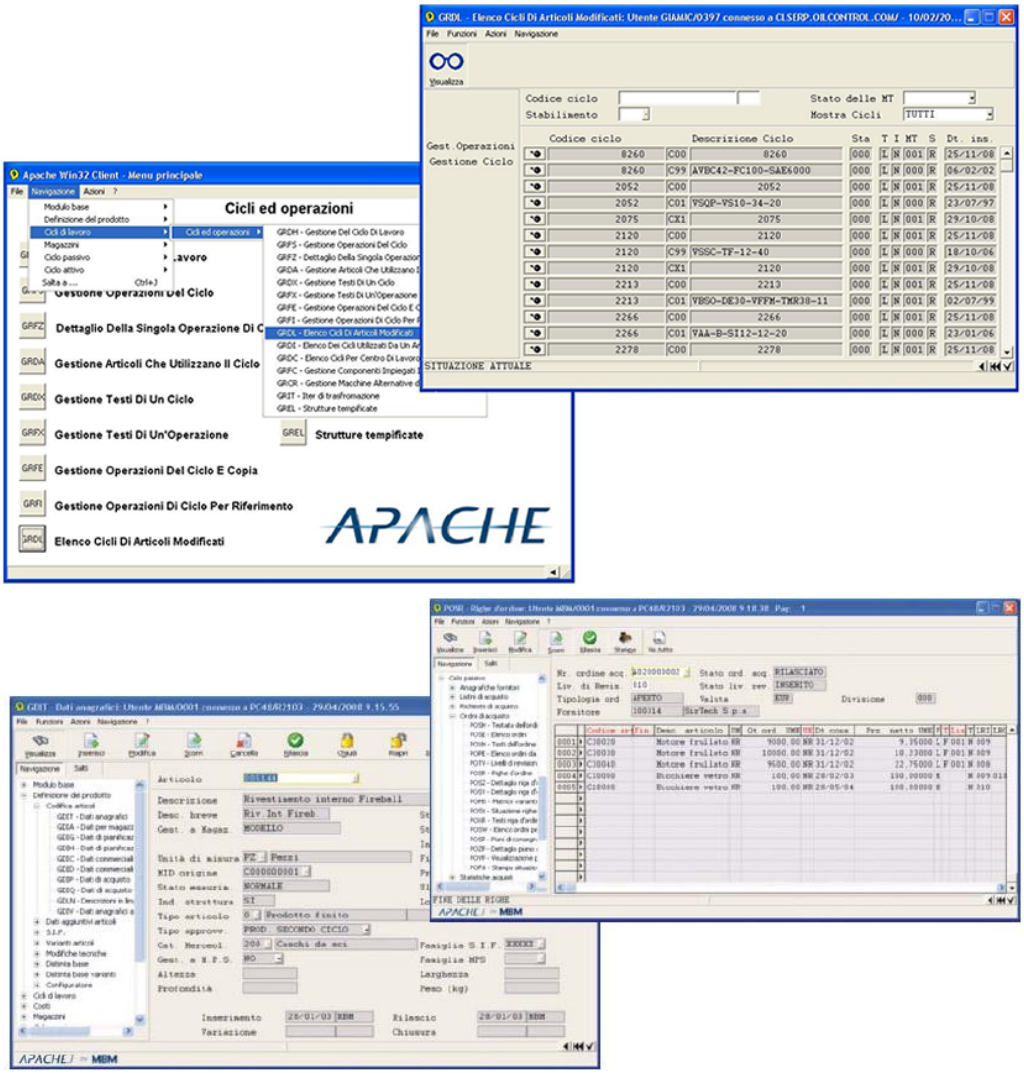

Figura 3.2: La vecchia e la nuova interfaccia grafica di Apache

3.2.4 Il database

I ***DBMS*** (***DataBase Management System***) sono sistemi software che permettono di gestire database, ovvero un insieme di dati logicamente correlati fra loro. Essi presentano seguenti caratteristiche:

- **Solidità**: offrono una struttura consolidata di gestione di archivi anche di enormi dimensioni;
- **Strumenti**: offrono un vasto insieme di strumenti di gestione dei dati;
- **Indipendenza fisica**: possibilità di variazione dello schema fisico dei dati senza la necessità di modificare le applicazioni che utilizzano quei dati;
- **Indipendenza logica**: possibilità di variare lo schema logico senza modificare il software applicativo;

- **Condivisione dei dati** fra più utenti e applicazioni;
- Utilizzo di **sistemi di protezione e autorizzazione** per l'accesso ai dati stessi.

In base alla loro struttura logica possiamo identificare diversi tipi di database da cui prendono il nome:

- **database gerarchici**: il *modello gerarchico* è basato su strutture ad albero nelle quali ogni dato che non sia a livello radice ha uno e un solo padre. È il modello che ha conosciuto il maggior utilizzo fino agli anni '80.
- **database reticolari**: il *modello reticolare* deriva da quello gerarchico rispetto al quale supera la rigidità della struttura ad albero nell'interdipendenza dei dati, ma la cui complessità ne ha impedito una larga diffusione.
- **database relazionali**: il *modello relazionale* organizza i dati in tabelle basandosi sulle relazioni fra essi. È il modello più diffuso.
- **database ad oggetti**: il *modello ad oggetti*, il più recente, estende i concetti del modello relazionale adattandoli alla programmazione ad oggetti.

Quindi i database più diffusi sono quelli relazionali che sono denominati *RDBMS* (*Relational DataBase Management System*), dove i più noti sono *Microsoft Access* e *SQL Server*, *Oracle*, *IBM DB2*, *MySQL* e *PostgreSQL*.
In azienda Apache si basa sul RDBMS Oracle 10g della Oracle Corporation, ma in generale Apache è un sistema indipendente dal database utilizzato e con caratteristiche multi-server.
Oracle utilizza il modello relazionale per organizzare le tabelle. Tale modello offre molta stabilità e flessibilità in modo da garantire la possibilità di sviluppare nuove ed impreviste applicazioni senza per questo dover riprogettare, o per lo meno ristrutturare, l'organizzazione dell'intero archivio dati (indipendenza logica), consentendo quindi un costante aggiornamento della sua struttura ed evitando la modifica di tutti i programmi applicativi che ne fanno uso.
Il database di Apache è composto solo da tabelle senza vincoli di integrità (ad esempio *vincoli not null*, *chiavi secondarie*, *unique*, *check*, *foreign key*) e le uniche chiavi presenti sono solo le chiavi primarie, dove peraltro sono stati costruiti sopra gli indici per migliorarne i tempi di accesso/ricerca sui dati. Quindi tutte le operazioni di relazione logica, d'integrità e di possibili valori che può assumere un attributo sono completamente gestite dall'interfaccia grafica di Apache, il quale impedisce le operazioni non consentite. Infatti tutte le operazioni non permesse, ad esempio l'inserimento di dati errati o solo parziali oppure inserimenti/modifiche/cancellazioni effettuate da utenti senza le relative autorizzazioni, sono impedite dalla stessa maschera dell'applicazione.

Quindi l'interfaccia gestisce tutte le operazioni possibili escludendo così eventuali situazioni di inconsistenza dei dati. Ma ciò non toglie che si possono essere effettuate modifiche direttamente sul database attraverso vari tool di sviluppo per inserire comandi SQL, con la possibilità di poter effettuare query di aggiornamento tramite programmi con interfaccia grafica oppure direttamente in SQL, facendo attenzione alle azioni di popolazione del database per evitare situazioni di dati non più consistenti tra loro.
Le tabelle in Apache sono numerose e sono suddivise logicamente in diversi moduli e sottomoduli. I figura 3.3 è rappresentata la parte più significativa.

Modulo	Sottomodulo	Modulo	Sottomodulo
Ciclo passivo	Ordini di produzione	**Cicli di lavoro**	Cicli ed operazioni
	Entrata materiali	**Modulo base**	Parametri applicativi
	Anagrafiche fornitori	**Gestione della produzione**	Ordini di produzione
	Listini di acquisto	**Definizione del prodotto**	Codifica articoli
	Controllo fatture passive	**Magazzini**	Periodi gestionali
	Richieste di acquisto	**Gestione della produzione**	Gestione KANBAN
Costi	Costi standard	**Ordini pianificati**	Pianificazione materiali
	Costi consuntivi	**Modulo base**	Parametri applicativi
Ciclo attivo	Anagrafiche clienti	**Definizione del prodotto**	Codifica articoli
	Spedizioni	**Controllo qualità**	Strumenti
	Gestione fatture	**M.P.S.**	-
	Packing list	**NICIM**	Articoli
	Listini di vendita		Ordini di produzione
	Ordini di vendita		Giacenze
	Resi clienti		
Definizione del prodotto	Codifica articoli		

Figura 3.3: Moduli e sottomoduli presenti in Apache

Ma il database comprende anche altre tabelle non elencate in precedenza, come quelle realizzate in azienda per poter sopperire a determinate lacune funzionali del sistema, le tabelle di base per il sistema MPS e altre tabelle realizzate per lo scambio di dati con gli altri sistemi, come ad esempio con il sistema Nicim per le informazioni su Ordini di produzione, Articoli e Giacenze.
Il sistema MPS contiene le tabelle che permettono di memorizzare i dati come:

- Costi aggiuntivi per commessa
- Testata piano di produzione
- Impegni legati ai piani
- Categorie della commessa
- Righe del piano di produzione
- Progetti
- Testata piani produzione MPS

- Piani previsionali mensili
- Proposte
- Piani di produzione MPS
- Stabilimenti in elaborazione MPS

Infatti il modulo MPS di Apache, alla fine dell'analisi, propone diversi piani di produzione dove sono presi in visione e scelto in modo opportuno il piano più consono alle esigenze del momento.

3.3 L'accesso ai dati del database

Lavorare con i dati in un RDBMS è abbastanza diverso dal lavorare con i dati in un elaboratore di testo o di un foglio di calcolo. In un documento di testo si possono includere i dati sotto forma di tabella ed eseguire un insieme limitato di funzioni sui dati stessi. In un foglio di calcolo alcune celle contengono funzioni che determinano il risultato che si desidera, mentre in altre celle ci sono dati che foniscono le informazioni sorgente per le funzioni, ma sono destinati ad uno particolare scopo ed è complicato utilizzare gli stessi dati per risolvere un problema differente.
Essendo un linguaggio dichiarativo, l'SQL non richiede la stesura di sequenze di operazioni ma piuttosto di specificare le proprietà logiche delle informazioni ricercate. Esso si divide in tre sottoinsiemi:

- ***Data Definition Language*** (***DDL***): serve a creare, modificare o eliminare gli oggetti in un database. Sono i comandi DDL a definire la struttura del database, e quindi dei dati ivi contenuti, ma non fornisce gli strumenti per modificare i dati stessi: per tale scopo si usa il DML. L'utente deve avere i permessi necessari per agire sulla struttura del database e questi permessi vengono assegnati tramite il DCL.
- ***Data Manipulation Language*** (***DML***): fornisce i comandi per inserire, modificare, eliminare o leggere i dati all'interno delle tabelle di un database. La struttura di questi dati deve già essere stata definita tramite il DDL. Inoltre, il permesso di accedere a tali dati deve essere assegnato all'utente tramite il DCL.
- ***Data Control Language*** (***DCL***): serve a fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi DML e DDL, oltre agli stessi comandi DCL (che gli servono per poter a sua volta modificare i permessi su alcuni oggetti).

- ***Device Media Control Language*** (***DMCL***): è il linguaggio per il controllo dei supporti di memorizzazione e consente di far corrispondere il modello logico definito con DDL al supporto fisico su cui scrivere i dati.

- ***Query Language*** (***QL***): è il linguaggio di interrogazione utilizzato per interrogare il database al fine di individuare i dati che corrispondono ai parametri di ricerca dell'utente.

In teoria, qualsiasi prodotto basato sul linguaggio SQL dovrebbe essere in grado di comunicare con altri prodotti basati sullo stesso linguaggio, vale a dire che dovrebbe essere possibile creare un'applicazione in grado di funzionare con i dati contenuti in più sistemi di gestione di database relazionali utilizzando lo stesso linguaggio di database.
Sebbene gli standard per SQL esistano, la maggior parte delle società fornitrici di software hanno implementato variazioni o estensioni di tale linguaggio, per poter gestire funzioni specifiche dei propri prodotti. Inoltre, numerosi prodotti si sono evoluti prima che gli standard fossero completamente definiti, pertanto le società che li producevano hanno inventato le proprie sintassi SQL che differiscono dagli standard ufficiali. Un'istruzione SQL, progettata per essere eseguita su una specifica piattaforma, può richiedere modifiche prima di essere eseguita da altri database che supportano il linguaggio SQL.
Ecco perché sono nate soluzioni per poter far comunicare le diverse piattaforme, come i driver ODBC o i provider OLE-DB. Con l'evoluzione tecnologica sono nate altre soluzioni, alcune che si basano su queste ide, altre che implementano differenti soluzioni.
Come già introdotto ad inizio capitolo, le diverse applicazioni realizzate su Access sono state sviluppate nel corso degli anni dietro richiesta dell'Ufficio commerciale e della Logistica, ed utilizzati come programmi di analisi dei dati presenti sul database del sistema ERP aziendale (Apache).
Gli uffici interessati accedono ai dati solo tramite il programma Access, mentre nel reparto ISY si può accedere anche tramite un tool di SQL avanzato poiché si ritiene che tale personale abbia il knowledge necessario per poter interagire tramite comandi SQL (ovviamente riguarda i responsabili che ricoprono il ruolo di gestione dei sistemi gestionali).
Alcuni modi per poter accedere al database Oracle utilizzato in azienda sono attraverso:

- ***Microsoft Access*** per gestire graficamente i dati, tabelle, relazioni e query;

- un tool di SQL avanzato come ***SQL Navigator*** della Quest Software.

I software appena citati non sono gli unici sistemi esistenti per poter effettuare il lavoro richiesto, ma sono legati all'attività quotidiana, ed in particolare il primo è abbastanza utilizzato nei diversi uffici.

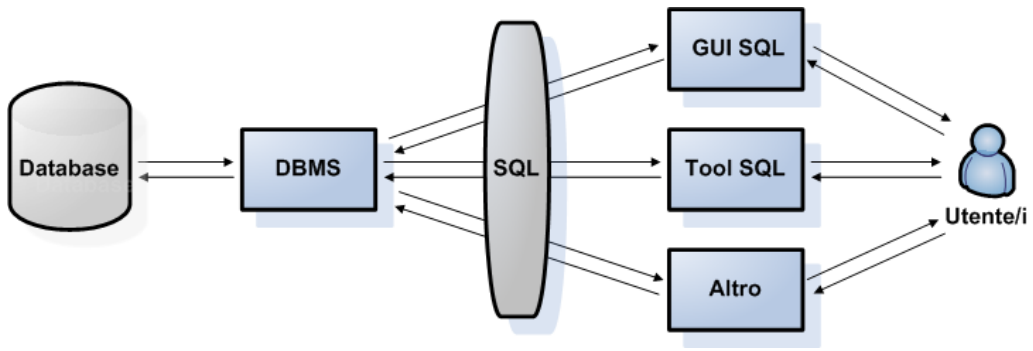

Figura 3.4: Schema semplificato di accesso al database da parte di un utente

3.3.1 Microsoft Access

Microsoft Access [19] è un RDBMS realizzato da Microsoft, incluso nel pacchetto Microsoft Office ed unisce il motore relazionale Microsoft Jet Database Engine con un'interfaccia grafica. Può utilizzare dati immagazzinati in formato Access/Jet, SQL Server, Oracle o qualsiasi database in formato compatibile ODBC. È possibile altresì utilizzarlo come Front-Editor verso Microsoft SQL Server, attraverso ADO/OLE-DB.
Può essere utilizzato a differenti livelli, sia da programmatori alle prime armi, sia da sviluppatori software esperti per realizzare applicativi anche molto complessi.
Access offre un sistema molto ricercato per lo sviluppo di applicazioni per Microsoft Windows in quanto permette di creare applicazioni rapidamente a partire da qualsiasi origine dati. Infatti è possibile costruire semplici applicazioni definendo le maschere e i report in base ai dati collegandoli con poche istruzioni di Microsoft VBA (Visual Basic for Applications).
Per le piccole imprese (e i consulenti che creano applicazioni per esse) le funzionalità di sviluppo desktop di Access sono sufficienti, considerati i loro tipici volumi di dati da archiviare e gestire. Insieme ad un database lato server, Access è ideale per molte società di medie dimensioni che devono creare nuove applicazioni Windows in modo rapido ed economico. Per le grandi imprese che investono in applicazioni di database relazionali su mainframe, nonché nella distribuzione di applicazioni di database desktop per PC, Access fornisce gli strumenti per collegare facilmente i dati dei PC e dei mainframe in una singola applicazione Windows.
Access dispone inoltre di una facile ma potente funzionalità grafica per la definizione delle query che si può utilizzare per specificare i dati di cui si ha bisogno per risolvere un problema. Con alcuni strumenti di puntamento, la tecnica di trascinamento e alcune sequenze di tasti, è possibile creare una query complessa in pochi secondi.

3.3.2 Power Quest SQL Navigator

Per lo sviluppo e l'amministrazione dei database si è utilizzato ***SQL Navigator*** [20], software commerciale realizzato dalla Power Quest, poiché è uno strumento che semplifica il flusso di lavoro aggiungendo un'interfaccia grafica all'ambiente di sviluppo SQL. Permette di incrementare la produttività degli utenti e migliorare la qualità del codice applicativo attraverso potenti funzionalità per la gestione degli oggetti dei database, lo sviluppo e il debug di codice *PL/SQL* (*Procedural Language/Structured Query Language*) e la creazione, l'esecuzione e l'ottimizzazione delle query SQL.
Il PL/SQL è un linguaggio di programmazione per database della Oracle Corporation; ha le caratteristiche di essere procedurale, server-based ed estensione dell'SQL. Linguaggi simili al PL/SQL sono inclusi in altri RDBSM SQL, come *Sybase* e il suo derivato *Microsoft SQL Server* hanno *Transact-SQL*, *PostgreSQL* ha *PL/pgSQL* (che cerca di emulare PL/SQL), *DB2* include *SQL Procedural Language* (*SQL PL*) e *MySQL* ha una versione di SQL molto simile a PL/SQL.
Tra i diversi componenti inclusi nell'applicazione, ricordiamo:

- **CodeXpert**: permette di evidenziare graficamente il codice dividendo le varie funzioni e di abbreviare il tempo speso per la scrittura del codice diminuendo il tasso di errori e i sforzi spesi per la manutenzione del codice;
- **E/R Diagram**: permette di disegnare graficamente lo schema E/R e vederne le dipendenze tra le varie tabelle;
- **Code Roadmap**: permette di visualizzare graficamente le complesse interdipendenze del codice PL/SQL con il database.

3.4 Il modello E/R del database

MBM ha rilasciato diverse documentazioni sul database di Apache riguardo l'utilizzo, le funzioni e le tabelle, specificando gli attributi e le funzionalità di ogni campo con la relativa descrizione. Quello che però manca, a primo impatto, è proprio uno schema principale che mostri come sono collegate tra loro le tabelle per poter meglio comprendere la logica di fondo utilizzata dal sistema. Ma bisogna considerare che la MBM fornisce l'interfaccia GUI proprio per far interagire con il sistema non conoscendone la sua struttura funzionale. Per individuare un filo logico dei collegamenti tra i campi delle diverse tabelle si può effettuare un'analisi sul nome delle tabelle e dei campi, che molte volte sono abbastanza simili, mentre altre volte non proprio. Comunque tutte le tabelle hanno nel nome in comune la Q iniziale ed una coppia di zero finali.
Ad esempio, sulle spedizioni del ciclo attivo si hanno due tabelle, una relativa alla testata

della prebolla/bolla, l'altra relativa alle righe di una determinata prebolla/bolla.

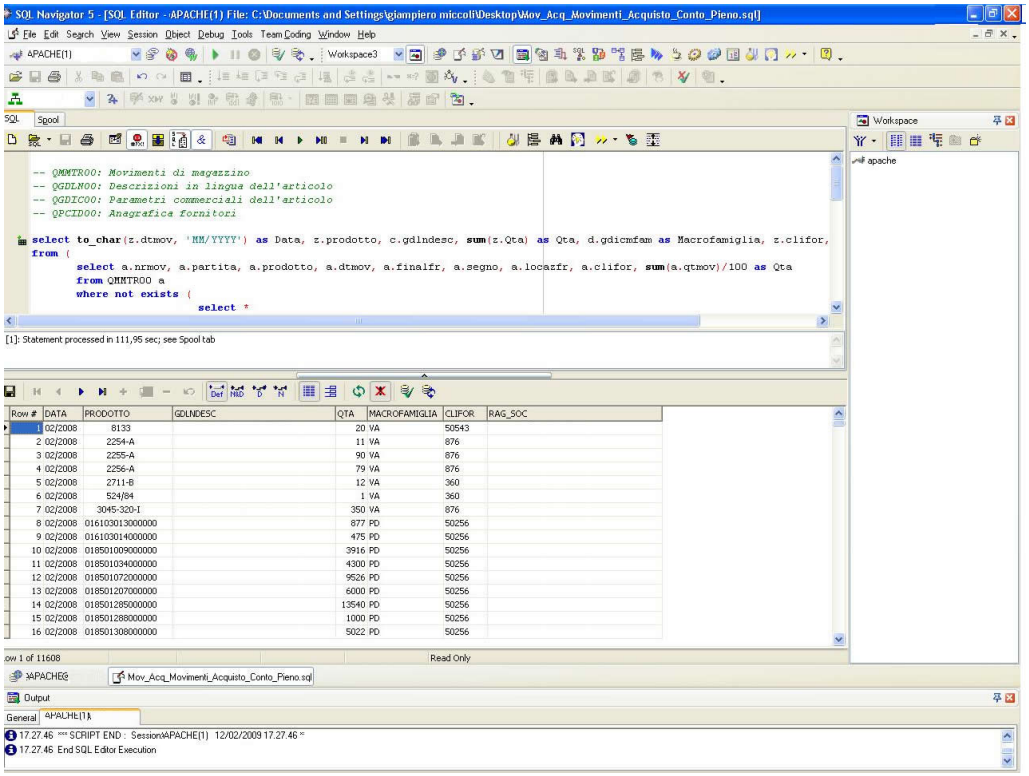

Figura 3.5: Esempio di query SQL realizzata attraverso SQL Navigator

Modulo	Sottomodulo	Tabella	Descrizione	Descrizione nome tabella
Ciclo attivo	Spedizioni	QCBLH00	Testata prebolla/bolla	B → Bolla LH → Testata (header)
Ciclo attivo	Spedizioni	QCBLN00	Righe prebolla/bolla	B → Bolla LN → Riga (line)

Modulo	Sottomodulo	Tabella	Descrizione	Campo	Descrizione nome campo
Ciclo attivo	Anagrafiche clienti	QCAID00	Anagrafica cliente	CAIDCLIE	CAID → Nome tabella CLIE → Codice del cliente
Ciclo attivo	Anagrafiche clienti	QCAIN00	Indirizzi del cliente	CAINCLIE	CAID → Nome tabella CLIE → Codice del cliente

Figura 3.6: Alcuni esempi di tabelle e campi del database Apache

Oppure si può notare come i campi sono logicamente collegati tra loro, anche se hanno nomi fisici diversi. In questo caso vediamo come due tabelle diverse, QCAID00 e QCAIN00, hanno in comune il campo del codice cliente. Come si è visto, i campi sono logicamente collegati tra loro, ma in mancanza di un modello concettuale, cioè di uno *schema E/R* (*Entity Relationship Diagram*), è abbastanza difficile capire come operano tra loro le diverse entità e come farle interagire in modo proficuo evitando di commettere errori.

Capitolo 4

Il Reverse engineering delle query SQL

4.1 Cosa si intende con il termine "Reverse engineering"

Col passare degli anni, modifica dopo modifica, i sistemi software invecchiano e la loro struttura tende inevitabilmente a degenerare. Spesso i manager e gli ingegneri del software sono costretti a dover gestire un sistema molto grande, vitale per l'organizzazione, costato molto e al tempo stesso difficile da comprendere e mantenere. A peggiorare le cose è il fatto che solitamente la documentazione non esiste e se esiste non è aggiornata. Questo punto di arrivo, che ha come conseguenza quella di rendere molto costosi e difficili tutti gli interventi di manutenzione, costringe spesso ad una scelta difficile: riscrivere il sistema da zero oppure ristrutturarlo completamente. Le tecniche di reverse engineering possono essere utili nel caso in cui viene scelta questa seconda soluzione.
Il ***Reverse engineering***, in italiano ingegneria inversa, è un processo di analisi di un oggetto o dispositivo (un componente elettronico, un aeroplano, un motore, un software, ecc.) che ha come obiettivo unicamente la comprensione. Solitamente il passo successivo del reverse engineering è quello di costruire un nuovo oggetto/dispositivo avente caratteristiche e funzionalità simili a quello analizzato.
Il reverse engineering è stato usato spesso dalle forze armate al fine di copiare la tecnologia bellica dei nemici, ma in ambito informatico lo scopo del reverse engineering è abbastanza diverso rispetto a quello militare. Solitamente l'ingegneria inversa è utilizzata dai progettisti per aumentare il grado di conoscenza di un sistema software quando questo deve essere sottoposto ad una operazione di modifica. Nei casi reali non è raro che i progettisti siano costretti ad eseguire operazioni di manutenzione del codice senza avere una conoscenza approfondita dello stesso. Questa eventualità si ha tutte le volte che sul software agiscono progettisti o ingegneri che non hanno partecipato alle fasi di sviluppo (caso assai frequente visto che alcuni sistemi hanno oramai più di 30 anni) oppure quando occorre modificare il proprio codice dopo che è passato un po' di tempo.

Le tecniche e i tool di reverse engineering forniscono i mezzi per generare un'adeguata documentazione del codice: in particolare sono in grado di produrre documenti mai esistiti o recuperare quelli che si sono persi con il passare degli anni. Il risultato di questo processo è un insieme di rappresentazioni alternative del sistema, spesso grafiche, che aiutano il progettista nella fase di manutenzione ed evoluzione del codice.

4.1.1 I sistemi Legacy

Per molti anni l'ingegneria del software si è concentrata soprattutto sullo sviluppo di nuove applicazioni trascurando in parte la fase di manutenzione. Ultimamente però, con il crescere dell'importanza del software e soprattutto con la proliferazione dei sistemi legacy, questa fase è stata notevolmente rivalutata sia nel mondo della ricerca che in quello dell'industria.
Uno dei momenti più critici della manutenzione è rappresentato dalla comprensione del codice, fase nella quale il programmatore cerca di capire la struttura interna del software (o parte del software) e il suo funzionamento prima di effettuare le dovute modifiche.
È proprio nella fase di comprensione che risultano particolarmente utili le tecniche di reverse engineering. Queste tecniche forniscono i mezzi per recuperare le informazioni perse con il passare degli anni o che non sono mai esistite. L'obiettivo è quello di costruire progressivamente dei modelli mentali del sistema in modo tale da migliorare la comprensione dello stesso. Mentre questa operazione non è complessa per piccoli sistemi software, dove lettura ed ispezioni del codice sono spesso sufficienti, diventa estremamente problematica nel caso di sistemi legacy a causa della loro grandezza e complessità.
Il sistema legacy è un sistema informatico esistente o un'applicazione che continua ad essere usata poiché l'utente (tipicamente un'organizzazione) non vuole o non può rimpiazzarla. Con questo termine si indicano quindi i sistemi IT che utilizzano tecnologie meno recenti e per questo motivo sono molto difficili da interfacciare con i sistemi più recenti. Per tale interfacciamento si può ricorrere a sistemi middleware ma il costoso utilizzo di questi ultimi spesso decreta la sostituzione del legacy con tecnologie odierne.
Un sistema legacy generalmente presenta le seguenti caratteristiche:

- è vitale per un'organizzazione ed è pesantemente utilizzato; per esempio deve essere operativo 24 ore su 24 e 7 giorni su 7.
- su di esso l'organizzazione ha investito molto, sia in termini economici che di tempo.
- non può essere dismesso facilmente in quanto è l'unico repository di conoscenza. Le procedure dell'azienda, le cosiddette business rules, non sono registrate in nessun altro posto e sono contenute solo nel codice.

- solitamente è un codice enorme costituito da milioni di linee e scritto in un linguaggio di vecchia generazione (per esempio: Cobol, Assembler, PL/1, RPG, ecc.) ed è progettato secondo vecchie concezioni (per esempio: programmazione non strutturata, salti incondizionati, ecc.). Inoltre spesso è eseguito su piattaforme obsolete.

- solitamente utilizza un database obsoleto sempre che non faccia uso di un file system (file ad accesso sequenziale, accesso diretto, ecc.).

Le ragioni che inducono a mantenere sistemi legacy sono soprattutto dovute ai costi sostenuti per la loro implementazione e ai costi da sostenere per la migrazione a nuovi sistemi. Molte persone usano questo termine per riferirsi a sistemi "antiquati".
È importante sottolineare che le tecniche di reverse engineering, originariamente pensate per i sistemi legacy, si sono rivelate estremamente utili anche in supporto alla comprensione di sistemi software scritti secondo approcci moderni (per esempio, codice orientato agli oggetti o applicazioni Web).
Dato un sistema legacy ci sono diversi approcci per cercare di ricostruire le funzionalità del software e la loro interazione con i dati:

- lettura della documentazione esistente e del codice: è difficile usare questo approccio quando la documentazione è datata, incompleta o inesistente. Inoltre la lettura del codice diventa improponibile quando le linee di codice sono nell'ordine del milione (si dice che questa tecnica non scala con il crescere del sistema).

- intervista agli utenti e agli sviluppatori: questo è un buon metodo ma è applicabile solo raramente. È infatti difficile trovare gli sviluppatori originali che hanno partecipato allo sviluppo del codice.

- utilizzo di tools e delle tecniche proprie del reverse engineering per generare rappresentazioni ad alto livello del codice.

In letteratura esistono due definizioni diverse di reverse engineering, una "forte" e una cosiddetta "debole":

- definizione forte: è un processo che a partire dal codice legacy permette di estrarre le specifiche formali del sistema. Le informazioni di design del sistema vengono derivate dal codice come passo intermedio (figura 4.1) e le specifiche formali estratte possono essere usate per creare una nuova implementazione del sistema. In questa definizione ci sono tre assunzioni implicite:

  - il processo è automatico;
  - le specifiche sono ad un buon livello di astrazione in modo tale che il sistema possa essere re-implementato in un altro linguaggio o secondo concezioni diverse;

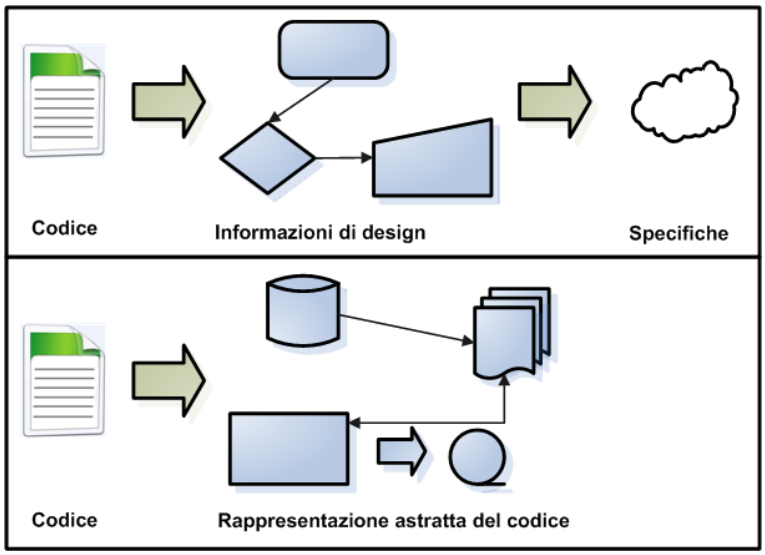

Figura 4.1: Definizione forte (in alto) e debole (in basso) di reverse engineering

  - il tempo e lo sforzo per derivare le specifiche è inferiore rispetto a quello richiesto per costruire il sistema da zero.

- definizione debole: è un processo automatico o semi-automatico (ovvero assistito dall'utente) che a partire dal codice legacy permette di derivare una base di conoscenza del sistema. La base di conoscenza è costituita da rappresentazioni alternative del codice legacy, spesso ad un livello più astratto e grafico, che mettono in risalto alcune proprietà e caratteristiche del sistema stesso.

Questa seconda definizione è molto meno ambiziosa rispetto alla precedente. Nella definizione debole non viene richiesto come risultato un insieme di specifiche formali del sistema ma solo una rappresentazione astratta del codice. Altra differenza fondamentale è che il processo non deve essere completamente automatico bensì assistito dall'utente.

4.1.2 Forward engineering, restructuring e re-engineering

Sono termini strettamente connessi al reverse engineering:

- ***Forward engineering*** è il tradizionale processo di sviluppo del software che inizia con l'analisi dei requisiti e termina con l'implementazione del sistema. In un certo senso il reverse engineering può essere considerato l'operazione inversa.

- ***Restructuring*** è una trasformazione di un programma da una rappresentazione ad un'altra che preserva il comportamento esterno del sistema, ovvero le funzionalità. Un esempio di restructuring a livello di codice è la conversione di un programma che fa uso di salti incondizionati (codice a "spaghetti") in uno strutturato senza

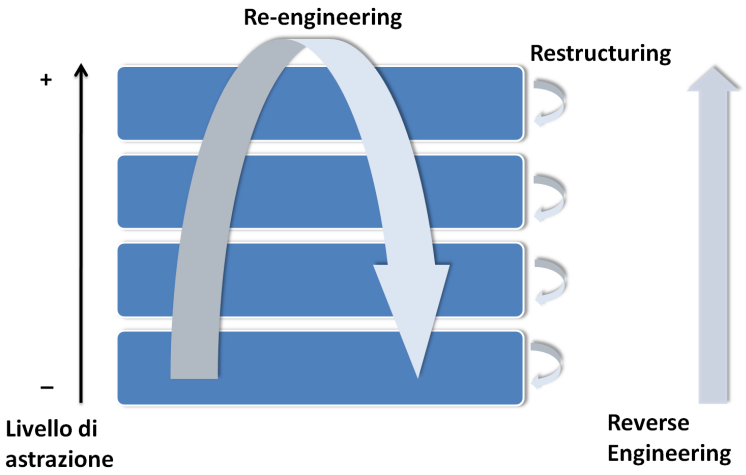

Figura 4.2: Relazione tra i diversi termini

i comandi di "goto". Un esempio di restructuring a livello di design è invece la sostituzione di una struttura dati con un'altra (per esempio: sostituire il file system con un DBMS).

- ***Re-engineering*** è un operazione composta da due attività (si veda la figura 4.2): analisi/comprensione del sistema (reverse engineering) e ricostruzione dello stesso in una nuova forma (forward engineering). Il processo di re-engineering può includere, a differenza del restructuring, delle modifiche rispetto ai requisiti del sistema originale. Un'operazione di re-engineering può essere eseguita per diverse ragioni. La migrazione di sistemi legacy, il riuso o la sicurezza del codice, la migrazione verso linguaggi più evoluti (per esempio: da C a Java) oppure la migrazione di sistemi sul Web.

4.2 La manutenzione dei Sistemi Software

Il ciclo di vita di un sistema software inizia con l'analisi del dominio e dei requisiti, continua con la fase di sviluppo (progettazione e codifica) e termina con la dismissione (momento in cui viene stabilito che il sistema software non è più utile per l'organizzazione).
Dopo il rilascio, cioè quando il sistema diventa fruibile da parte degli utenti, il software entra in una fase molto delicata che solitamente dura molto, la manutenzione.
Per manutenzione si intende il complesso di operazioni necessarie a:

- conservare il sistema software funzionante ed efficiente con il passare degli anni;
- mantenere le funzionalità del sistema continuamente aggiornate.

La manutenzione è una fase molto critica e ancora oggi, nonostante il miglioramento delle tecniche di sviluppo e dei processi di produzione, rappresenta per tutte le aziende una spesa molto cospicua, addirittura si può stimare in un valore compreso tra il 50% e il 75% della spesa totale di produzione del software. Mantenere ed evolvere un sistema software esistente è difficile perché bisogna tenere presenti diversi aspetti: il veloce "turnover" degli sviluppatori, la continua crescita dei sistemi software in termini di grandezza e complessità e l'evoluzione della tecnologia. I costi sono molto alti perché la manutenzione è composta di una serie di attività estremamente complesse:

- **comprensione del codice**: consiste in un'analisi approfondita del software o solo di una parte. Prima di effettuare una modifica occorre capire come funziona il codice e dove apportarla. Questa è sicuramente l'attività più complessa tra tutte ed è estremamente complicato comprendere il proprio codice e peggio ancora capire il software scritto da altri senza l'adeguato supporto di documenti e commenti del codice utilizzato;
- **analisi di impatto**: serve a capire quali funzionalità vanno aggiornate in modo da renderle consistenti con la modifica da implementare;
- **modifica del codice**: consiste nella fase di codifica del cambiamento;
- **validazione**: consiste nel verificare che la modifica non abbia introdotto nuovi errori.

La manutenzione, oltre ai costi diretti, implica anche un insieme di costi indiretti. Il più evidente in ambito industriale è lo sbilanciamento nell'allocazione delle risorse verso la fase di manutenzione. Quello che accade in realtà è che i programmatori sono impegnati, quasi sempre, a mantenere codice già rilasciato dedicando poco tempo allo sviluppo di nuovi progetti. Il risultato è che le risorse effettive impegnate per lo sviluppo di nuovi progetti sono spesso insufficienti.
Un altro aspetto che rende questa fase molto delicata è il fatto che una manutenzione non strutturata e caotica porta al rapido degrado del sistema, facendo diventare le successive modifiche sempre più complesse. Un sistema software raggiunge lo stato di legacy quando subisce nel tempo diverse modifiche e la fase di manutenzione diventa sempre più complessa e costosa.

4.3 Principi di Reverse engineering applicati su Database

[31] In questa tesi il concetto di reverse engineering è stato applicato in un campo diverso da quello trattato nel paragrafo precedente, ovvero non in ambito di codice software ma

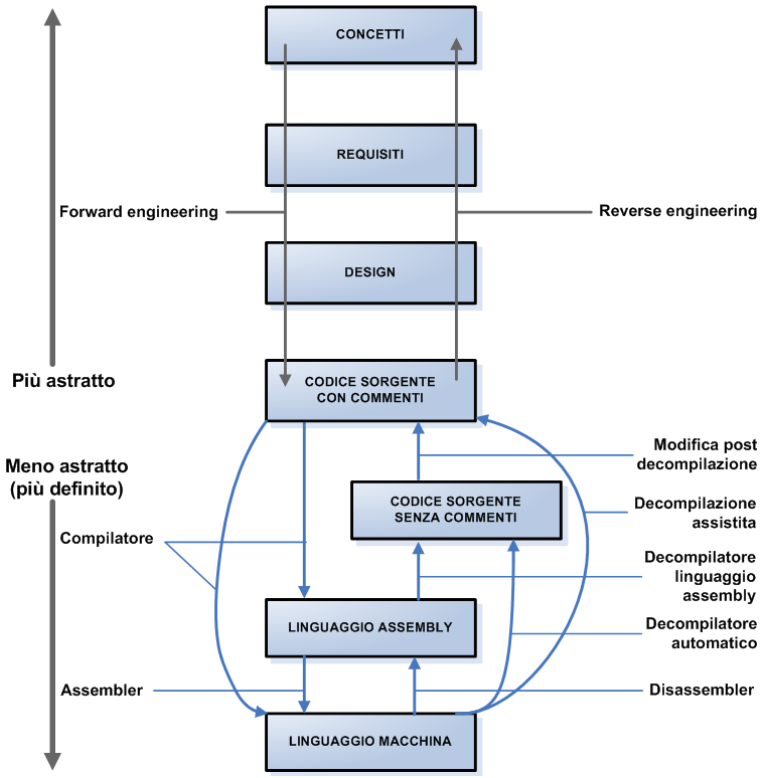

Figura 4.3: Ciclo di vita del software, dai concetti e requisiti fino al linguaggio macchina

in ambito di codice SQL. Si intende partire dal livello più basso, orientato ai dati, fino ad arrivare al livello più alto, orientato al problema da risolvere. Passare da un livello all'altro comporta sempre una trasformazione delle informazioni, ed anche una perdita di significato, in quanto è abbastanza difficile capire, partendo dal livello fisico, per quale scopo sia stato sviluppato tale database.

Il lavoro che è stato svolto quindi riguarda diverse attività (comprensione del codice, analisi di impatto, modifica del codice e validazione) su programmi sviluppati in Access e da dover effettuare il porting su un sistema Oracle, cercando di migliorarne l'accesso ai dati, le prestazioni e, soprattutto, che sia trasparente all'utente finale. L'insieme di queste operazioni possono essere definite *re-engineering*.

Fino ad adesso gli utenti, utilizzando Access, potevano interagire direttamente sui dati attraverso i diversi programmi creati e messi a loro disposizione, ma potendo interagire direttamente sui dati possono anche effettuare eventuali inserimenti/modifiche/cancellazioni non autorizzate: ecco quindi la necessità di creare un'applicazione apposita per il reparto destinatario, con le opportune query abilitate ad eseguire.

Con il temine porting su piattaforma Oracle non si intende il copiare le query, ma si intende il lavoro svolto di analisi e riscrittura completa delle stesse. Infatti i programmi in Access sono prevalentemente composti da diverse macro, dove ogni macro è composta da diverse query che estrapolano i dati seguendo una logica, e salvano i dati in locale

all'interno del file mdb (database locale di Access). Molte di queste query copiano i dati dal database server sul disco locale e da qui effettuano i filtri opportuni, ma ciò comporta un notevole spreco di spazio e di tempo per effettuare ciò. Inoltre non si utilizza la possibilità di far eseguire le query direttamente sul database server che risulta più veloce per effettuare calcoli, con un guadagno netto sia in termini di prestazioni e sia in termini di accesso ai dati.

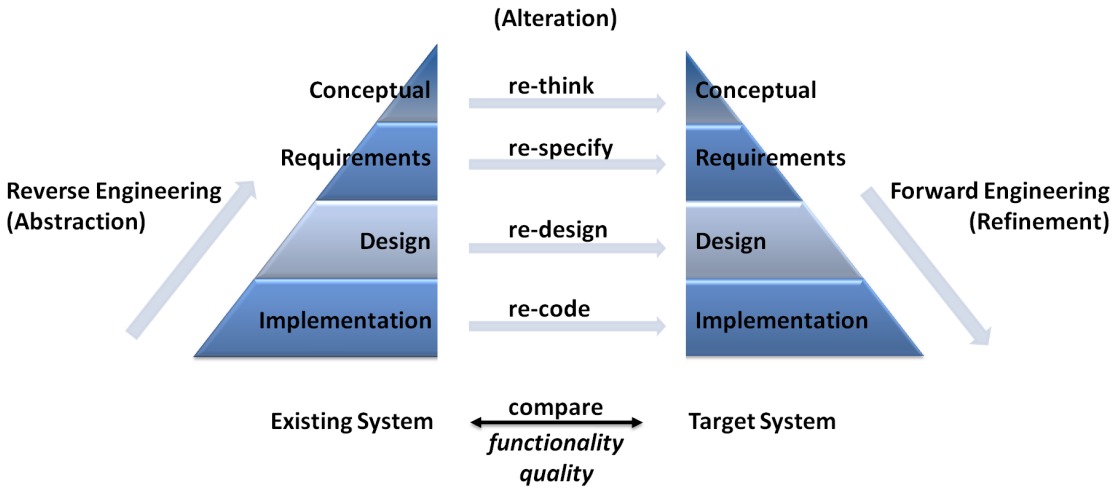

Figura 4.4: Reverse engineering e Forward engineering

Il reverse engineering che si è effettuato per ogni query/programma sviluppato è composto dai seguenti passi:

- analizzare le query, la logica di funzionamento e i risultati ottenuti dalla sua esecuzione;
- capire i punti chiave del contesto in cui si sta lavorando ed il motivo della creazione della query/programma;
- effettuare la scrittura delle query SQL attraverso il porting su piattaforma Oracle (non sono stati usati tool o altro visto che le tabelle non presentano legami di foreign key), cercando di migliorare il metodo di accesso ai dati e di rimuovere i passaggi inutili eliminando il superfluo in coerenza con i risultati attesi;
- testing su RDBMS Oracle e confronto dei risultati ottenuti con quelli aspettati.

4.3.1 Analisi, normalizzazione e denormalizzazione

Analizzando le tabelle presenti nel database di Apache si può subito capire come ad ogni tabella corrispondono diverse chiavi primarie, ma nessuna chiave esterna (o chiave

forestiera) collegata logicamente alla chiave primaria dell'altra relazione: ciò comporta la situazione di avere tante tabelle non "collegate" tra loro con la conseguenza che risulta difficile effettuare un lavoro sulle stesse senza un diagramma/modello da poter seguire. L'effetto di avere tabelle non logicamente collegate lo si ha dalla decomposizione, che si vedrà un po' più avanti, specialmente se si cambiano i nomi originali degli attributi, ma nonostante ciò, con un'attenta analisi, si può ricostruire il modello E/R, e magari ristrutturare le tabelle per poter far rispettare le forme normali.
Ad esempio, se si vogliono rappresentare le informazioni su alcuni prodotti (numero d'ordine, data dell'ordine, tipo di prodotto, nomi e indirizzi degli acquirenti) il tutto in una sola tabella risulta possibile, ma si ignora completamente la teoria della normalizzazione. Questa soluzione è caratterizzata tutt'altro che da vantaggi:

- l'informazione è ridondante;
- una ricerca può richiedere di scansionare tutte le righe dell'unica tabella;
- l'aggiornamento di un dato, tipo l'indirizzo di un compratore, può richiedere la modifica di più righe;
- la cancellazione di un ordine o di una riga può cancellare anche tutte le altre informazioni, come quelle del compratore se aveva fatto un solo ordine.

Per risolvere questi problemi è normale suddividere la tabella in diverse tabelle specializzate a contenere informazioni il più possibile correlate tra loro che rispettino il livello di qualità imposto dalla terza formale.
Per rappresentare i diversi livelli di qualità rispetto allo schema logico sono state introdotte delle forme normali: questo processo di verifica attuato tramite queste proprietà è detto appunto *normalizzazione*.
Ma perché avere un database con i vincoli e che rispetti le forme di normalizzazione?
La normalizzazione è un procedimento volto a verificare la complessità dei database rispetto a dei criteri per la presenza o meno di anomalie, come la ridondanza e l'inconsistenza delle informazioni presenti nel database. Questo processo si fonda su un semplice criterio: se una relazione presenta più concetti tra loro indipendenti, la si decompone in relazioni più piccole, una per ogni concetto. Questo tipo di processo non è purtroppo sempre applicabile in tutte le tabelle, dato che in taluni casi potrebbe comportare una perdita d'informazioni.
Invece per *denormalizzazione* di un database si intende il processo per cui si portano i dati sottoposti ad un processo di normalizzazione a quelli originari. Per sua natura la denormalizzazione porta a una ripetizione dei dati o all'aggiunta di dati non necessari. Si dice quindi denormalizzata una relazione non in forma normale.

In genere la denormalizzazione porta alla rinuncia alla terza forma normale, anche se in certi casi occorre rinunciare anche alla seconda forma normale. Questo è il caso tipico in cui una relazione molti-a-molti viene risolta con una tabella con chiave primaria che sostituisce un insieme di campi che altrimenti sarebbero una chiave candidata multipla.
La denormalizzazione è molto usata specie in contesti come il *datawarehouse*. In quel contesto, dove la velocità di risposta del database ad una query ha un'importanza maggiore rispetto all'organizzazione dei dati, è utile avere una denormalizzazione delle tabelle per evitare di fare delle join aggiuntive. Generalmente la denormalizzazione si usa trasformando le relazioni gerarchiche del tipo uno-a-molti in un'unica relazione e quindi in un'unica tabella.
Quindi i principali vantaggi della normalizzazione sono:

- l'eliminazione della ridondanza dell'informazione: l'informazione è memorizzata una volta sola e non duplicata;
- la semplicità di gestione dell'informazione: l'informazione, non essendo duplicata, può essere corretta o eliminata senza particolari ricerche e senza il rischio di non completare correttamente l'operazione.

Tali vantaggi sono persi con una struttura sottoposta al processo di denormalizzazione in quanto:

- l'informazione viene duplicata (e quindi c'è una maggiore occupazione di spazio di memoria);
- l'informazione, se errata, deve essere corretta in più punti e si incorre nella possibilità, in caso di eliminazione, di lasciare presenti accidentalmente dei dati che si ritenevano rimossi;
- la logica del programma che deve gestire quanto precedentemente esposto è necessariamente più complessa rispetto a quella che gestisce un database normalizzato.

La presenza dell'informazione denormalizzata rende però molto più rapida la ricerca e la presentazione dei dati, in quanto l'informazione è più accessibile e la struttura realizzata in genere è molto più vicina a quella di immediata fruizione rispetto a quella ottimizzata per la gestione.

4.4 Perché non utilizzare dei tool automatici?

In commercio e nel mondo accademico esistono diversi programmi che sono classificati come tool di reverse engineering. Tra i tool più utilizzati per scopi di comprensione del codice troviamo:

1. ***pretty printer***: formattano il codice in una forma più leggibile. Questi programmi si rivelano particolarmente utili tutte le volte che il codice è stato scritto con convenzioni di formattazione obsolete o particolarmente difficili da capire. In tale ambito possiamo citare *DMS* [24], un tool commerciale di trasformazione automatica del codice che dispone tra le varie funzionalità anche quella di ri-formattare il codice.

2. ***visualizzatori di codice*** (*code viewer*): producono viste alternative del software. Alcune sono testuali altre grafiche. A questa categoria appartengono i generatori di diagrammi di flusso, per esempio il tool commerciale *Code Visual to Flowchart* [25], ed i navigatori di codice, per esempio il tool *Source-Navigator* [26] rilasciato in licenza GNU (General Public License) che permettono di "navigare" con facilità all'interno del codice grazie all'ausilio di particolari viste ad alto livello.

3. ***generatori di diagrammi***: analizzano il codice sorgente e producono in output dei diagrammi che rappresentano alcune proprietà del sistema. Tra i diagrammi generati troviamo il "data flow", il grafo delle chiamate, il grafo delle dipendenze tra file e il grafo delle relazioni di ereditarietà tra le varie classi di un sistema ad oggetti. Un tool commerciale che rientra in questa categoria e produce in output diversi tipi di diagrammi, tra i quali il grafo delle chiamate e il grafo delle dipendenze tra file, è *Imagix 4D* [27].

4. ***tool di analisi***: recuperano dal codice sorgente un insieme di metriche utili ai fini della comprensione. Alcune metriche sono semplici, come le linee di codice o il numero di metodi/funzioni per file, altre sono più complesse come la coesione e l'accoppiamento tra moduli.

5. ***tool di recupero del design***: recuperano le relazioni interne tra le varie componenti software (per esempio, relazioni tra classi nei sistemi ad oggetti) e producono in output un diagramma che rappresenta, in un qualche formalismo, il design del sistema analizzato. Un esempio di tool commerciale che estrae diagrammi UML da codice Java è *EclipseUML* [28]. Oltre a ciò, la stessa azienda offre soluzioni anche per il database reverse engineering.

6. ***tool di generazione ed analisi delle tracce di esecuzione***: l'uso di informazione dinamica, cioè informazione raccolta durante l'esecuzione del software, è stata utilizzata spesso nel contesto del reverse engineering ad integrazione delle tecniche statiche (cioè quelle tecniche che si basano solo sull'analisi del codice). Le tecniche statiche sono conservative, nel senso che ogni possibile comportamento del sistema sotto analisi è rappresentato nei risultati, ma meno potenti di quelle dinamiche (problema dell'indecidibilità di alcune asserzioni a livello statico). Le tec-

niche di analisi dinamica hanno tra i vantaggi la garanzia dell'esistenza di almeno un caso in cui il comportamento del sistema è quello descritto dall'analisi. Tra gli svantaggi troviamo la parzialità delle analisi (non tutti i comportamenti del sistema sono rappresentati nei risultati) e il limite in termini di scalabilità ed interpretazione dei risultati.

7. ***tool di pattern matching***: ricercano, all'interno del codice, i cosiddetti pattern, ovvero soluzioni software consolidate per un problema ricorrente. I pattern possono essere a diversi livelli di astrazione: a livello di design (design pattern), a *livello di architettura* (per esempio, l'architettura client-server) e a *livello concettuale*. Il riconoscimento in modo automatico di alcuni pattern a livello sintattico o meglio ancora di alcuni concetti a livello semantico semplifica la fase di comprensione del codice.

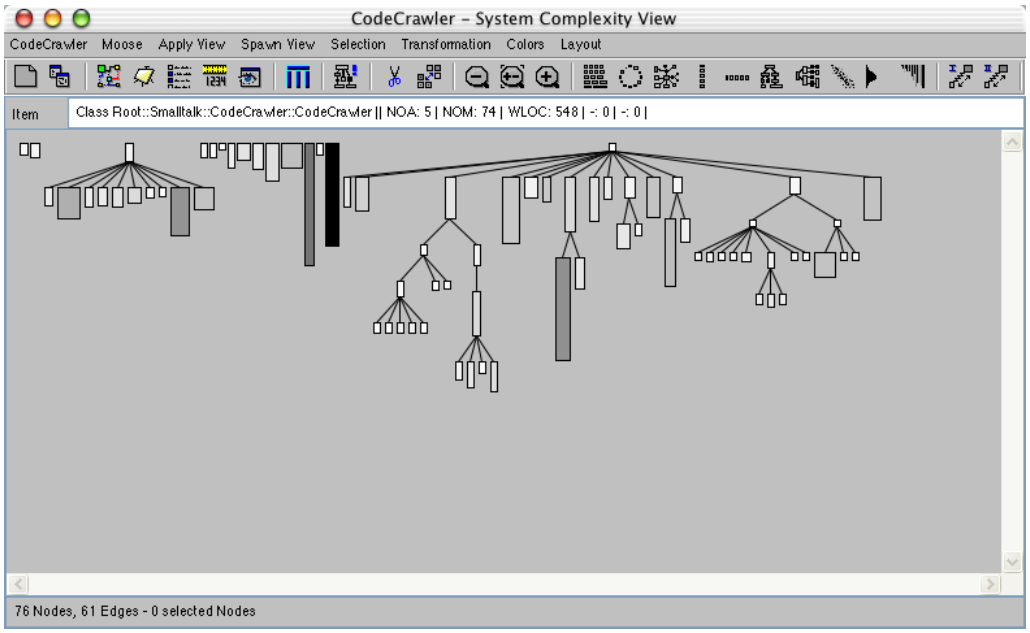

Figura 4.5: Vista "complessità del sistema" prodotta da CodeCrawler

Tra i tanti tool di reverse engineering quelli che forse hanno avuto un maggior impatto a livello industriale sono stati *Codecrawler* e *Rigi*.
Il Codecrawler [29] è un tool open-source di visualizzazione del codice che permette diversi tipi di viste grafiche.
Questo tool, scritto completamente in VisualWorks Smalltalk, è stato usato con successo in diversi progetti industriali di reverse engineering.
L'altro tool che vogliamo menzionare è Rigi [30], un tool altamente personalizzabile che permette di ricavare alcune viste e report (decomposizione del sistema basata sui tipi, struttura a file, architettura del sistema, analisi delle librerie condivise, ecc.) dal codice in

modo interattivo o non interattivo. Gli autori sostengono che per sistemi molto grandi è necessario lasciare all'utente la possibilità di manipolare le viste in modo non-interattivo, ovvero facendo uso di script.
Rigi supporta un linguaggio di scripting basato su Tcl (Tool Command Language) ed un esteso insieme di comandi con i quali è possibile produrre astrazioni personalizzate del sistema e report. Il tool può analizzare diversi linguaggi di programmazione.

4.5 Potenzialità e limiti riscontrati

Bisogna essere realistici a riguardo dei risultati che si possono ottenere da un'operazione di reverse engineering. Sia in ambito industriale che in quello accademico si sa che è un compito laborioso, difficile, dai costi molto elevati e non è mai fine a se stesso; dopo la fase di reverse engineering spesso seguono fasi altrettanto difficili e laboriose come il *restructuring* o il *re-engineering* (visti in precedenza).
Il limite più grande di questo approccio è che una parte sostanziale di lavoro deve essere svolta manualmente. I tool sono utili, ma vanno "guidati" nel processo di recupero delle informazioni, ed inoltre i risultati ottenuti non sono quasi mai immediatamente usabili e per essere utili devono essere raffinati manualmente.
I risultati prodotti in modo automatico devono essere, da un lato, integrati; infatti non tutte le informazioni utili possono essere ricavate dal codice o dalla sua esecuzione (informazione mancante) e, dall'altro, filtrati (sovrabbondanza di informazione). Capita spesso che la quantità di informazioni prodotte da un tool di reverse engineering sia ingestibile. L'altro difetto dei tools di estrazione del design è la sovrabbondanza di informazione. Spesso i diagrammi recuperati sono troppo grandi e quindi poco usabili. In un sistema ad oggetti medio, nell'ordine delle 20 mila linee di codice, è abbastanza comune avere 50-100 classi. In casi tipici come questo, il diagramma delle classi estratto dal codice è difficile da leggere per un umano le cui abilità cognitive sono attorno alle 10 entità come limite approssimativo.
Esistono due possibili soluzioni per cercare di risolvere questo problema:

- *il filtraggio*: operazione guidata dall'utente che esclude dal diagramma generato l'informazione irrilevante;
- *le viste multiple*: l'utente suddivide il sistema in viste e, durante la fase di reverse engineering, decide quali elementi (per esempio in un sistema ad oggetti: classi, metodi, campi, ecc.) appartengono a quale vista.

Le tecniche di reverse engineering, originariamente pensate per i sistemi legacy, si sono rivelate potenzialmente utili anche in supporto alla comprensione di sistemi software scritti secondo approcci più moderni, come i sistemi open source che solitamente sono rilasciati

senza o con poca documentazione, ad esclusione dei progetti ampiamente riconosciuti.

Attraverso i tool di reverse engineering si possono creare i *database model* direttamente da database esistenti. Questi strumenti mostrano graficamente la struttura stessa del database e gli elementi che lo compongono, come tabelle e viste, relazionate tra loro senza mostrare i dati memorizzati. Ciò permette di creare nuovi database o di aggiornarli capendo la struttura di uno già esistente.
In questo ambito esistono in commercio diversi software creati appositamente per aiutare il progettista a realizzare il lavoro più facilmente e velocemente; addirittura alcuni permettono di poter estrapolare il modello logico del database in modo del tutto trasparente all'utente, basta fornire i parametri minimi per stabilire una connessione, come *nome database*, *password di accesso*, *nome dell'host* (indirizzo IP oppure DNS (Domain Name System)) e *tipo di database* (Oracle, IBM DB2, Microsoft SQL Server, Microsoft Access, ecc.).
Questi sistemi sono abbastanza comodi se si deve estrarre lo schema E/R del database, le entità, convertire un database da una piattaforma ad un'altra (ad esempio da SQL Server verso Oracle o viceversa) ed includono tante altre funzionalità per la gestione delle tabelle, indici, viste, ecc.; nel nostro caso, visto che si ha un database senza relazioni tra le diverse tabelle, si possono solo avere i dettagli delle singole tabelle ma sono non utili da sole per poter effettuare il reverse engineering di cui si ha bisogno. Di conseguenza il lavoro risultante di tutte le fasi di reverse engineering e di porting sono state interamente effettuate via cognitiva e manuale. Inoltre questi sistemi non sono del tutto automatici e richiede sempre la presenza di una persona qualificata per svolgere il lavoro.

È stato utilizzato il tool *Entity Developer* per poter effettuare l'estrazione delle tabelle e dei campi presenti nel database di Apache. *Entity Developer for dotConnect* è uno strumento della Devart che permette di generare modelli e codice per LINQ to SQL e ADO.NET Entity Framework. Permette di progettare un'entità model partendo da zero oppure di poter effettuare un reverse engineering da un database esistente, includendo anche le relazioni se esistenti. Questo tool verrà affrontato in modo più approfondito nel capitolo 5, dove si parlerà di integrazione con l'ambiente di sviluppo Visual Studio e LINQ (Language-Integrated Query). Questo non è l'unico tool esistente per effettuare ciò ma è stato ritenuto quello che più si addice alle nostre esigenze.

4.6 Alcuni esempi di database reverse engineering e porting realizzati

In seguito verrà prima descritto il problema affrontato per poi esplicare il lavoro svolto. Non sono presenti tutti i programmi realizzati, ma verranno introdotti due casi esempio di programmi che si distinguono per il fine e per la presenza di campi parametrici all'interno della query.
Dei diversi programmi realizzati, sono descritti in dettaglio:

- ***Controllo fatture***;
- ***Analisi movimento carico Venduto Macrofamiglia***.

4.6.1 Programma "Controllo fatture"

Il database di Access è denominato "*Controllo fatture*" e fa parte di una serie di programmi utilizzati negli uffici dell'Amministrazione per estrarre informazioni operando sulle fatture (testata, righe e anagrafica fornitori) del ciclo passivo.
Nell'immagine sono raffigurate le tre situazioni che troviamo dentro il database riguardo le tabelle, le query e le macro. Si può notare come il database abbia una tabella locale, essenzialmente di appoggio, e cinque tabelle pass-through che sono connesse direttamente al database di Apache. Nella seconda parte dell'immagine sono presenti diverse query (*query di creazione tabella*, *query di aggiornamento*, *query di selezione* e *query SQL pass-through*) ed infine, nell'ultima parte, si vedono le macro che vengono lanciate per eseguire le applicazioni.
Dopo aver lanciato la macro, i dati vengono elaborati, visualizzati su schermo e salvati sul disco locale: in alcuni casi la directory è impostata dalla stessa macro, ma nella maggior parte dei casi l'applicazione permette di far scegliere la locazione (attraverso la funzione *Salva con nome*) inserendo i dati dentro un foglio di calcolo.
Con Access è possibile utilizzare query pass-through, cioè un tipo di query che permette di accedere ad un database non Access e che risiede/può risiedere su un'altra macchina, ricavando i risultati che interessano mediante i comandi di quel sistema di database. Questo tipo di query, insieme a *Unione* e *Definizione dati*, sono accomunate dal fatto che non è prevista la possibilità di comporle graficamente, nella griglia di struttura, ma si deve scrivere direttamente l'enunciato SQL che le definisce.
In figura 4.6 è possibile vedere come graficamente Access differisce le due tipologie di tabelle, una normale, che risiede all'interno del database file (icona a forma di tabella), e le altre che sono "collegamenti" diretti alle tabelle esterne (icona a forma di mondo).

In elenco le macro presenti in questo programma con le rispettive abbreviazioni utilizzate per comodità descrittiva:

- ***Controllo ordini fatture*** - *COFT*;
- ***Controllo ordini fatture riepilogo totale*** - *COFRT*;
- ***Controllo ordini fatture non confermate*** - *COFNC*.

Ogni macro è composta da diverse query, salvate all'interno del database di Access, dove la maggior parte di tali query sono di creazione tabella e di aggiornamento, ma ciò non toglie la possibilità di aver la presenza di alcune specifiche azioni, come il disabilitare i messaggi di sistema ed l'esportazione dei dati visualizzati su file.

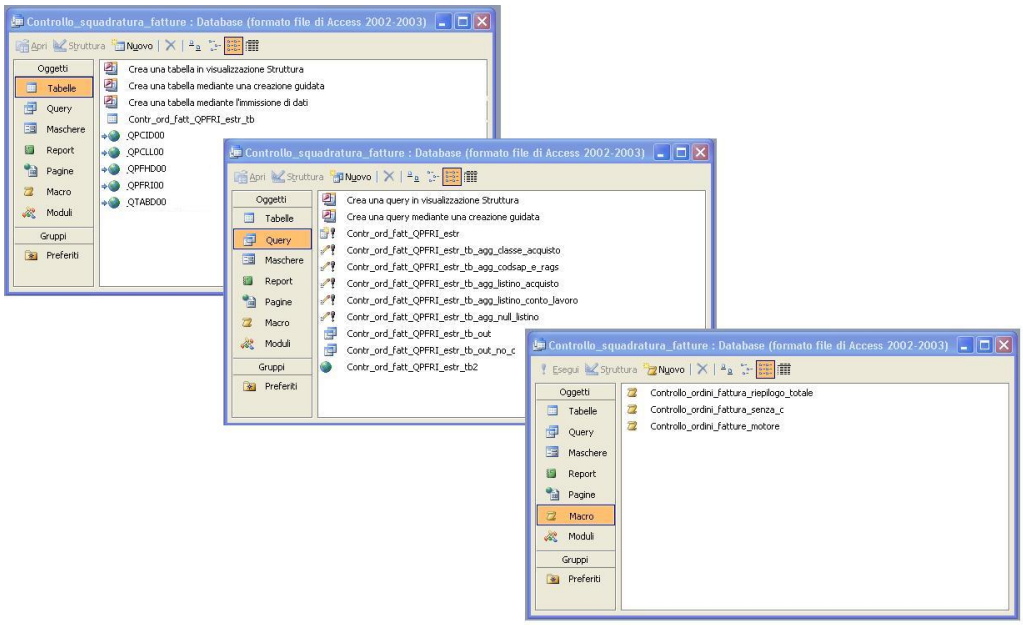

Figura 4.6: Tabelle, query e macro presenti nel database "*Controllo ordini fatture*"

C'è anche la possibilità di poter includere una macro in altre: infatti le macro *COFRT* e *COFNC* includono di base *COFT*, come si può notare nella figura 4.7. Poiché le due macro sono costruite basandosi sopra ad una, è opportuno iniziare ad analizzare quella basilare per poi poter procedere alla costruzione della nuova query in SQL.
In questo caso le due macro non differiscono di molto da quella basilare, ma si discostano di un filtro aggiuntivo posto alla fine delle query per poter estrarre e visualizzare le informazioni richieste dal singolo programma.
In questi programmi non sono presenti finestre di inserimento dati che si interfacciano con l'utente, ma in alcuni casi le applicazioni possono richiedere l'inserimento di date del periodo di interesse. Infatti per poter inserire nelle nuove query su Oracle l'interattività

fornita dalle query di Access abbiamo dovuto utilizzare le query parametriche. Una query parametrica è una query costruita con gli stessi criteri usati per qualsiasi query di selezione nella quale si vogliano porre delle limitazioni al contenuto di qualche campo (vincoli). Invece di avere una limitazione fissa, essa può essere scelta tramite un parametro immesso dall'utente al momento del lancio.

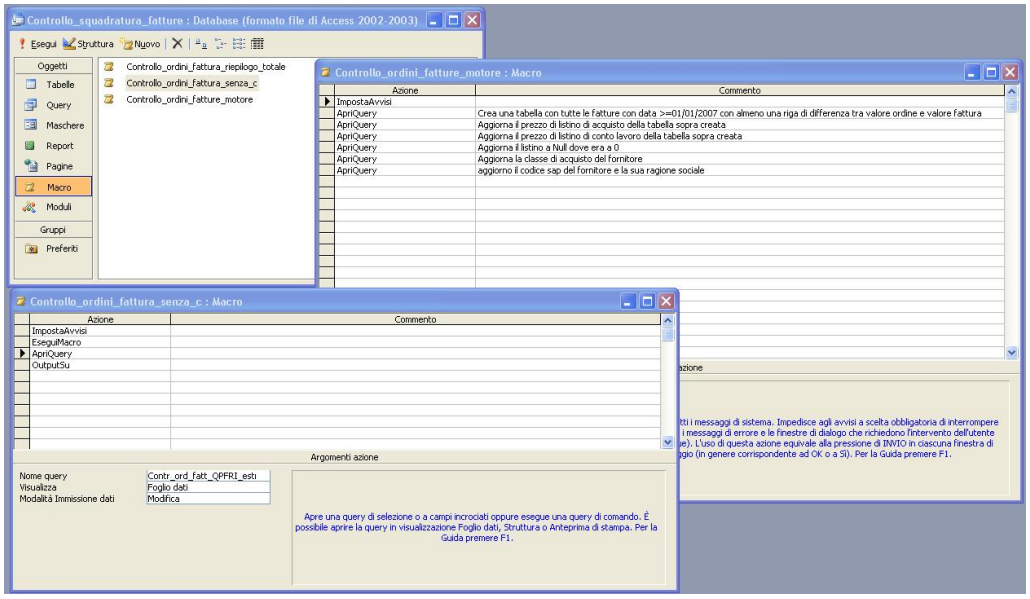

Figura 4.7: Caratteristiche di una macro contenente un'altra

Le query sono formate in cascata, ovvero partendo da una query che essenzialmente seleziona diversi campi da una o più tabelle, le altre interessate ne elaborano i dati eseguendo diversi join e raggruppamenti. La filosofia di questo tipo di query è quella di copiare in locale inizialmente tutti dati di cui si ha bisogno per poi procedere al filtraggio, ma ciò comporta un costo davvero oneroso in termini di prestazioni, di tempo e di risorse occupate dalla singola applicazione.
Infatti i programmi possono impiegare diversi minuti prima di poter dare i risultati finali, ma con il nuovo approccio che si è voluto sviluppare questo è uno dei punti chiave dove si cercherà di dare una soluzione.
In questo caso il programma in questione è abbastanza semplice, ma altri programmi analizzati sono molto più complessi e con un numero di query elevato, e la loro esecuzione può arrivare ad impiegare un tempo variabile tra 5 e 10 minuti. Generalmente questo tipo di query non sono di semplice estrazione dati, ma sono utilizzate per effettuare particolari report, ad esempio sul venduto, sul fatturato, sui movimenti di magazzino, ecc.

4.6.1.1 Macro "Controllo ordini fatture"

Questa macro estrae le informazioni operando sulle fatture (testata, righe e anagrafica fornitori) del ciclo passivo. Le tabelle interessate nella macro dalle diverse query sono:

- *QPFHD00*: *Testata fatture passive*;
- *QPFRI00*: *Righe fatture passive*;
- *QPCLL00*: *Righe del listino di acquisto*;
- *QPCID00*: *Anagrafica fornitori*;
- *QTABD00*: *Descrizioni dei parametri applicativi.*

I database presenti e gestiti all'interno dell'azienda sono vari, di cui alcuni dedicati per operazioni di testing: questi sono utili soprattutto quando si devono effettuare operazioni critiche, a discapito di non avere dati perfettamente aggiornati (la copia del database viene effettuata una o due volte al mese), ma la struttura essenzialmente è la stessa.
Questa macro è parte integrante degli altri programmi che sono descritti nei paragrafi successivi a questo, quindi estrae più informazioni di quelle che si ha effettivamente bisogno. Infatti nelle macro successive si procederà a filtrare meglio tali risultati.
La tabella *QTABD00* è una delle tabelle create appositamente all'interno del database in modo tale da poter contenere diverse informazioni aggiuntive per poter soddisfare le esigenze dei reparti. Infatti tale tabella contiene pochi campi a differenza delle altre e con più attributi per poter diminuire il costo di accesso alle informazioni a scapito della ridondanza dei dati.
In figura 4.8 viene visualizzato in dettaglio come è composta la macro, con il nome delle query presenti all'interno, il tipo ed una breve descrizione del suo operato.

- **1° passo**: la query iniziale parte dalla tabella *QPFRI00* (tabella che contiene le righe delle fatture del ciclo passivo) effettuando il join con la tabella *QPFHD00* (tabella che contiene invece le testate delle fatture del ciclo passivo). Da qui vengono "portati avanti" tutti i dati essenziali di una fattura, come codice fornitore, codice fattura, codice riga, codice prodotto, descrizione, data fattura, quantità, valore fattura, numero fattura, flag conferma fattura, ecc. I dati estratti sono memorizzati nella tabella "*Contr_ ord_fatt_ QPFRI_ estr_ tb*", creato, ed eventualmente sovrascritto se già esistente, nel database locale di Access.

- **2° passo**: viene presa la tabella *QPCLL00* contente le righe del listino di acquisto e posta in join con la tabella precedentemente memorizzata apponendo determinati filtri su diversi campi. Poiché è una query di aggiornamento, le variazioni vengono salvate sovrascrivendo la tabella precedentemente salvata ed utilizzata.

Passo	Azione	Nome query	Query di	Descrizione
1	Apri Query	Contr_ord_fatt_QPFRI_estr	Creazione tabella	Crea una tabella con tutte le fatture con data ≥ 01/01/2007 con almeno una riga di differenza tra valore ordine e valore fattura
2	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_agg_listino_acquisto	Aggiornamento	Aggiorna il prezzo di listino di acquisto della tabella sopra creata
3	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_agg_listino_conto_lavoro	Aggiornamento	Aggiorna il prezzo di listino di conto lavoro della tabella sopra creata
4	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_agg_null_listino	Aggiornamento	Aggiorna il listino a Null dove era uguale a 0 (il Null nel database viene considerato come una serie di 9)
5	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_agg_classe_acquisto	Aggiornamento	Aggiorna la classe di acquisto del fornitore
6	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_agg_codsap_e_rags	Aggiornamento	Aggiorna il codice SAP del fornitore e la sua ragione sociale

Figura 4.8: Elenco dei passi eseguiti dalla macro "*Controllo ordini fatture*"

- **3° passo**: nuovo join con la *QPCLL00* apponendo diversi filtri che includono nuovi record ed aggiorna nuovamente la tabella.
- **4° passo**: aggiornamento della tabella escludendo i valori NULL.
- **5° passo**: la tabella è posta in join con la tabella *QPCID00* (relativa all'anagrafica fornitori), a sua volta con la tabella *QTABD00* (contenente l'elenco dei parametri applicativi).
- **6° passo**: come ultimo passo la tabella viene posta infine nuovamente in join con la tabella *QPCID00* per selezionare i campi ultimi da visualizzare sulla tabella finale.

Molti di questi passi sono stati realizzati per poter far svolgere all'applicazione i passi necessari con il fine di raggiungere i risultati aspettati, ma analizzando in dettaglio ci sono molte parti che non sfruttano alcune peculiarità dei dati e prelevano dati inutili ai fini dell'applicazione. Infatti, andando ad analizzare meglio, alcuni passi possono essere eseguiti "insieme", come i diversi join dei passi 2 e 3, ed alcuni addirittura esclusi, come il passo 4.
Nella figura 4.13 c'è il codice SQL che deriva dall'analisi e si nota come attraverso SQL innestate si può aver lo stesso risultato del programma Access, ma con un notevole risparmio di tempo e risorse. Il codice può essere eseguito direttamente sul server Oracle attraverso un SQL tool, come SQL Navigator o con un altro software con le stesse caratteristiche, oppure tramite un programma appositamente realizzato che permette di eseguire le query SQL, con la possibilità di poter eseguire anche query parametriche.
Inoltre verificando i tempi impiegati dal server per poter eseguire le query, si può notare

come impieghi sempre meno tempo per eseguire l'elaborazione per effetto del caching effettuato sul server.

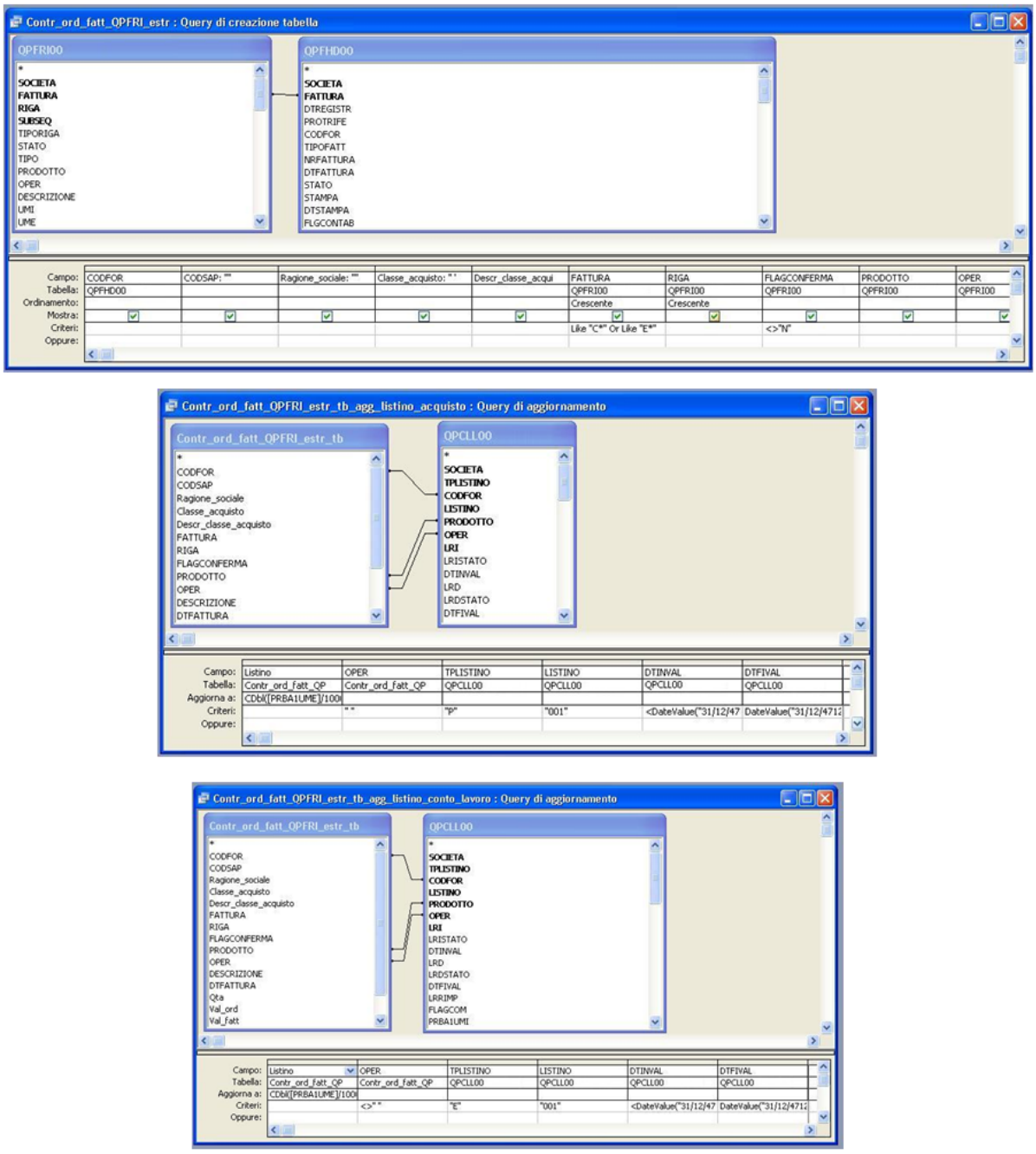

Figura 4.9: Passi della macro "*Controllo ordini fatture*"

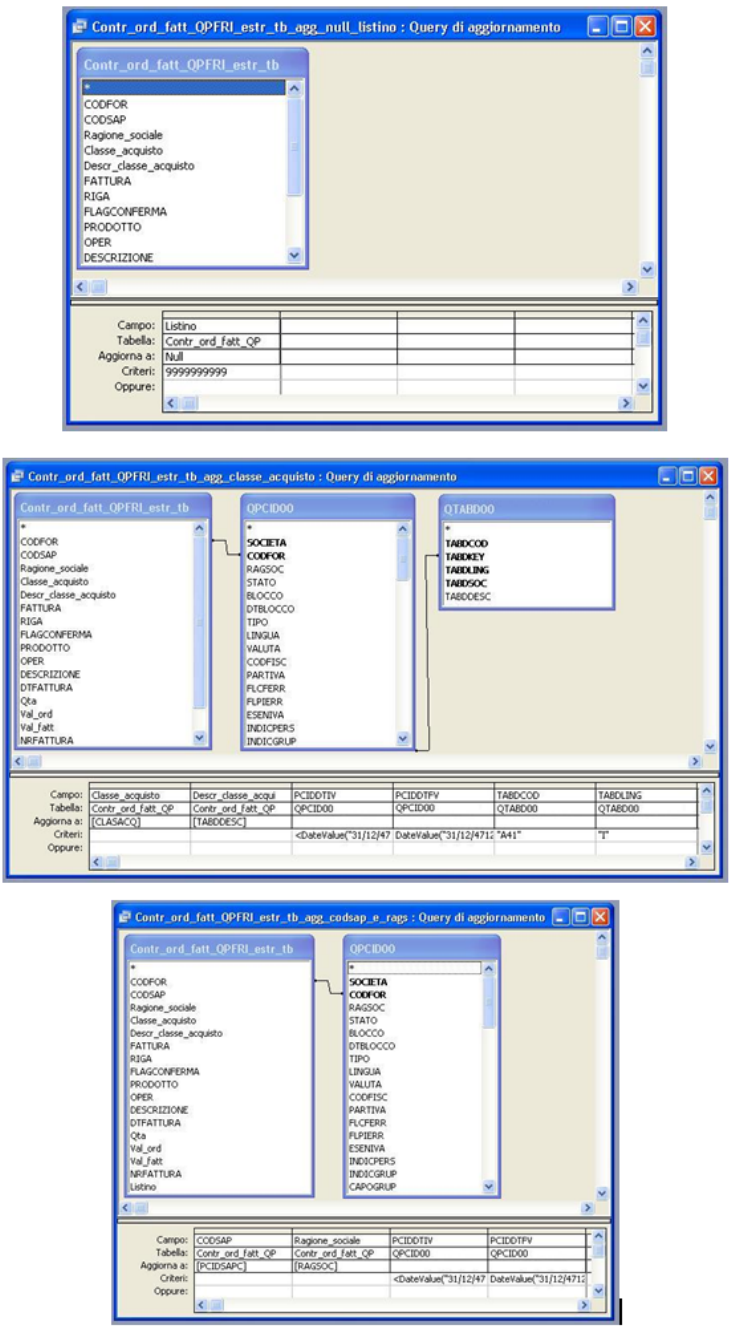

Figura 4.10: Passi (4-6) della macro "*Controllo ordini fatture*"

```
select z.codfor, d.pcidsapc as Codsap, d.ragsoc as Ragione_sociale, d.clasacq as Classe_acquisto,
       e.tabddesc as Descr_classe_acquisto, z.fattura, z.riga, z.flagconferma as Flagconferma, z.prodotto,
       z.oper, z.descrizione, z.dtfattura, z.Qta, z.Val_ord, z.Val_fatt, z.nrfattura,
       decode(z.oper, ' ', c.prbalume, f.prbalume)/10000 as Listino
from (
        select a.codfor, a.fattura, b.riga, b.prodotto, b.oper, b.descrizione, a.dtfattura,
               sum(b.quantita)/100 as Qta, sum(b.valoreord)/10000 as Val_ord, sum(b.valore)/10000 as Val_fatt,
               a.nrfattura, b.flagconferma
        from QPFHD00 a
        inner join QPFRI00 b on ( b.fattura = a.fattura and b.societa = a.societa
                                  and substr(b.fattura,1,1) in ('C','E')
                                )
        where a.dtfattura >= '01-jan-2007'
        and a.societa = '01'
        and b.flagconferma <> 'N'
        group by a.codfor, a.fattura, b.riga, b.prodotto, b.oper, b.descrizione, a.dtfattura, a.nrfattura,
                 b.flagconferma
) z
left join QPCLL00 c on ( c.codfor = z.codfor
                         and c.prodotto = z.prodotto
                         and c.oper = z.oper
                         and c.societa = '01'
                         and c.oper = ' '
                         and c.tplistino = 'P'
                         and c.listino = '001'
                         and c.dtinval < '31-dec-4712'
                         and c.dtfival = '31-dec-4712'
                         and c.flagcom <> ' '
                       )
left join QPCLL00 f on ( f.codfor = z.codfor
                         and f.prodotto = z.prodotto
                         and f.oper = z.oper
                         and f.societa = '01'
                         and f.oper <> ' '
                         and f.tplistino = 'E'
                         and f.listino = '001'
                         and f.dtinval < '31-dec-4712'
                         and f.dtfival = '31-dec-4712'
                       )
left join QPCID00 d on ( d.societa = '01'
                         and d.codfor = z.codfor
                         and d.pciddtiv < '31-dec-4712'
                         and d.pciddtfv = '31-dec-4712'
                       )
left join QTABD00 e on ( e.tabdkey = d.clasacq
                         and e.tabdcod = 'A41'
                         and e.tabdling = 'I'
                         and e.tabdsoc = ' '
                         )
group by z.codfor, d.pcidsapc, d.ragsoc, d.clasacq, e.tabddesc, z.fattura, z.riga, z.flagconferma, z.prodotto,
         z.oper, z.descrizione, z.dtfattura, z.Qta, z.Val_ord, z.Val_fatt, z.nrfattura, decode(z.oper, ' ',
         c.prbalume,f.prbalume)/10000
```

Figura 4.11: Query SQL della macro "*Controllo ordini fatture*"

4.6.1.2 Macro "Controllo ordini fatture riepilogo totale"

Questa macro si basa completamente su quella precedente, ma con l'aggiunta di una query finale di selezione che imposta due altri vincoli sui campi, visualizza i dati sullo schermo ed li esporta su un foglio elettronico (Controllo_fatture_ordini.xls), eventualmente sovrascritto se già esistente. Ora viene visualizzato in dettaglio come è composta la macro, con il nome delle query presenti all'interno, il tipo ed una breve descrizione del suo operato.

Dalla precedente query scritta per Oracle, è abbastanza facile ricavare tale seguente query inserendo due vincoli in fondo (*clasacq = 001*, *dtfattura >= 01-jan-2008*), prima del raggruppamento finale.

Passo	Azione	Nome query	Query di	Descrizione
1	Esegui Macro	Controllo_ordini_fatture		Esegue tutte le query presenti nella macro
2	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_ out	Selezione	Aggiunge alcuni filtri al risultato finale delle query precedenti
3	Output Su Excel			Esporta i dati su un foglio elettronico (spreadsheet)

Figura 4.12: Elenco dei passi eseguiti dalla macro "*Controllo ordini fatture riepilogo totale*"

```
...
left join QTABDOO e on ( e.tabdkey = d.clasacq
                         and e.tabdcod = 'A41'
                         and e.tabdling = 'I'
                         and e.tabdsoc = ' '
                         )
where d.clasacq = '001'
and z.dtfattura >= '01-jan-2008'
group by z.codfor, d.pcidsapc, d.ragsoc, d.clasacq, e.tabddesc, z.fattura, z.riga, z.flagconferma, z.prodotto,
         z.oper, z.descrizione, z.dtfattura, z.Qta, z.Val_ord, z.Val_fatt, z.nrfattura, decode(z.oper, ' ',
         c.prba1ume,f.prba1ume)/10000
```

Figura 4.13: Query SQL della macro "*Controllo ordini fatture riepilogo totale*"

4.6.1.3 Macro "Controllo ordini fatture non confermate"

Anche questa macro si basa completamente su *COFT* ed ha le stesse caratteristiche della *COFRT* a differenza di un filtro aggiunto alla fine. I dati visualizzati sullo schermo vengono esportati su un foglio elettronico (*Controllo_fatture_ ordini.xls*), eventualmente sovrascritto se già esistente.
Ora viene visualizzato in dettaglio come è composta la macro, con il nome delle query presenti all'interno, il tipo ed una breve descrizione del suo operato.
Dalla precedente query scritta, è abbastanza facile ricavare tale query inserendo una clausola in fondo (*flagconferma <> 'C'*), prima del raggruppamento finale.
Il campo flagconferma è l'indicatore dello stato di conferma della riga e viene utilizzato per indicare:

- A: In attesa nota;
- C: Controllata;
- D: Da confermare;
- I: Verifica in corso;
- M: Chiusa manualmente;
- N: Nessuna conferma.

Quindi si va a selezionare tutte le righe di fattura che non sono state ancora confermate.

Passo	Azione	Nome query	Query di	Descrizione
1	Esegui Macro	Controllo_ordini_fatture		Esegue tutte le query presenti nella macro
2	Apri Query	Contr_ord_fatt_QPFRI_estr_tb_out_no_c	Selezione	Aggiunge alcuni filtri al risultato finale delle query precedenti
3	Output Su Excel			Esporta i dati su un foglio elettronico (spreadsheet)

Figura 4.14: Elenco dei passi eseguiti dalla macro "*Controllo ordini fatture non confermate*"

```
...
left join QTABDOO e on ( e.tabdkey = d.clasacq
                         and e.tabdcod = 'A41'
                         and e.tabdling = 'I'
                         and e.tabdsoc = ' '
                         )
where d.clasacq = '001'
and z.dtfattura >= '01-jan-2008'
and z.flagconferma <> 'C'
group by z.codfor, d.pcidsapc, d.ragsoc, d.clasacq, e.tabddesc, z.fattura, z.riga, z.flagconferma, z.prodotto,
         z.oper, z.descrizione, z.dtfattura, z.Qta, z.Val_ord, z.Val_fatt, z.nrfattura, decode(z.oper, ' ',
         c.prba1ume,f.prba1ume)/10000
```

Figura 4.15: Query SQL della macro "*Controllo ordini fatture non confermate*"

4.6.1.4 Lo Schema E/R ricavato

Le tabelle utilizzate dal database di Apache presentano tutte la caratteristica di non avere nessuna chiave forestiera, ma solo chiavi primarie e ciò lo si può notare dalle diverse tabelle che presentano molti campi, solitamente da 10 a 60 attributi.
Nelle macro affrontate le relazioni tra le tabelle sono abbastanza semplici, di solito su uno o due attributi, invece sui vincoli sono impostati molti criteri sulle date di validità (spesso per definire i livelli di revisione) e su alcuni attributi.
Nel programma sopra trattato le macro sono basate tutte su una, quindi verrà mostrato lo schema E/R ricavato dalla macro basilare (Macro "*Controllo ordini fatture*").

QGDIF00: Dati non storicizzati dell'articolo
PK: Gdifitem, Gdiffini
QGDIA00: Dati tecnici dell'articolo
PK: Gdiaitem, Gdiafini, Gdialvrv
FK: Gdiaitem REFERENCES QGDIF00(Gdifitem)
FK: Gdiafini REFERENCES QGDIF00(Gdiffini)
QGDLN00: Descrizioni in lingua dell'articolo
PK: Gdlnitem, Gdlnfini
FK: Gdlnitem *REFERENCES* QGDIF00(Gdifitem)
FK : Gdlnfini *REFERENCES* QGDIF00(Gdiffini)

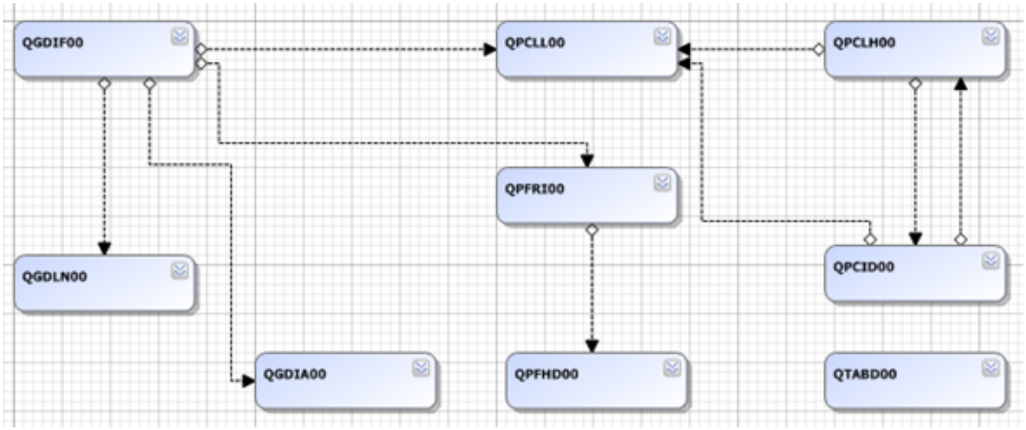

Figura 4.16: Schema generale ricavato dalla macro "*Controllo ordini fatture non confermate*"

QPCID00: Anagrafica fornitori
PK: Codfor, Pcidlvrv
QPFHD00: Testata fatture passive
PK: Fattura
QPFRI00: Righe fatture passive
PK: Fattura, Riga, Subseq
FK: Fattura *REFERENCES* QPFHD00(Fattura)
FK: Prodotto *REFERENCES* QGDIF00(Gdifitem)
QPCLH00: Testata del listino di acquisto
PK: Tplistino, Listino, Codfor, Lri
FK: Codfor *REFERENCES* QPCID00 (Codfor)
FK: Lri *REFERENCES* QPCLG00(Modtec)
QPCLL00: Righe del listino di acquisto
PK: Tplistino, Listino, Codfor, Lri, Prodotto
FK: Tplistino *REFERENCES* QPCLH00(Tplistino)
FK: Listino *REFERENCES* QPCLH00(Listino)
FK: Codfor *REFERENCES* QPCID00(Codfor)
FK: Lri *REFERENCES* QPCLG00(Modtec)
FK: Prodotto *REFERENCES* QGDIF00(Gdifitem)
QTABD00: Descrizioni dei parametri applicativi
PK: Tabdcod, Tabdkey, Tabdling
FK: Tabcod *REFERENCES* QTABX00(Tabxcodt)

Sopra sono state indicate solo le chiavi necessarie per poter identificare univocamente ogni record e le chiavi forestiere con le relative tabelle di appartenenza. Inoltre sono state

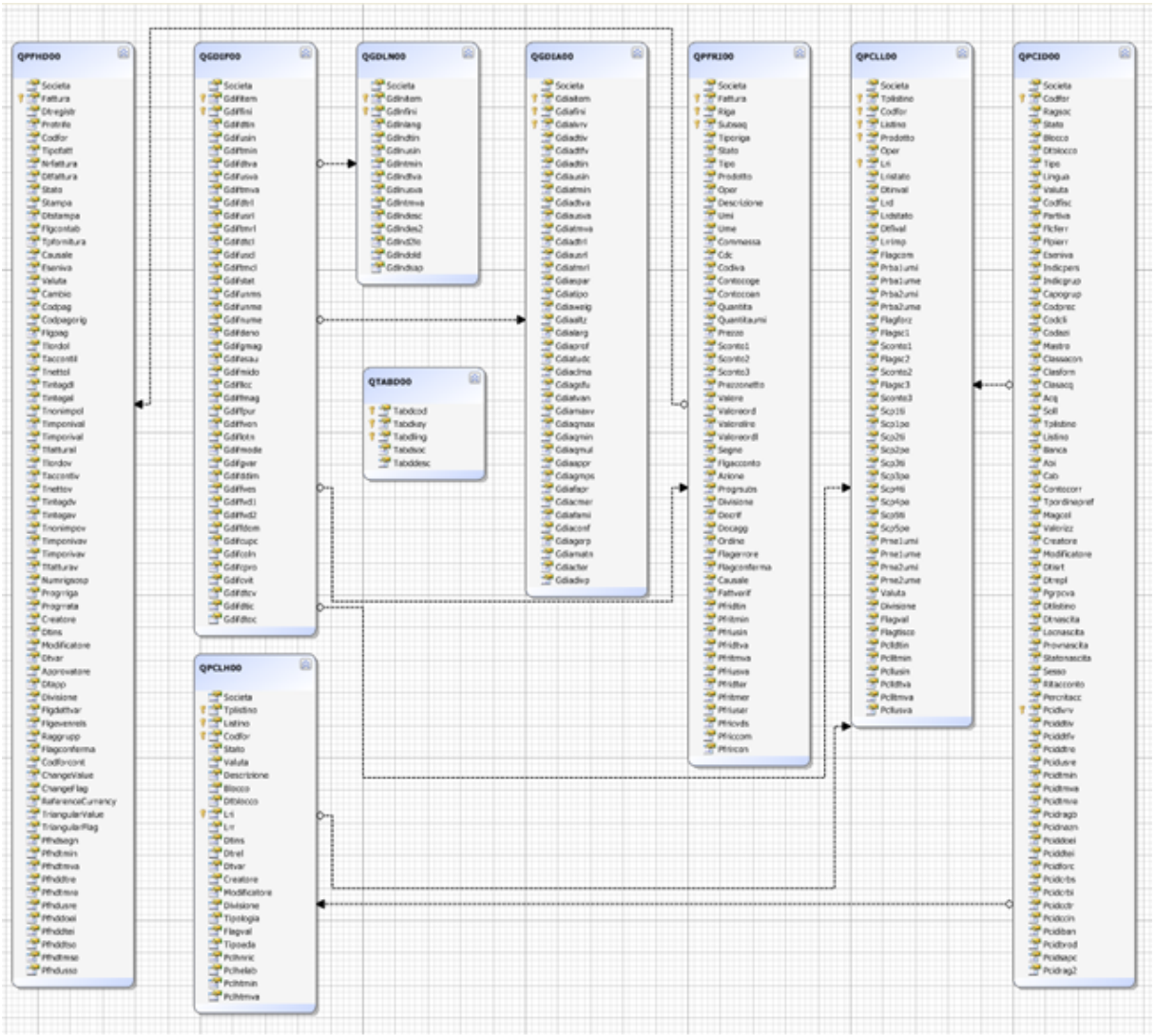

Figura 4.17: Schema dettagliato ricavato dalla macro "*Controllo ordini fatture non confermate*"

inserite nello schema anche altre tabelle non collegate direttamente nella query esaminata, ma logicamente implicate nello studio per la realizzazione dello schema, come la *QPCLG00* (*Livelli di revisione del listino di acquisto*), la *QTABX00* (*Testi dei parametri applicativi*) e le tabelle *QGDIF00* e *QGDIA00*, dove sono state analizzate stabilendo le chiavi e le relazioni con le altre tabelle.

Il recupero dello schema E/R non è stato per niente facile in quanto il database è stato strutturato in modo tale da avere diverse tabelle indipendenti tra loro e con un alto tasso di informazioni, il tutto per ridurre i tempi di ricerca (tipica caratteristica dei datawarehouse). Quindi il lavoro che si è potuto fare è stato quello di verificare quali attributi presentino caratteristiche simili per poterli "legare tra loro" attraverso chiavi primarie e foreign key. Se si vuole avere uno schema che rispetti le forme normali, soprattutto aspi-

rando almeno dalla *3ª forma normale* in su, si dovrebbe ristrutturare il database in modo più approfondito, ma così perdendo lo scopo principale per cui è stato realizzato nella forma attuale.
Da notare che le chiavi indicate negli schemi generali sottostanti sono state impostate attraverso il tool *Entity Developer* (descritto in precedenza) e non sono state ricavate attraverso un tool di database reverse engineering.

4.6.2 Programma "Analisi movimento carico Venduto Macrofamiglia"

Questo programma presenta le stesse caratteristiche di quello già presentato in precedenza, ma con la differenza che accetta anche le query parametriche. Da come si può vedere in figura 4.18 queste macro sono più pesanti poiché richiedono più passaggi intermedi.
Il database di Access è denominato "*Analisi movimento carico*" e fa parte di una serie di programmi utilizzati dall'ufficio di Amministrazione per estrarre le informazioni riguardo la movimentazione, la produzione e gli ordini di vendita. Il termine carico in ambito dei sistemi gestionali viene utilizzato per indicare l'aggiunta di risorse nel magazzino, al contrario di scarico quando vengono prelevati pezzi dal magazzino per poter realizzare i componenti.
In elenco le macro presenti in questo programma con le rispettive abbreviazioni utilizzate per comodità descrittiva:

- ***Movimenti acquisto conto pieno*** - *MACP*;
- ***Analisi mov prod interna e conto lavoro*** - *MPCL*;
- ***Venduto Macrofamiglia*** - *VDMF*.

Delle diverse macro sopra elencate, ci occuperemo più in dettaglio della *MACP* e della *VDMF* poiché le altre sfruttano la stessa ideologia, anche se sono state realizzate per scopi diversi.
Queste macro sono composte da sole query, quindi non è prevista la soluzione di una macro dentro ad un'altra.
Come si vede in figura, i programmi includono diverse tabelle, molte create dalle query interne (query di creazione), e ne risulta una non facile comprensione dell'operato delle diverse macro. Quindi per poter capire il principale motivo della creazione di tali query bisogna analizzare tutte le macro in dettaglio prima di poter effettuare il porting.

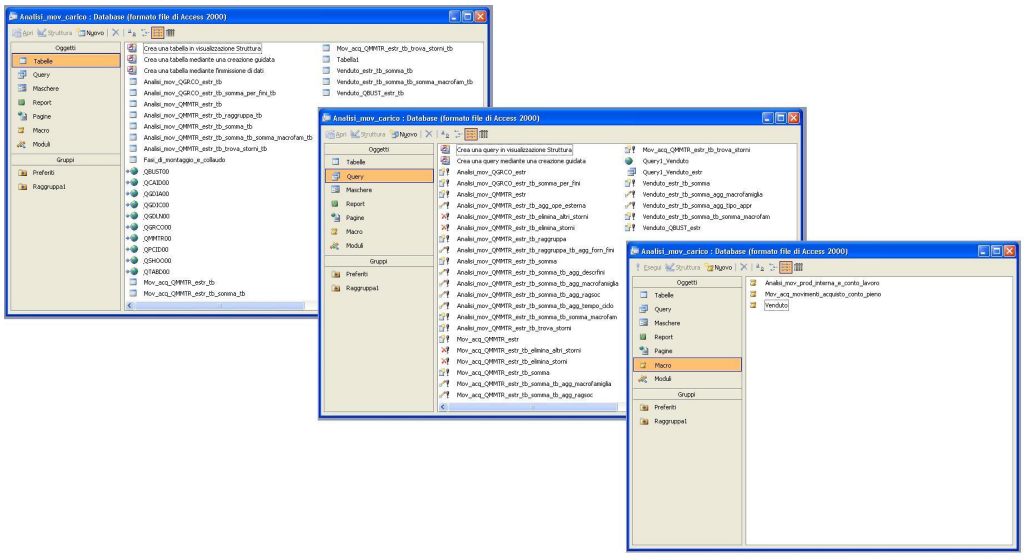

Figura 4.18: Tabelle, query e macro presenti nel database "*Analisi movimento carico Venduto Macrofamiglia*"

4.6.2.1 Macro "Movimenti acquisto conto pieno"

Questa macro estrae le informazioni operando sui movimenti di magazzino. Le tabelle interessate nelle diverse query sono:

- *QMMTR00: Movimenti di magazzino*;
- *QGDLN00: Descrizioni in lingua dell'articolo*;
- *QGDIC00: Parametri commerciali dell'articolo*;
- *QPCID00: Anagrafica fornitori.*

Ora viene visualizzato in dettaglio come è composta la macro, con il nome delle query presenti all'interno, il tipo ed una breve descrizione del suo operato.
Descrizione dei singoli passi:

- **1° passo**: la query iniziale parte dalla tabella *QMMTR00* (tabella che contiene i movimenti di magazzino) selezionando tutti i valori apponendo opportuni filtri. Qui è già presente l'interazione con l'utente per far inserire le date del range di riferimento. I dati filtrati sono memorizzati nella tabella "*Mov_acq_QMMTR_estr_tb*", creata nel database locale di Access ed eventualmente sovrascritta se già esistente.
- **2° passo**: cerca dalla tabella precedentemente memorizzata tutti i valori con il campo *segno='S'* e salva i risultati nella tabella "*Mov_acq_QMMTR_estr_tb_trova_storni_tb*". Attraverso il campo segno si determina il tipo di movimento (N indica *Normale*, mente S indica *Storno*).

Passo	Azione	Nome query	Query di	Descrizione
1	Apri Query	Mov_acq_QMMTR_estr	Creazione tabella	Estrae i movimenti di magazzino dalla QMMTR00
2	Apri Query	Mov_acq_QMMTR_estr_tb_trova_storni	Creazione tabella	Trova storni
3	Apri Query	Mov_acq_QMMTR_estr_tb_elimina_storni	Eliminazione	Elimina storni
4	Apri Query	Mov_acq_QMMTR_estr_tb_elimina_altri_storni	Eliminazione	Elimina altri storni
5	Apri Query	Mov_acq_QMMTR_estr_tb_somma	Creazione tabella	Somma le quantità
6	Apri Query	Mov_acq_QMMTR_estr_tb_somma_tb_agg_macrofamiglia	Aggiornamento	Aggiorna la macrofamiglia
7	Apri Query	Mov_acq_QMMTR_estr_tb_somma_tb_agg_ragsoc	Aggiornamento	Aggiorna la ragione sociale
8	Output Su Excel			Esporta i dati su un foglio elettronico (spreadsheet)

Figura 4.19: Elenco dei passi eseguiti dalla macro "*Movimenti acquisto conto pieno*"

- **3° passo**: eliminare dalla tabella "*Mov_acq_QMMTR_estr_tb*" memorizzata i valori con il campo *segno='S'*.
- **4° passo**: seleziona da "*Mov_acq_QMMTR_estr_tb*" tutti i valori che sono in join con "*Mov_acq_QMMTR_estr_tb_trova_storni_tb*".
- **5° passo**: effettua un raggruppamento sulla tabella precedente e pone in join con la tabella *QGDLN00* per poter estrapolare le descrizioni relative agli articoli. I dati ottenuti vengono salvati sulla tabella "*Mov_acq_QMMTR_estr_tb_somma_tb*".
- **6° passo**: dalla tabella precedente aggiorna il campo macrofamiglia dell'articolo ponendolo in join con la tabella *QGDIC00* relativa ai parametri commerciali dell'articolo, impostando la validità dei campi delle date di inizio e fine validità: *GDICDTIV<DateValue("31/12/4712")* e *GDICDTFV=DateValue("31/12/4712")*. In questo modo si è sicuri di selezionare il valore correntemente utilizzato e non uno non valido presente nella tabella, in quanto il database gestisce lo storico dei valori.
- **7° passo**: stesso comportamento del passo precedente, ma aggiorna il campo Ragsoc ponendo la tabella in join col la tabella *QPCID00* relativa all'anagrafica dei fornitori.
- **8° passo**: come ultimo passo la tabella "*Mov_acq_QMMTR_estr_tb_somma_tb*" viene salvata su un foglio elettronico denominato "*mm_aa_Acquisto.xls*", eventualmente sovrascrivendolo se già esistente.

Anche in questa query ci sono dei passaggi abbastanza particolari, dove i passi 2, 3, e 4 si possono ricavare con una semplice query innestata (vedi righe del codice SQL riscritto riguardante la tabella *QMMTR00*).

```
select to_char(z.dtmov, 'MM/YYYY') as Data, z.prodotto, c.gdlndesc, sum(z.Qta) as Qta,
       d.gdicmfam as Macrofamiglia, z.clifor, e.ragsoc as Rag_soc
from (
       select a.nrmov, a.partita, a.prodotto, a.dtmov, a.finalfr, a.segno, a.locazfr,
              a.clifor, sum(a.qtmov)/100 as Qta
       from QMMTR00 a
       where not exists (
                          select *
                          from QMMTR00 b
                          where b.segno = 'S'
                          and b.partita = a.partita
                          and b.societa = a.societa
                          and b.prodotto = a.prodotto
                         )
       and a.societa = '01'
       and a.causmov = 'RP'
       and a.operfr = ' '
       and a.locazfr = '001'
       and a.dtmov between :Data_da and :Data_a
       and a.segno <> 'S'
       group by a.nrmov, a.partita, a.prodotto, a.dtmov, a.finalfr, a.segno, a.locazfr, a.clifor
) z
inner join QGDLN00 c on ( c.societa = '01'
                          and z.prodotto = c.gdlnitem
                          and c.gdlnfini = ' '
                          and c.gdlnlang = 'I'
                        )
left join QGDIC00 d on ( d.societa = '01'
                         and d.gdicitem = z.prodotto
                         and d.gdicdtiv < '31-dec-4712'
                         and d.gdicdtfv = '31-dec-4712'
                       )
left join QPCID00 e on ( e.societa = '01'
                         and e.codfor = z.clifor
                         and e.pciddtiv < '31-dec-4712'
                         and e.pciddtfv = '31-dec-4712'
                       )
group by to_char(z.dtmov, 'MM/YYYY'), z.prodotto, c.gdlndesc, d.gdicmfam, z.clifor, e.ragsoc
order by Data, z.prodotto
```

Figura 4.20: Query SQL della macro "*Movimenti acquisto conto pieno*"

4.6.2.2 Macro "Venduto Macrofamiglia"

Questa macro estrae le informazioni operando sui movimenti di magazzino. Le tabelle interessate nelle diverse query sono:

- *QBUST00*: *Tabella del venduto creata per varie esigenze lavorative*;
- *QCAID00*: *Anagrafica cliente*;
- *QGDLN00*: *Descrizioni in lingua dell'articolo*;
- *QGDIC00*: *Parametri commerciali dell'articolo*;
- *QTABD00*: *Descrizioni dei parametri applicativi*;
- *QGDIA00*: *Dati tecnici dell'articolo.*

Ora viene visualizzato in dettaglio come è composta la macro, con il nome delle query presenti all'interno, il tipo ed una breve descrizione del suo operato.

Passo	Azione	Nome query	Query di	Descrizione
1	Apri Query	Venduto_QBUST_estr	Creazione tabella	Estrae il venduto dalla tabella QBUST
2	Apri Query	Venduto_estr_tb_somma	Creazione tabella	Somma per articolo
3	Apri Query	Venduto_estr_tb_somma_agg_ macrofamiglia	Aggiornamento	Aggiorna la macrofamiglia
4	Apri Query	Venduto_estr_tb_somma_tb_so mma_macrofam	Creazione tabella	Somma per macrofamiglia
5	Output Su Excel			Esporta i dati su un foglio elettronico (spreadsheet)

Figura 4.21: Elenco dei passi eseguiti dalla macro "*Venduto Macrofamiglia*"

Descrizione dei singoli passi:

- **1° passo**: la query iniziale parte dalla tabella *QBUST00*, ovvero la tabella che contiene i dati del venduto, selezionando tutti i valori con alcuni filtri sugli articoli. Qui è posta in join la la tabella *QGDLN00*, per poter estrarre le descrizioni degli oggetti, e la tabella *QCAID00*, per poter estrarre i nominativi dei clienti che hanno acquistato i prodotti. In questo passo c'è l'interazione con l'utente per far inserire le date del range di riferimento sul venduto da analizzare. I dati filtrati sono memorizzati nella tabella "*Venduto_ QBUST_ estr _tb*".

- **2° passo**: partendo dalla tabella precedente effettua diversi raggruppamenti (*GROUP BY*) e somme (*SUM*) e salva i risultati ottenuti nella tabella "*Venduto_ estr_ tb_ somma_ tb*".

- **3° passo**: riprende la tabella inizialmente creata, la "*Venduto_ QBUST_ estr_ tb*", ponendola in join con la tabella *QGDIC00* (tabella dei parametri commerciali dell'articolo), a sua volta in join con la *QTABD00* (descrizioni dei parametri applicativi), in modo tale da poter ricavare il valore e la descrizione della MacroFamiglia Commerciale di cui l'oggetto fa parte.

- **4° passo**: effettua diversi raggruppamenti finali e salva tutti i dati nella tabella "*Venduto_ estr_ tb_ somma_ tb_ somma_ macrofam_ tb*",

- **5° passo**: come ultimo passo la tabella creata nel passo 4 viene salvata su un foglio elettronico denominato "*mm_ aa_ Venduto_ per_ macrofamiglia.xls*".

```
select d.gdicmfam as Macrofamiglia, e.tabddesc as Descr_macrofamiglia, sum(Qta) as Tot_qta,
       sum(Importo) as Tot_importo
from (
        select a.item, c.gdlndesc, a.cust, b.caidrags, sum(a.qnty)/100 as Qta,
               sum(a.vacu)/10000 as Importo, concat(concat(a.month,'/'),a.year) as Data
        from QBUSTOO a
        inner join QCAIDOO b on ( b.caidclie = a.cust
                                  and b.caiddtiv < '31-dec-4712'
                                  and b.caiddtfv = '31-dec-4712'
                                )
        inner join QGDLNOO c on ( c.societa = a.societa
                                  and c.gdlnitem = a.item
                                  and c.gdlnfini = ' ' and c.gdlnlang = 'I'
                                )
        where a.societa = '01'
        and a.item not in ('__',' ','              RD')
        and a.acti not in ('R','T','Z')
        and concat(a.year,a.month) between to_char(to_date(:Data_da),'YYYYMM')
                                   and to_char(to_date(:Data_a),'YYYYMM')
        group by  a.item, c.gdlndesc, a.cust, b.caidrags, concat(concat(a.month,'/'), a.year)

) z
left join QGDICOO d on ( d.societa = '01'
                         and d.gdicitem = z.item
                         and d.gdicdtiv < '31-dec-4712'
                         and d.gdicdtfv = '31-dec-4712'
                       )
inner join QTABDOO e on ( e.tabdsoc = ' '
                          and e.tabdcod = 'A80'
                          and e.tabdkey = d.gdicmfam
                          and e.tabdling = 'I'
                        )
left join QGDIAOO f on ( f.societa = '01'
                         and f.gdiaitem = z.item
                         and f.gdiafini = ' '
                         and f.gdiadtiv < '31-dec-4712'
                         and f.gdiadtfv = '31-dec-4712'
                       )
group by d.gdicmfam,e.tabddesc
order by Macrofamiglia
```

Figura 4.22: Query SQL della macro "*Venduto Macrofamiglia*"

4.6.2.3 Lo Schema E/R ricavato

Anche in questo caso abbiamo seguito gli stessi passi effettuati per poter realizzare lo schema della macro "Controllo ordini fatture". Alcune di queste tabelle sono state già analizzate in precedenza, come la *QGDIF00* (e di conseguenza la *QGDIA00* e la *QGDLN00*), la *QTABD00* e la *QPCID00*.

Schema E/R Macro "Movimenti acquisto conto pieno"

QMMTR00: Movimenti di magazzino

PK: Nrmov

FK: Prodotto *REFERENCES* QGDIF00(Gdifitem)

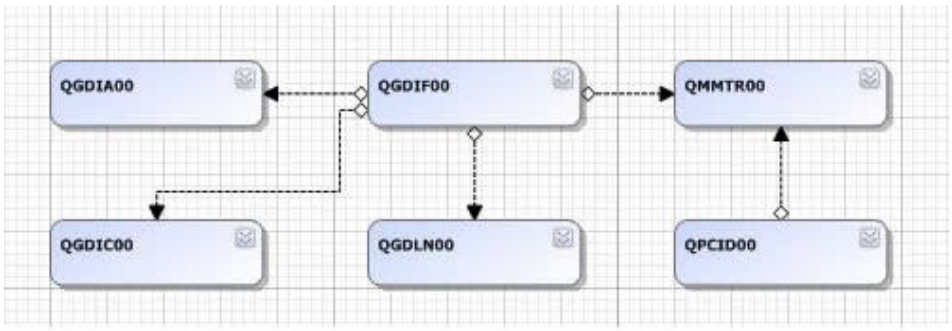

Figura 4.23: Schema generale ricavato dalla macro "*Movimenti acquisto conto pieno*"

FK: Clifor *REFERENCES* QPCID00(Codfor)

QGDIF00: Dati non storicizzati dell'articolo

PK: Gdifitem, Gdiffini

QGDIA00: Dati tecnici dell'articolo

PK: Gdiaitem, Gdiafini, Gdialvrv

FK: Gdiaitem *REFERENCES* QGDIF00(Gdifitem)

FK: Gdiafini *REFERENCES* QGDIF00(Gdiffini)

QGDLN00: Descrizioni in lingua dell'articolo

PK: Gdlnitem, Gdlnfini

FK: Gdlnitem *REFERENCES* QGDIF00(Gdifitem)

FK: Gdlnfini *REFERENCES* QGDIF00(Gdiffini)

QGDIC00: Parametri commerciali dell'articolo

PK: Gdicitem, Gdicdivi, Gdiclvrv

FK: Gdicitem *REFERENCES* QGDIF00(Gdifitem)

QPCID00: Anagrafica fornitori

PK: Codfor, Pcidlvrv

Schema E/R Macro "Venduto Macrofamiglia"

QBUST00: Tabella del venduto creata per varie esigenze lavorative

PK: Year, Month, Cust, Item, Curr

FK: Item *REFERENCES* QGDIF00(Gdifitem)

QCAID00: Anagrafica cliente

PK: Caidclie, Caidlvrv

QGDIC00: Parametri commerciali dell'articolo

PK: Gdicitem, Gdicdivi, Gdiclvrv

FK: Gdicitem *REFERENCES* QGDIF00(Gdifitem)

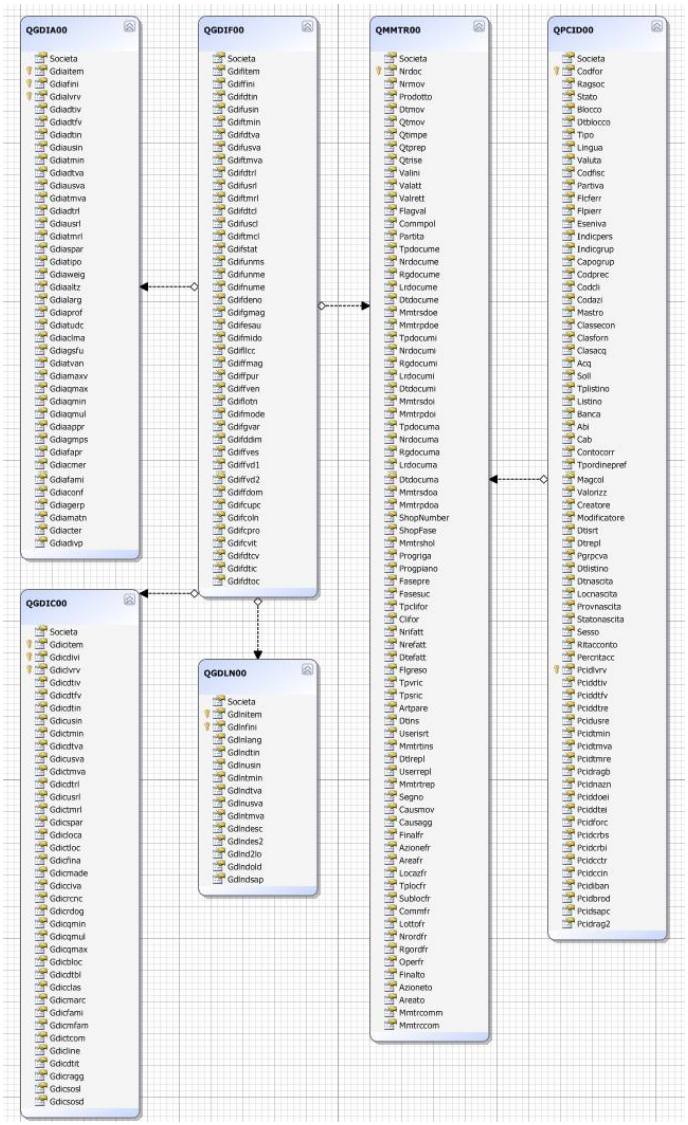

Figura 4.24: Schema dettagliato ricavato dalla macro "*Movimenti acquisto conto pieno*"

QTABD00: Descrizioni dei parametri applicativi

PK: Tabdcod, Tabdkey, Tabdling

FK: Tabcod *REFERENCES* QTABX00(Tabxcodt)

QGDIF00: Dati non storicizzati dell'articolo

PK: Gdifitem, Gdiffini

QGDIA00: Dati tecnici dell'articolo

PK: Gdiaitem, Gdiafini, Gdialvrv

FK: Gdiaitem *REFERENCES* QGDIF00(Gdifitem)

FK: Gdiafini *REFERENCES* QGDIF00(Gdiffini)

QGDLN00: Descrizioni in lingua dell'articolo

PK: Gdlnitem, Gdlnfini

FK: Gdlnitem *REFERENCES* QGDIF00(Gdifitem)

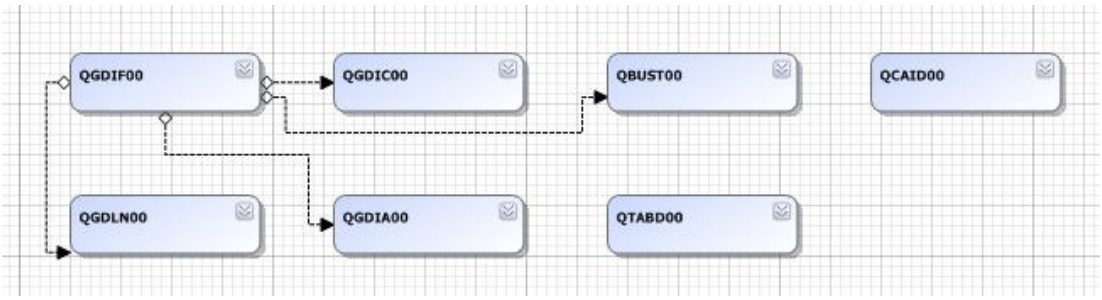

Figura 4.25: Schema generale ricavato dalla macro "*Venduto macrofamiglia*"

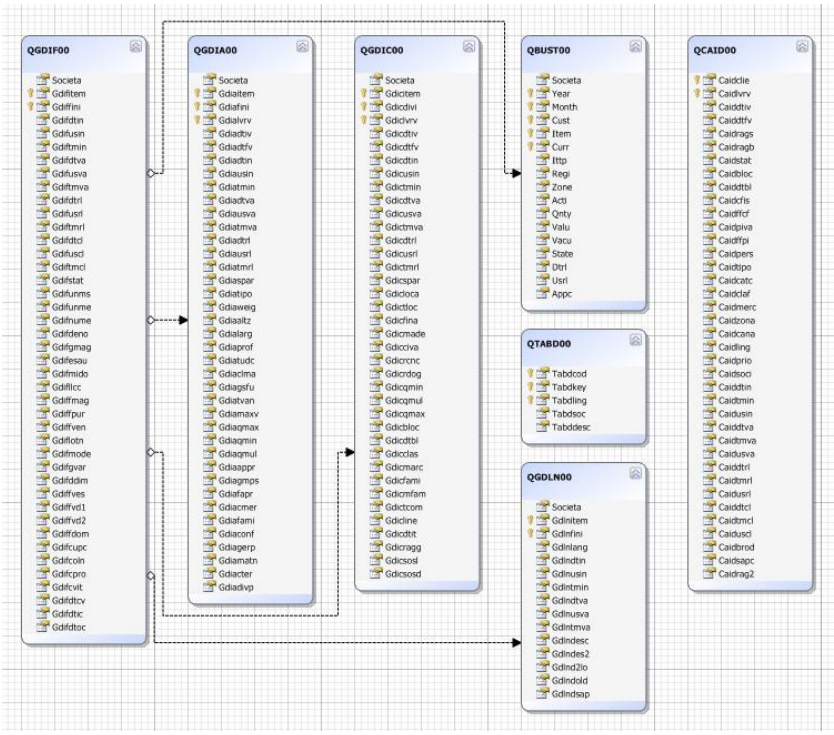

Figura 4.26: Schema dettagliato ricavato dalla macro "*Venduto Macrofamiglia*"

FK: Gdlnfini *REFERENCES* QGDIF00(Gdiffini)

4.7 Il programma per eseguire le query

Fino ad adesso gli utenti potevano eseguire i programmi utilizzando Access e visualizzare i risultati ottenuti, ma con le nuove query sviluppate direttamente in SQL l'approccio diviene diverso. Access può sempre essere utilizzato come interfaccia per poter visualizzare i dati, dove le nuove query possono essere avviate da dentro Access ma con le opportune modifiche sul codice poiché in alcune query sono presenti particolari funzioni di Oracle non riconosciute da Access (ad esempio *decode*, *nvl*, ecc.). Ma utilizzare Access può comportare un elevato rischio di inserimenti/modifiche/cancellazioni sui dati da parte degli utenti e nessun controllo di accesso ai dati a loro non autorizzati. Poiché Access viene utilizzato dalla maggior parte degli utenti quasi esclusivamente per visualizzare informazioni e non per effettuare modifiche, ha senso utilizzare altri programmi, magari realizzati appositamente per le proprie esigenze.
Inoltre non è pensabile distribuire un tool di SQL, come SQL Navigator, a tutti gli utenti dato che non tutti hanno approfondite conoscenze dei principali comandi SQL, ma soprattutto è ancora più facile intervenire sui dati ed effettuare cancellazioni davvero dannose, come l'eliminazione dell'intero database (per effettuare la cancellazione del database basta un semplice comando di *drop*).
Quindi si è deciso di sviluppare un nuovo programma adatto ad eseguire le query, anche quelle parametriche, e che permetta di visualizzare i risultati attraverso un'interfaccia semplice e che protegga dai rischi sopra citati.

4.7.1 Il linguaggio di programmazione C#

Quest'applicazione è stata sviluppata essenzialmente per gli utenti finali e per permettere loro di poter eseguire facilmente le applicazioni attraverso una semplice interfaccia grafica cliccando direttamente sul programma da eseguire, ed eventualmente impostare le date su cui eseguire la ricerca.
Per poter realizzare il programma è stato utilizzato l'ambiente di sviluppo Visual Studio 2008. Da qui, tra i diversi linguaggi di programmazione offerti dallo stesso ambiente, è stato scelto il C# (si pronuncia "c sharp") perché può essere considerato il linguaggio di programmazione per eccellenza del *Framework .NET* diversamente dagli altri linguaggi, come Visual Basic o C++.
È un linguaggio di programmazione object-oriented sviluppato dalla Microsoft all'interno dell'iniziativa .NET, e successivamente approvato come standard ECMA. Da quando è divenuto uno standard ISO (ISO/IEC 23270) si sono sviluppate diverse implementazioni indipendenti di .NET e del C#, fra cui il progetto *Mono* (di Ximian) e *dotGNU* & *Portable.NET* (della Free Software Foundation).

I programmi C# compilati hanno estensione .exe che però non sottintende un programma eseguibile, ma esso rappresenta il risultato di una codifica in un linguaggio intermedio, *IL* (*Intermediate Language*) oppure *MSIL* (*Microsoft Intermediate Language*). Esso non è codice macchina (del processore in dotazione) ma una serie di particolari istruzioni descrittive che consentono la traduzione esatta in codice eseguibile da parte del framework. Quando si lancia un programma C#, entra in gioco il cosiddetto *CLR* (*Common Language Runtime*) che rappresenta un motore di esecuzione ad elevate prestazioni. Il codice cui il runtime si riferisce e la cui esecuzione è gestita da esso viene detto codice gestito (managed code). La responsabilità per attività quali la creazione di oggetti, l'esecuzione di chiamate a metodi e così via, è demandata al Common Language Runtime che consente di fornire servizi aggiuntivi al codice in esecuzione. Nella versione C# 3.0 proposta dalla Microsoft sono implementate nuove caratteristiche ispirate ai linguaggi di programmazione funzionali come *Haskell* e *ML*, nonché l'introduzione del linguaggio *Language Integrated Query* (*LINQ*) appartenente ai linguaggi CLR.

4.7.2 La struttura e le caratteristiche del software realizzato

L'applicazione sviluppata si presenta in due versioni, una con vista utente e l'altra con vista amministrativa: sono essenzialmente uguali ma con la differenza dell'interfaccia grafica e di alcuni particolari interni del codice. Al momento della realizzazione il programma si basava sul .NET Framework 3.5, una delle ultime release rilasciata dalla Microsoft.
Oltre alla selezione della query, il programma esegue la connessione al database, gli invia la stringa SQL, riceve i dati dal database e li visualizza su una griglia di dati (DataGridView). Dalla griglia l'utente può effettuare diversi filtri attraverso un piccolo menu che viene richiamato con il click destro del mouse, dove può effettuare le operazioni di copia dei dati, di filtraggio e di esportazione su file.
Dell'ambiente di studio si sono utilizzate le Windows Forms personalizzate in C# per creare le maschere da visualizzare all'utente (ad esempio la maschera iniziale che permette di scegliere quale query lanciare e la maschera di visualizzazione dei risultati) e file di C# contenete solo codice (ad esempio le classi per effettuare la connessione al database). Windows Forms è il nome dato alla parte di GUI del framework Microsoft .NET.

Il programma, all'apertura della query, analizza i file SQL e ne verifica la presenza dei campi parametrici in modo tale da abilitare/disabilitare i campi di ingresso presenti nell'interfaccia principale. Inoltre effettua controlli sulla presenza e sui valori delle date inserite prima dell'avvio della query.

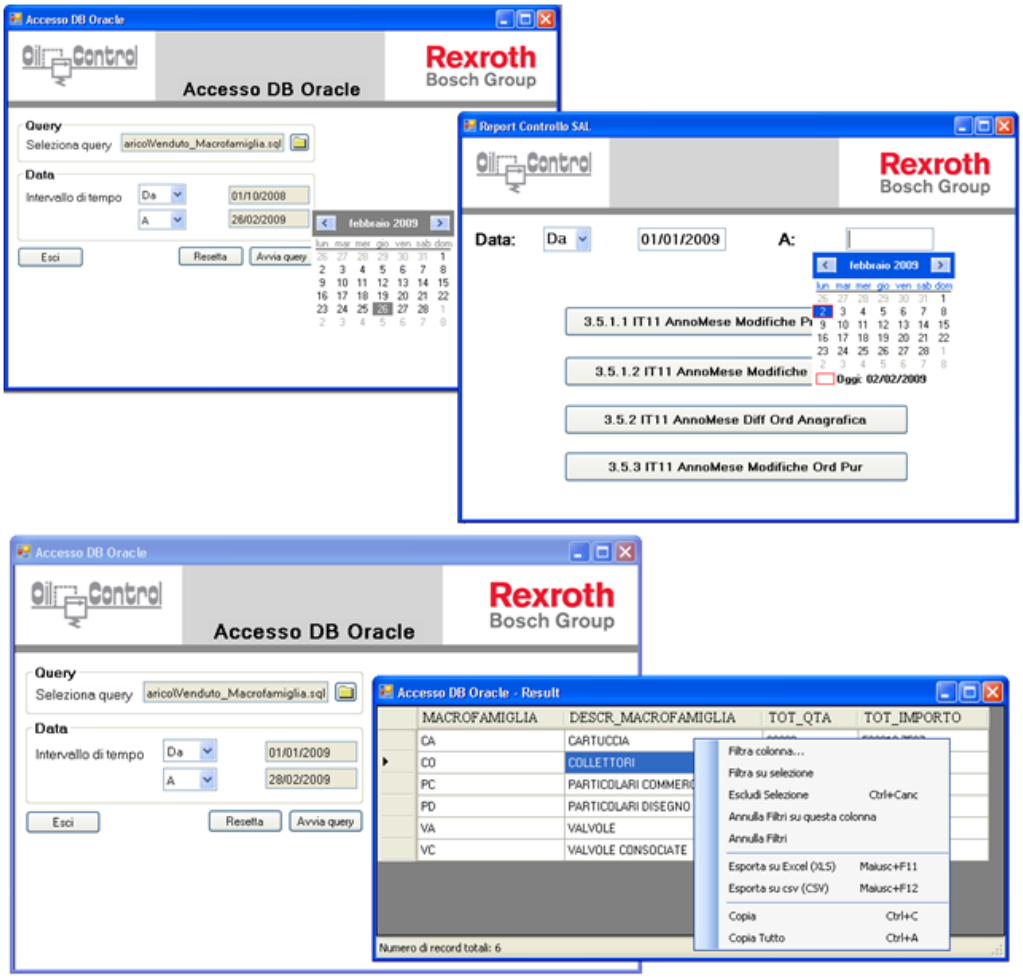

Figura 4.27: Le diverse maschere che compongono il programma realizzato

Le due versioni realizzate sono:

- ***La versione utente***: è realizzata e personalizzata in base all'ufficio destinatario in modo tale da poter semplificare, inoltre per rendere più sicure, le azioni che può effettuate l'utente di un determinato reparto e quali programmi può avviare. Ogni bottone presente nell'interfaccia grafica corrisponde ad un file di SQL presente su un disco remoto in una determinata locazione accessibile solo al reparto ISY. Quindi l'utente non ha molta libertà di accesso ai dati, ma può solo scegliere quale query lanciare, visualizzare i risultati, effettuare degli opportuni filtri e salvare i dati. Inoltre in alcune query è obbligatorio l'inserimento dell'intervallo di tempo su cui effettuare la ricerca.

- ***La versione amministrativa***: è stata realizzata basandosi su quella precedente, ma con la principale differenza che non esiste una lista di query già selezionate, ma si può eseguire qualsiasi tipo di query, anche differenti da quelle trattate in quest'es-

perienza lavorativa, e si può utilizzare il programma per visualizzare i risultati a schermo ed esportarli su file.

La scelta di possibili filtri sulle date possono essere:

- su un determinato range di tempo (Da - A);
- al di sotto di una data (< , <=);
- al di sopra di una data (> , >=);
- uguale ad una data (=);
- senza limiti di tempo (All).

Tale scelta è abilitata solo se nella query è presente almeno un campo parametrico. In tutte le query i parametri li abbiamo definiti come *:Data_ da* e *:Data_ a*.

```
...
and a.item not in ('__',' ','              RD')
and a.acti not in ('R','T','Z')
and concat(a.year,a.month) between to_char(to_date(:Data_da),'YYYYMM')
                           and to_char(to_date(:Data_a),'YYYYMM')
group by  a.item, c.gdlndesc, a.cust, b.caidrags, concat(concat(a.month,'/'), a.year)
...
```

Figura 4.28: Parte del codice SQL della macro "*Venduto macrofamiglia*" riguardante i campi parametrici

La griglia (componente DataGridView) è stata ricostruita utilizzando come guida un articolo tecnico di MSDN (Microsoft Developer Network) della Microsoft denominato ***Building a Drop-Down Filter List for a DataGridView Column Header Cell*** [32], dove è possibile inserire dei filtri sulle testate delle colonne e selezionare un valore attraverso un menu a tendina. È possibile ottenere le celle, righe o colonne selezionate da un controllo DataGridView mediante l'utilizzo delle proprietà corrispondenti: SelectedCells, SelectedRows e SelectedColumns.
Da questa guida è stato molto utile apprendere come è possibile effettuare il filtraggio sulla griglia dei dati in modo tale da poter effettuare il filtro in base alle esigenze. Inoltre è stato implementato un filtro completamente dinamico che contiene lo storico dei filtri inseriti in precedenza, ovvero un filtro appena selezionato non elimina quello precedentemente inserito a meno che non lo si cancelli di proposito.
In basso è stato inserito anche il contatore dei record presenti in tabella, aggiornando il campo ad ogni filtro inserito/eliminato. Il menù a tendina sulle testate elenca tutti i valori distinti presenti nella colonna, ma non è stato preso in considerazione il suo utilizzo poiché dalla prova sul campo ci si è accorti che i dati dei campi, nella maggior parte dei

casi, risultavano diversi tra loro, causando una tendina molto lunga e non utile: quindi non è stata abilitarla sostituendo la sua funzionalità con la maschera *FiltroColonna*. Tale maschera permette di effettuare il filtraggio su un singolo dato con diverse funzionalità (vedi figura 4.29). Inoltre i nomi che compaiono sulle testate sono i nomi effettivi dei campi presenti nella tabella del database.

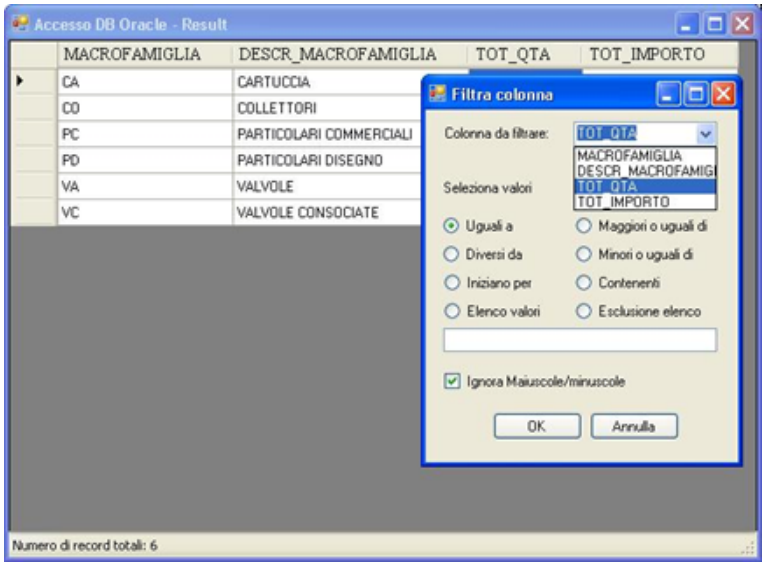

Figura 4.29: Maschera FiltroColonna

Le altre voci presenti nel menu riguardano:

- *Copia e Copia tutto*: copia le celle sezionate o tutte le celle presenti in griglia negli appunti di Windows (clipboard);
- *Filtra colonna*: avvia la maschera che permette di effettuare un filtraggio più avanzato;
- *Filtra su selezione*: seleziona da tutta la griglia i valori che corrispondono al contenuto delle celle selezionate;
- *Escludi selezione*: esattamente l'azione opposta del filtra su selezione;
- *Annulla filtri su questa colonna* e *Annulla filtri*: elimina i filtri impostati in precedenza;
- *Esporta su Excel* (*XLS*) e *Esporta su Excel* (*CSV*): salva i dati su foglio di lavoro nel rispettivo formato selezionato.

Per poter effettuare l'esportazione dei dati su un foglio di lavoro in formato Excel si è utilizzata l'implementazione *Aspose.Cells* della Aspose [33].

Aspose.Cells è un componente aggiuntivo all'ambiente di sviluppo per la creazione di report non grafici in fogli di calcolo Excel, grazie al quale le applicazioni .NET o Java possono leggere e scrivere fogli di calcolo Excel senza utilizzare Microsoft Excel, permettendone un'alta personalizzazione dello stile, colore e posizione delle celle e del testo.

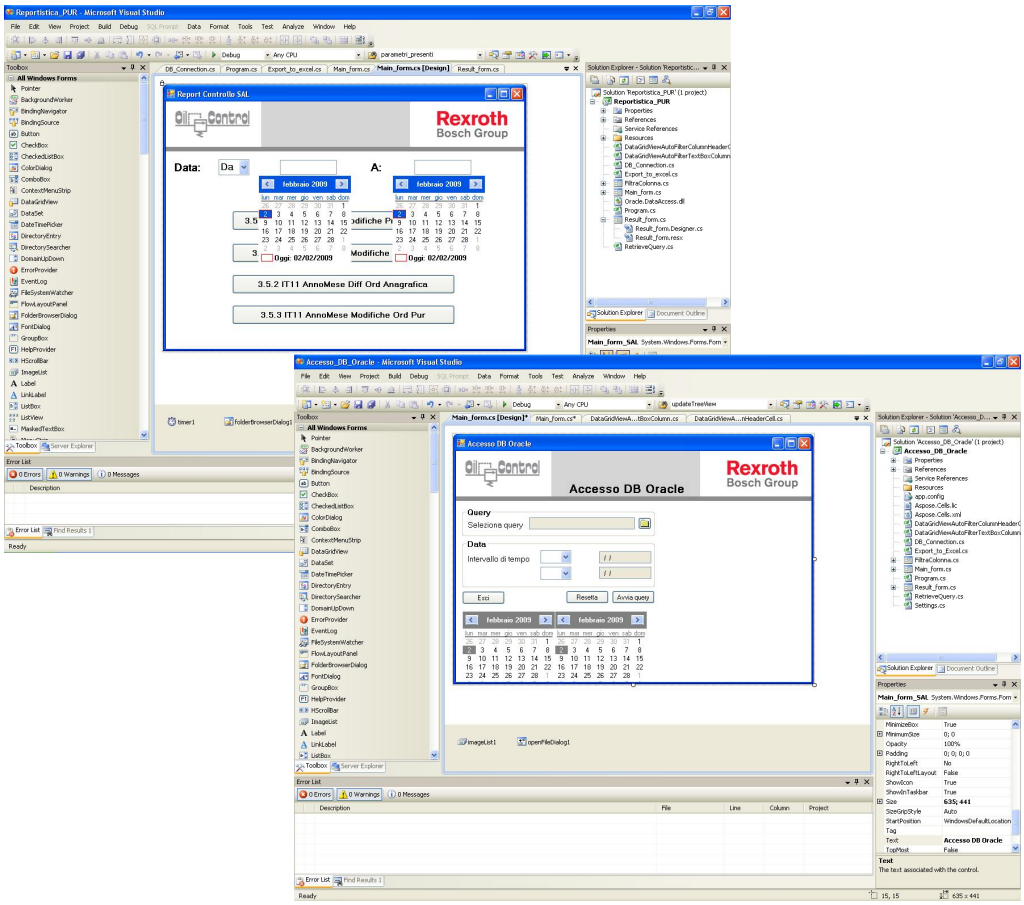

Figura 4.30: Interfaccia principale di Visual Studio del programma utente

La Windows Form, ovvero la finestra dell'applicazione, è il fulcro di ogni applicazione Windows. Attraverso le Windows Forms si può interagire attivamente con l'utente attraverso un'interfaccia grafica dove ci si può rendere subito conto delle operazioni che può effettuare l'applicazione.
I Windows Forms sono altamente personalizzabili attraverso il Windows Forms Designer presente nell'ambiente di sviluppo Visual Studio.
Nel progetto sono state realizzate diverse classi differenziandole in base alle nostre esigenze, dividendo la parte che si interfaccia con l'utente dalla parte che effettua la connessione con il database di Oracle e l'esportazione su foglio elettronico. Le classi presenti nel progetto possono essere visualizzate attraverso il Solution Explorer.

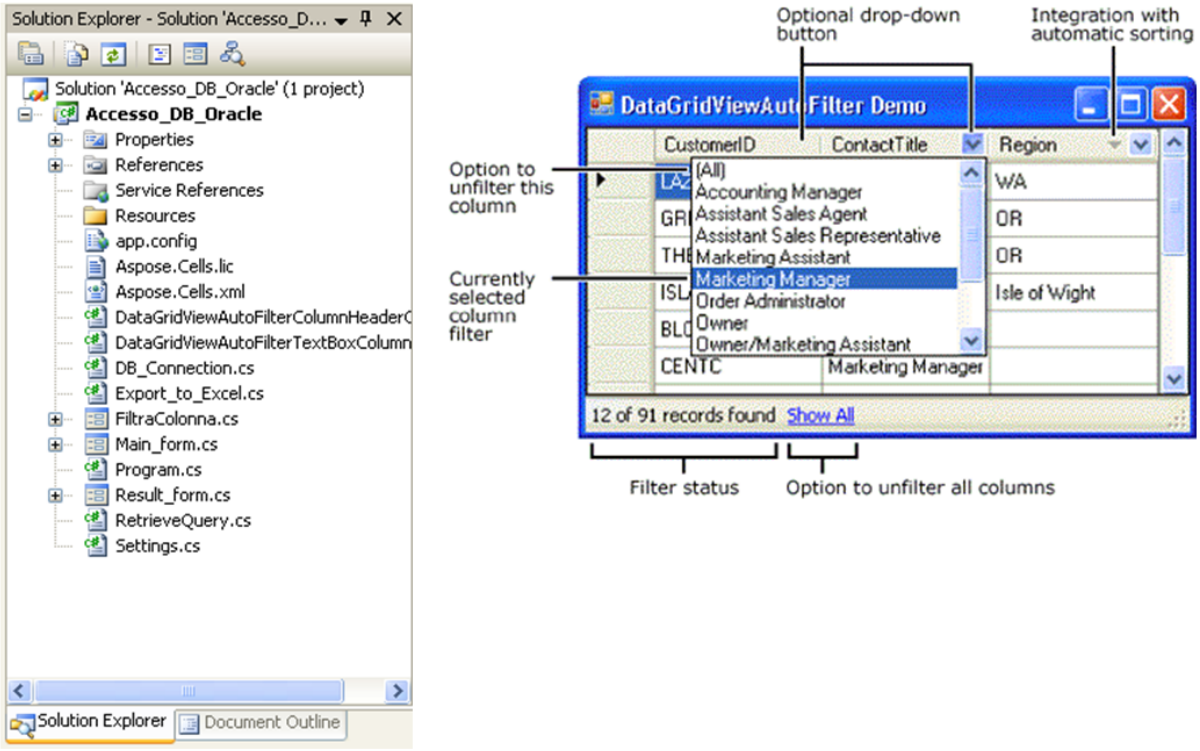

Figura 4.31: Solution Explorer del programma realizzato e la maschera della guida MSDN utilizzata

Dal Solution Explorer (figura 4.31) si può notare la presenza di diverse classi realizzate in C, e da questo si può capire che tipo di componenti sono presenti guardando l'icona assegnata dall'ambiente di studio. I componenti realizzati sono in figura 4.33.
È stato utilizzato il controllo offerto da MSDN poiché il controllo DataGridView di default presente nell'ambiente di studio da solo non include nessuna possibilità di effettuare filtri ed ordinamenti automatici, ma tale controllo è altamente personalizzabile (dimensioni, colori, font, style, ecc.) e si può "espandere" il suo comportamento con altri componenti. Infatti proprio per questo abbiamo utilizzato il componente *DataGridViewAutoFiler* che fornisce le seguenti funzionalità:

- supporta il filtro multi colonna: nella tendina viene visualizzata la lista dei valori distinti presenti in quella colonna;
- supporta l'ordinamento automatico;
- supporta speciali opzioni su filtri: (All), (Blanks) e (NonBlanks).

Adoperando questa caratteristica abbiamo potuto realizzare una griglia contenente i dati e personalizzabile in modo dinamico attraverso alcune funzioni che abbiamo inserito nel menù a tendina abilitato dall'utente, dove si possono impostare/disabilitare filtri in base

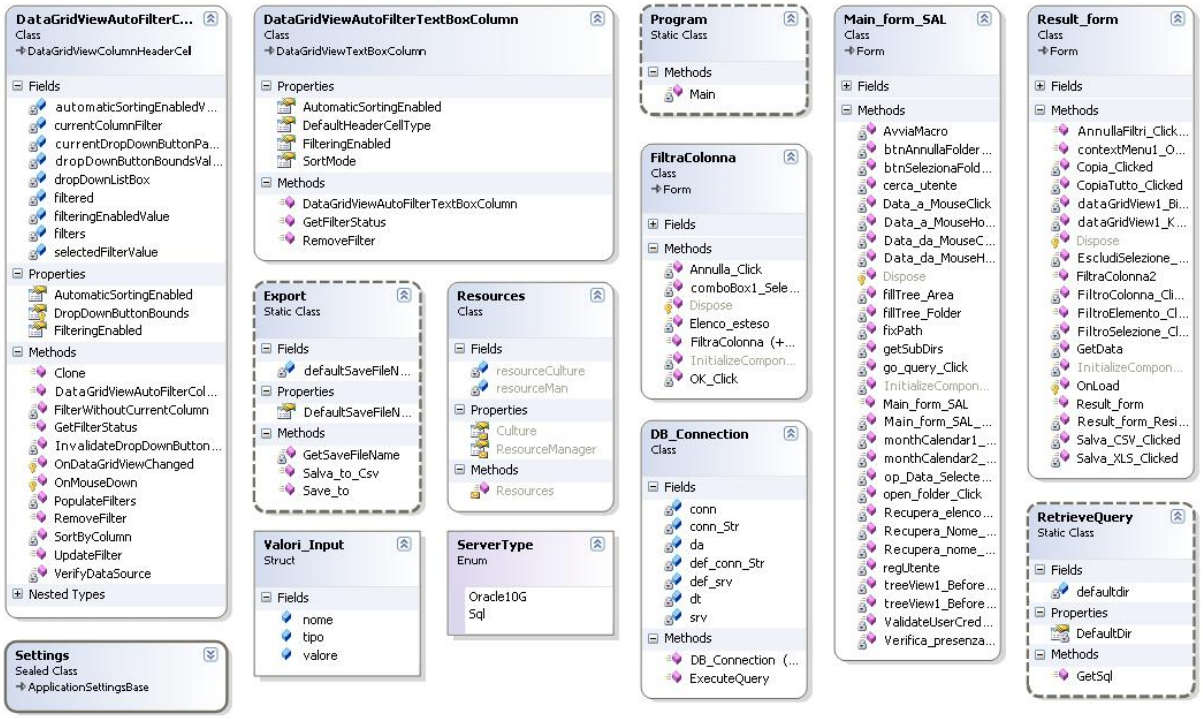

Figura 4.32: Class Diagram del programma C#

alle esigenze. Poiché il filtro a tendina sulla testata non è utile per i nostri scopi, è stato disabilitato.
L'implementazione di tale soluzione richiede come requisiti minimi, oltre all'ambiente di sviluppo Visual Studio, il framework .NET 2.0 e Windows Forms 2.0 (praticamente presenti dalla versione Visual Studio 2005).

4.7.3 Possibili estensioni e problemi riscontrati

Il programma è estremamente personalizzabile e si possono aggiungere altre funzionalità per poter soddisfare particolari esigenze oppure rendere automatiche alcune azioni, come l'esecuzione di alcuni programmi, che sono eseguiti dagli utenti ad ogni particolar evento: ad esempio, ogni fine mese è lanciato il programma "*Giacenza portafoglio ordini*" da un database Access e riguarda la programmazione delle giacenze correnti e delle richieste dei tre mesi a venire; tale programma è stato riscritto interamente in SQL con i metodi visti in precedenza, ed ora può essere eseguito direttamente da uno dei server presenti in azienda in modo del tutto autonomo ed automatico ogni fine del mese, con la possibilità di inoltrare una email di conferma ad avvenuta operazione con in allegato i risultati ottenuti.
Alcune funzionalità che possono essere implementate nel software realizzato sono:

- integrazione tra il sistema operativo Microsoft Windows ed i servizi *LDAP* (*Lightweight Directory Access Protocol*) in modo tale da poter ricavare il profilo, autenticazione

Nome componente	Tipo	Descrizione
Main_form	Windows Form	Interfaccia principale dove l'utente sceglie la query da avviare ed imposta eventuali date
Result_form	Windows Form	Maschera che visualizza i risultati della query
FiltraColonna	Windows Form	Maschera che permette di effettuare eventuali filtri dai risultati ottenuti
DataGridViewAutoFilterColumnHeaderCell	Classe	Permette di personalizzare le testate delle celle e di inserire il menu a tendina contenente i valori distinti presenti nella colonna
DataGridViewAutoFilterTextBoxColumn	Classe	Permette di filtrare la griglia in base ai valori settati
DB_Connecrion	Classe	Classe utilizzata per stabilire la connessione con il database Oracle
RetrieveQuery	Classe	Classe utilizzata per recuperare la stringa SQL dal file impostato nell'interfaccia principale
Export_to_Excel	Classe	Classe utilizzata per salvare i dati su foglio elettronico nei due formati previsti (in XLS e CSV) attraverso il tool Aspose.Cells

Figura 4.33: Descrizione dei componenti presenti nel progetto

e autorizzazione dell'utente che ha effettuato il login sul computer in uso, e di poter rendere il programma adatto alle sue funzioni.

- integrare le query con file *XML* (*eXtensible Markup Language*) in modo da poter rendere la DataGridView ancora più personalizzabile, con la possibilità di poter rinominare le testate di alcuni campi e/o nasconderli se non necessari o non volutamente visibili. Un possibile esempio di struttura XML lo si può vedere nel listato qui sottostante, dove *msdaora* nella stringa di connessione è il *Microsoft Data Access - OLE DB Provider for Oracle*.

Listing 4.1: Possibile documento XML da allegare alla query da eseguire

```
<?xml version="1.0" encoding="utf-8"?>
<Report>
  <Settings>
    <Key Name="Connection" Value="Provider=msdaora; Datasource=Nome Database;User ID=Nome
        User;Password=Password User" />
    <Key Name="SqlQuery" Value="Locazione query con estensione .sql" />
    <Key Name="MailFrom" Value="Indirizzo email mittente" />
    <Key Name="MailTo" Value=" Indirizzo email destinatario" />
    <Key Name="MailObject" Value="Oggetto email" />
    <Key Name="MailText" Value="Testo email" />
    <Key Name="AttachmentName" Value="Locazione allegato con estensione (esempio in .xls)
        " />
  </Settings>
  <Fields>
    <Key id="Nome colonna" Name="Denominazione nome colonna" Value="Tipo dato e
        dimensione" />
    <Key id="Nome colonna" Name="Denominazione nome colonna" Value="Tipo dato e
        dimensione" />
  </Fields>
</Report>
```

Il programma realizzato è estremamente personalizzabile con la possibilità di poter aggiungere nuove funzionalità, ma bisogna risolvere alcuni problemi, come ad esempio l'esecuzione da remoto. Infatti l'utente che vuole eseguire il programma da remoto (oltre ai requisiti da rispettare sul PC locale per eseguire il programma) deve rispettare altri requisiti di sicurezza legati l'esecuzione di programmi.
Uno dei principali requisiti software dei PC locali utilizzati riguarda la presenza del framework .NET, almeno della versione 2. Infatti il progetto realizzato è stato effettuato basandosi sul framework .NET 3.5 (anche se nel momento della realizzazione l'ultima versione era il framework 3.5 SP1), ma le classi utilizzate sono compatibili fino alla versione 2.
Il problema riscontrato sulla gestione della sicurezza (*Security Policy*) implementata nel framework 2 riguarda l'esecuzione di codice da cartella remota. Ricordiamo che la versione 3 è un "add-on" basata sulla 2 e con tale versione non è possibile eseguire i programmi in mancanza di una policy definita appositamente. Tale metodo è stato implementato per evitare la propagazione sulla rete di virus. Però si è riscontrato che sui PC che presentano installata la versione 3.5 SP1 non si presenta nessun problema di accesso e il programma può essere eseguito come se fosse posizionato in locale, mentre ciò non avviene con versioni precedenti del framework. Quindi per l'esecuzione su PC provvisti di versioni antecedenti la 3.5 SP1 si devono adottate tecniche di *Trusted software integrity* denominato *CAS* (*Code Access Security*), dove sulle macchine interessate devono essere installati i certificati.

Capitolo 5

LINQ to Oracle

Dalle applicazioni sviluppate e viste nel capitolo precedente si possono implementare nuove soluzioni, come un nuovo metodo di accesso alle informazioni senza avere una conoscenza approfondita di come sono strutturati fisicamente i dati oppure dare la possibilità di poter eseguire le applicazioni tramite browser web. Questo è solo un piccolo esempio di idee e soluzioni che possono essere realizzate in quanto dall'insieme delle tecnologie oggi offerte dal mondo dell'informatica si potrebbe pensare a realizzare una situazione interamente web based, ma chiedere questo oggi forse è un po' troppo data l'azienda e la bassa priorità destinata alla realizzazione di questo tipo di applicazioni. Inoltre è da considerare anche la bassa priorità, ma anche libertà, che viene lasciata dalla casa madre Bosch per poter realizzare ciò, in quanto si tiene la linea direttiva per poter avere una standardizzazione di tutta l'infrastruttura software presente in tutte le divisioni.
Quindi quello che oggi manca non è la tecnologia per poter creare e realizzare nuove soluzioni, ma è la vera necessità nel svilupparli ed adattarli alle specifiche esigenze aziendali ed alla situazione reale che si vive ogni giorno in azienda, ricordando che la DCOC è principalmente un'azienda dove l'informatica deve essere di supporto alla linea produttiva.
Nonostante ciò in questo capitolo si vuole trattare in parte un nuovo metodo di accesso ai dati realizzato dalla Microsoft, denominato ***LINQ*** (***Language-Integrated Query***). Tale argomento non è stato trattato direttamente in azienda ma è una propensione personale che si vuole dare guardando al futuro il lavoro appena concluso. Quindi questo capitolo tratta argomenti non richiesti dall'azienda, ma che possono essere utilizzati per poi poterli realizzare all'effettiva necessità.
Si cercherà di dare un'introduzione generale di tale tecnologia e come può essere praticamente utilizzata nelle applicazioni viste in precedenza.

5.1 Il Domain Model

La maggior parte delle applicazioni aziendali viene attualmente scritta per l'accesso ai dati di database relazionali. A un certo punto, queste applicazioni dovranno interagire con i dati rappresentati in forma relazionale. Il modello relazionale è ottimizzato per garantire efficienza di archiviazione e recupero, non per la modellazione concettuale utilizzata nella programmazione orientata a oggetti. Più tabelle normalizzate corrispondono spesso a una singola classe e le relazioni tra le classi non sono rappresentate nello stesso modo delle relazioni tra le tabelle. Gli sviluppatori di applicazioni aziendali devono spesso utilizzare due o più linguaggi di programmazione: un linguaggio di alto livello per i livelli di presentazione e della logica di business (ad esempio Visual C# o Visual Basic) e un linguaggio di query per interagire con il database (ad esempio Transact-SQL). Tale necessità non solo richiede una conoscenza approfondita di diversi linguaggi da parte degli sviluppatori, ma provoca anche problemi di mancata corrispondenza tra linguaggi nell'ambiente di sviluppo. In un'applicazione che utilizza un'API di accesso ai dati per eseguire una query su un database, ad esempio, la query viene specificata come valore letterale stringa racchiuso tra virgolette. Questa stringa di query è tuttavia opaca per il compilatore e non è possibile eseguire su di essa un controllo per verificare la presenza di errori, ad esempio l'utilizzo di sintassi non valida o l'effettiva esistenza delle colonne o delle righe cui fa riferimento. Non viene eseguito il controllo dei tipi dei parametri di query, né è disponibile il supporto per IntelliSense.
Ormai è consuetudine (ed auspicio) quella di scrivere codice attinente a pattern architetturali ben consolidati e opportunamente inseriti nel disegno dell'architettura del software che si sta implementando. Le applicazioni *data-driven*, cioè quelle che basano gran parte della logica sulla manipolazione di informazioni memorizzate su un database, sono senz'altro tra le più diffuse. Un approccio sempre più utilizzato consiste nel modellare i dati sfruttando i vantaggi della programmazione Object oriented, secondo il pattern comunemente denominato *Domain Model*. Questo è uno dei pattern architetturali più "in voga" ai giorni d'oggi, che parte dal principio di fondo secondo il quale si vuole avere una astrazione quanto più alta possibile della parte di persistenza dei dati. Ciò al fine di poter lavorare con oggetti piuttosto che con entità vere e proprie e, soprattutto, di poter, laddove possibile, non cablare nel codice logica relativa ad un tipo di database piuttosto che ad un altro (SQL, piuttosto che Oracle, piuttosto che MySQL, ecc.). [36]
Tramite l'utilizzo di questo pattern, infatti, come si evince dallo stesso diagramma riportato in figura 5.1, si tende a rappresentare l'insieme dei dati mediante una modellazione degli stessi in oggetti, così come dettano i principi della Object oriented. Ciò, molto a grandi linee, avviene facendo in modo di far corrispondere ad ogni dato presente nello specifico database un'istanza di classe, e ad ogni relazione tra le tabelle dello stesso, uno

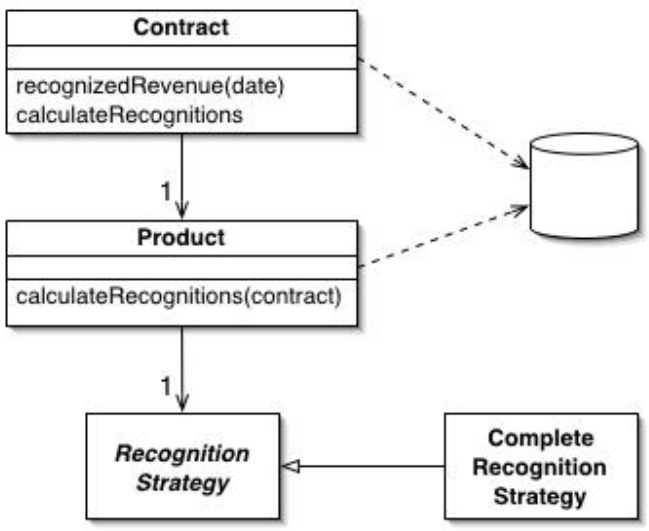

Figura 5.1: Domain Model Pattern

specifico riferimento.
Questo, ovviamente, può valere limitatamente ad applicazioni non eccessivamente complesse, dal lato della persistenza almeno, perché in database con tante tabelle e tante relazioni tra di esse l'andare a scrivere le classi e a gestire i vari riferimenti tra le stesse può risultare impresa non banale.
Questo risulta il vero principale limite dell'adozione di un pattern di questo tipo, ma limitazione che viene quasi interamente superata mediante l'adozione di un ***ORM*** (***Object Relational Mapping***) cioè di una tecnica che fornisce un mapping tra oggetti e tabelle di un database relazionale, e che provvede a mettere a disposizione del programmatore tutta una serie di strumenti per la gestione delle operazioni fondamentali relative alla persistenza e alla manutenibilità dei dati del database (il salvataggio, la lettura, la cancellazione, ecc.).
L'ORM è quindi in grado di fornire delle funzionalità ad un livello di astrazione abbastanza elevato da poter permettere allo sviluppatore di lavorare con oggetti che fisicamente mappano record di una o più tabelle diverse nel database togliendo, tra le altre cose, in tal modo, il pensiero di andare a scrivere query in join particolarmente complesse. Inoltre un buon ORM è in grado di attivare tutti i meccanismi di *rollback* delle transizioni, cancellazioni di record di tipo *cascade* e lettura dei dati di tipo *lazy*, cioè la possibilità di leggere dal database solo i dati che servono di un oggetto piuttosto che andare a fare query molto pesanti per estrapolare tutte le informazioni relative allo stesso.

La gestione della persistenza e del fetch di un grafo di oggetti, a prima vista operazione assolutamente banale, nasconde in realtà una serie di insidie che sfociano spesso in com-

plicazioni difficili da superare. In figura 5.2 si può vedere un semplice modello di dominio di un ipotetico gestionale, e da sola rende l'idea di quanto possa essere articolato un grafo di oggetti e come possano essere di conseguenza complesse (oltre che noiose e ripetitive) le operazioni di memorizzazione e lettura dei dati da database. Quello in figura è solo un piccolo dominio composto da diverse tabelle, mentre sul sistema gestionale ERP Apache se ne possono contare molte di più.

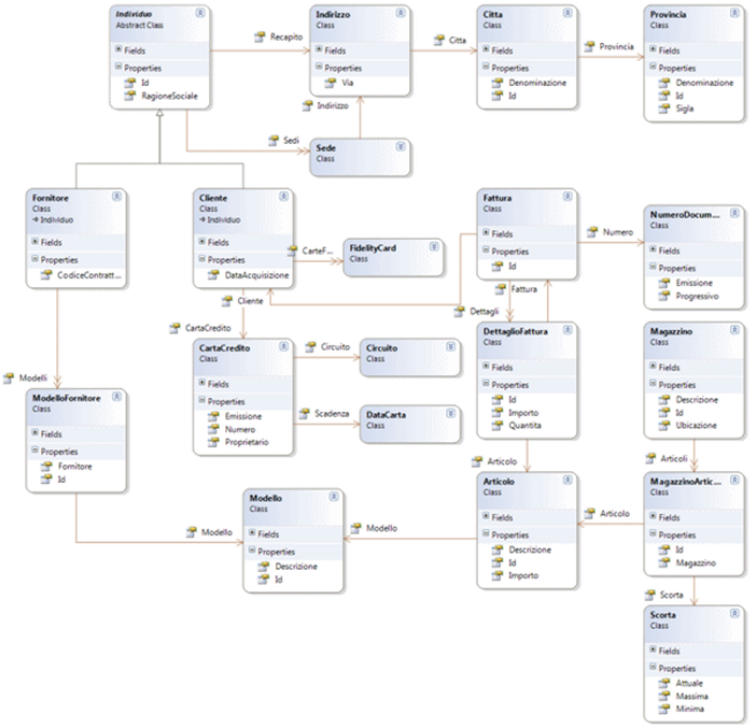

Figura 5.2: Esempio di un grafo delle dipendenze di un Domain Model

Immaginiamo di dover leggere una fattura da database o di doverne memorizzare una. Operazioni di questo tipo, di per sé estremamente comuni, pongono già i primi dilemmi su quali siano le query da effettuare: dal momento che il grafo è allo stesso tempo complesso e notevolmente interconnesso, quali dati devono essere letti e quali possono essere invece trascurati? A cosa si deve far puntare le reference rimaste vuote? Riguardo l'operazione di memorizzazione di una fattura, quali entità (e quindi quali tabelle) vengono coinvolte? Si potrebbe continuare a lungo con questi interrogativi, e questo perché il compito di gestire la persistenza di un domain model di questo tipo (tra l'altro volutamente molto semplice) è effettivamente molto complesso.
Ecco perché si ha bisogno di utilizzare strumenti che permettono di gestire domain model complessi, inoltre se si vuole interagire con il database attraverso linguaggi di programmazione object oriented si ha necessariamente bisogno di un ORM, strumento che collega due mondi completamente diversi.

5.2 ORM (Object/Relational Mapping)

L'***ORM*** (***Object/Relational Mapping***) è un framework che si propone di risolvere le problematiche per convertire dati fra RDBMS e linguaggi di programmazione orientati agli oggetti, sostanzialmente generando in maniera intelligente, mirata e soprattutto automatica, query SQL a partire da istruzioni di alto livello impartite dallo sviluppatore.
In buona sostanza, associa a ogni operazione e elemento usato nella gestione del database degli oggetti con adeguate proprietà e metodi, astraendo l'utilizzo del database dal DBMS specifico.

I principali vantaggi nell'uso di questo sistema sono i seguenti:

- il superamento (più o meno completo) dell'incompatibilità di fondo tra il progetto orientato agli oggetti ed il modello relazionale sul quale è basata la maggior parte degli attuali DBMS utilizzati;
- un'elevata portabilità rispetto alla tecnologia DBMS utilizzata: cambiando il DBMS non devono essere riscritte le routine che implementano lo strato di persistenza e generalmente basta cambiare poche righe nella configurazione del prodotto per l'ORM utilizzato;
- facilità d'uso, poiché non si è tenuti ad utilizzare direttamente un linguaggio di query come l'SQL (che comunque rimane disponibile per eseguire compiti complessi o che richiedono un'elevata efficienza).

I prodotti ORM attualmente più diffusi offrono spesso nativamente funzionalità che altrimenti andrebbero realizzate manualmente dal programmatore:

- caricamento automatico del grafo degli oggetti secondo i legami di associazione definiti a livello di linguaggio;
- gestione della concorrenza nell'accesso ai dati durante conversazioni;
- meccanismi di caching dei dati (con conseguente aumento delle prestazioni dell'applicazione e riduzione del carico sul sistema RDBMS).

Esistono diverse soluzioni ORM, in base al linguaggio Object oriented utilizzato, come C++, Java e .NET, oppure anche altri linguaggi come Delphi, PHP e Ruby. Tra tutti una nota di rilievo ha sicuramente ***Hibernate*** (o ***NHibernate*** nella versione per .NET) nato come progetto open source per Java e utilizzato con profitto in un gran numero di applicazioni di livello enterprise diventando ormai il punto di riferimento di questo settore.

Una possibile alternativa è quella introdotta dalla Microsoft con ***LINQ*** incluso nel nuovo framework (versione 3.5). Questo è un ORM nato con lo scopo di tradurre le query integrate al linguaggio scelto (quindi query di tipo LINQ) in query SQL per l'esecuzione sulla base dati e, in seguito, di tradurre il risultato della query in strutture dati tabulari costituenti delle istanze di oggetti vere e proprie.

5.3 LINQ (Language-Integrated Query)

LINQ (si pronuncia "link") [34] è l'acronimo di ***Language-Integrated Query*** e rappresenta il primo framework Microsoft per l'accesso ai dati, indipendente dall'architettura e dalle strutture cui si tenta di accedere e totalmente integrato all'interno dei linguaggi .NET di alto livello. È un insieme di funzionalità del .NET Framework 3.5 ed estende le potenzialità di esecuzione delle query alla sintassi dei linguaggi .NET (C# e Visual Basic) per avere la possibilità di effettuare interrogazioni su oggetti utilizzando una sintassi simile ad SQL.
Con LINQ si possono eseguire query e manipolare dati sfruttando un modello indipendente dalle varie tipologie di fonti; si può infatti accedere a database, file di testo, file XML, array, file Excel, file di configurazione, informazioni su assembly, chiavi di registro e qualsiasi altro oggetto riconducibile ad una collezione di oggetti enumerabile; il tutto utilizzando un unico modello di programmazione che riunisce molteplici tecniche differenti di accesso ai dati.
In genere, le query sui dati vengono espresse come stringhe semplici senza il controllo dei tipi in fase di compilazione o il supporto IntelliSense. Inoltre è necessario conoscere un linguaggio di query diverso per ogni tipo di origine dati: database SQL, documenti XML, diversi servizi Web e così via. LINQ rende una query un costrutto di linguaggio di prima categoria in C# e Visual Basic. È possibile scrivere query su insiemi di oggetti fortemente tipizzati utilizzando le parole chiave del linguaggio e gli operatori comuni.
LINQ definisce un insieme di operatori che possono essere usati per interrogare, proiettare e filtrare dati in matrici, classi enumerabili, XML, database relazionali e sorgenti dati di terze parti.
Consente l'interrogazione di ogni sorgente di dati che rappresenti i dati sotto forma di oggetti. Per questa ragione, se la sorgente non memorizza in maniera nativa i dati come oggetti, è necessario l'utilizzo di un connettore per accedere ai dati.
Il risultato di una query viene restituito come una collezione di oggetti in memoria che possono essere enumerati.
L'ambiente di sviluppo Visual Studio 2008 include l'assembly del provider LINQ che consente di utilizzare LINQ con insiemi di .NET Framework, database di SQL Server, DataSet ADO.NET e documenti XML.

Scrivere una query per cercare all'interno di un grafo di oggetti è sicuramente più semplice che ciclare gli elementi uno ad uno, alla ricerca di quelli che rispettino tali caratteristiche (LINQ to Objects).
Scrivere una query con LINQ, piuttosto che utilizzare XPath, per ritrovare dati in formato XML, è molto più semplice e veloce dal punto di vista dello sviluppo e della leggibilità del codice (LINQ to XML).
Ma il vero punto di forza di LINQ è nel colloquio con il database. Infatti, i dati contenuti in un database relazionale hanno una rappresentazione che è molto diversa da quella che poi hanno in un modello ad oggetti di un'applicazione. Ad esempio, una classica relazione molti a molti tra *prodotti* e *fornitori* (dove un fornitore offre più prodotti e un prodotto è venduto da più fornitori) viene rappresentata in un database con una tabella *Fornitori*, una *Prodotti* ed una *ProdottiPerFornitore* che contiene le relazioni. Gli stessi dati vengono rappresentati in maniera molto diversa quando si tratta di utilizzare un dominio di oggetti. Infatti, per lo stesso problema si usano due sole classi e non tre: una classe *Fornitori* con una proprietà che contiene la lista dei prodotti offerti, ed una classe *Prodotti* che contiene la lista dei fornitori che li vendono.
Questa diversità viene generalmente gestita dallo strato di accesso ai dati che serve proprio per disaccoppiare l'applicazione dal database. Costruire un *DataLayer* che mappi dal database alle classi è un'operazione che richiede tempo e che comunque prevede la scrittura di istruzioni SQL che estraggono dati, e di codice per portare questi dati dal formato relazionale al formato ad oggetti come detto in precedenza. Questa operazione può essere fatta realizzando un DataLayer proprietario per l'applicazione, oppure rivolgendosi ad un ORM che, sfruttando opportune configurazioni, è in grado di generare le istruzioni SQL ed il codice necessario ad eseguire le query e popolare gli oggetti.
Tutto questo però ha un costo in termini di performance, di compile-time checking e quindi di velocità di sviluppo. Infatti, le query SQL scritte nel codice o generate automaticamente da un ORM non sono altro che stringhe, quindi possono contenere anche istruzioni errate e l'unico modo per scoprirlo è lanciare l'applicazione. Questo significa che in fase di compilazione non si possono identificare eventuali errori con conseguente perdita di tempo. Inoltre, il dover eseguire le query e popolare degli oggetti può risultare un'operazione lenta, sia in termini di sviluppo che di esecuzione (LINQ to SQL).

5.3.1 Panoramica sulle varie versioni

LINQ è costituito da una serie di estensioni sintattiche per i linguaggi gestiti e permette di eseguire interrogazioni su molteplici tipologie di sorgenti dati, costituendo così cinque implementazioni che consentono l'accesso ai dati:

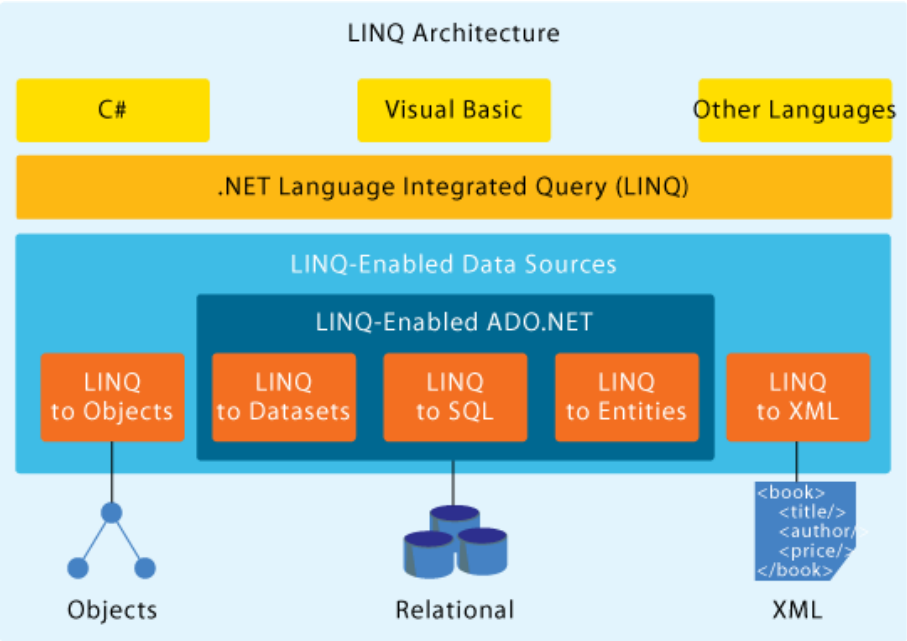

Figura 5.3: Panoramica dell'architettura LINQ

- ***LINQ to Objects***: permette di eseguire delle query su collezioni di oggetti in memoria. Ciò consente di utilizzare i dati in memoria nello stesso modo in cui si utilizzano dati da qualsiasi altra origine.
- ***LINQ to DataSet***: permette di eseguire complesse funzionalità di query sia sui DataSet normali che su quelli tipizzati. Ciò consente di creare ed eseguire query su join tra DataTable in un DataSet.
- ***LINQ to SQL***: supporta il rapido sviluppo di applicazioni che inviano query utilizzando oggetti presenti a livello di programmazione associabili direttamente agli oggetti di database negli schemi di SQL Server, ad esempio tabelle, viste, stored procedure e funzioni definite dall'utente.
- ***LINQ to Entities***: supporta un mapping più flessibile di oggetti a tabelle relazionali, viste, stored procedure e funzioni definite dall'utente. È possibile utilizzare LINQ to Entities per accedere ai dati di SQL Server e di altri database relazionali attraverso il provider ADO.NET.
- ***LINQ to XML***: è un'*API* (*Application Programming Interface*) di programmazione XML in memoria progettata per approfittare delle più recenti innovazioni del linguaggio .NET Framework.

Dalle versioni elencate si cercherà di approfondire ***LINQ to SQL*** e ***LINQ to Entities***.

5.3.2 La sintassi

La sintassi LINQ è molto simile a quella utilizzata per accedere ad una qualsiasi database, ma è importante specificare che LINQ non è solo un tool per utilizzare SQL.
In generale esistono in LINQ due tipi di sintassi che si possono utilizzare: la *Query syntax* e la *Method syntax*. La principale differenza tra le due sintassi risulta nella semplicità e maggior leggibilità della query syntax rispetto alla method syntax, ma non vi è nessuna differenza semantica tra le due sintassi.
Inoltre il .NET Common Language Runtime (CLR) non utilizza la query syntax e pertanto, in fase di compilazione, le espressioni vengono convertite in chiamate al metodo in modo da poter essere utilizzate da CLR. Questi metodi sono denominati operatori di query standard e includono *Where*, *SelectGroupBy*, *JoinMax*, *Average* e così via.

Proprio per permettere la scrittura di query LINQ sono state introdotte nei linguaggi C# e VB.NET (Visual Basic.NET) queste nuove keyword:

- *from*: è la keyword di inizio di ogni query LINQ e specifica la fonte di dati nella quale dovrà essere eseguita la query.
- *where*: è la clausola che specifica quali elementi saranno ritornati dalla query; applica una sorta di filtro di selezione.
- *select*: è la clausola che definisce i tipi di valori che saranno prodotti dalla query.
- *group* (Group By in VB.NET): è la clausola che raggruppa i risultati secondo una certa chiave.
- *orderby* (Order By in VB.NET): effettua l'ordinamento (ascendente o discendente).
- *join*: permette di effettuare prodotti cartesiani tra più fonti di dati, come in SQL. Anche qui possiamo definire inner join o outer join.
- *into* (valida solo per C#): è la keyword contestuale che indica in quale variabile temporanea vengono salvati i risultati di una select, di un group o di un join.
- *let* (valida solo per C#): è la keyword che permette di salvare temporaneamente il risultato di una subquery per poi utilizzarlo all'interno della query principale.
- *Take* (valida solo per VB.NET:): clausola che ritorna il numero specifico dei numeri contigui dall'inizio di una collezione.
- *Distinct*: clausola che restringe il numero dei valori ritornati da una query eliminando i duplicati (valida solo per VB.NET ma in C# si può usare il metodo *Distinct()*).

Grazie alle modifiche sui linguaggi del .NET Framework e alle aggiunte funzionali come le variabili implicite si può utilizzare LINQ per effettuare rapidamente operazioni, su collezioni di oggetti, che prima avrebbero richiesto molte righe di codice, cicli e condizioni di controllo.

5.3.3 Esempio #1

Qui viene posto un esempio pratico di come differiscono le due sintassi realizzato in C#. [35]

Listing 5.1: Differenza tra Query syntax e Method syntax

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;

namespace EsempioLINQ
{
   class QueryVMethodSyntax
   {
       static void Main()
       {
           int[] numeri = { 5, 10, 8, 3, 6, 12};

           //Query syntax:
           IEnumerable<int> numQuery1 =
               from num in numeri
               where num % 2 == 0
               orderby num
               select num;

           //Method syntax:
           IEnumerable<int> numQuery2 = numeri.Where(num => num % 2 == 0).OrderBy(n => n)
               ;

           foreach (int i in numQuery1)
           {
               Console.Write(i + " ");
           }
           Console.WriteLine(System.Environment.NewLine);

           foreach (int i in numQuery2)
           {
               Console.Write(i + " ");
           }

           // Lascia aperta la console in debug mode.
           Console.WriteLine(System.Environment.NewLine);
           Console.WriteLine("Press any key to exit");
           Console.ReadKey();
       }
    }
}
```

Lo scopo del programma è abbastanza semplice, ovvero individuare quali numeri sono pari, metterli in ordine crescente e visualizzarli sullo schermo. Come si può vedere dal listato, la method syntax è più compatta ma a discapito della comprensione. Da notare che nell'esempio proposto si ha una fonte di dati rappresentata da un array di interi predefinito, una variabile di tipo IEnumerable che rappresenta il risultato della query di selezione e la query vera e propria che seleziona i numeri pari (utilizzando la variabile *num* come valore generico della collezione di oggetti su cui effettuare il controllo tramite la clausola where). Da notare che per poter utilizzare la sintassi LINQ bisogna aggiungere una direttiva *using* o un'istruzione *Imports* per *System.Linq*.

5.3.4 Esempio #2

Qui viene posto un altro esempio, ma questa volta realizzato sia in C# che in VB.NET dove si selezionano dalla tabella dei clienti quelli che risiedono a Milano. Ricordiamo che la query verrà poi eseguita all'interno del database secondo questa sintassi SQL aumentando notevolmente la facilità di scrittura del codice per l'accesso ai dati relazionali all'interno delle proprie applicazioni (considerando che non ci si deve più preoccupare di creare una connessione al database, lanciare un comando, ecc.).

Listing 5.2: Esempio in C#

```
ORMDataContext context = new ORMDataContext();

var customers = from c in context.Customers
                where c.City == "Milano"
                orderby c.ContactName
                select c;
```

Listing 5.3: Esempio in VB.NET

```
Dim context As New ORMDataContext()

Dim result = From c In context.Customers _
             Where c.City = " Milano " _
             Select c
```

Listing 5.4: Query effettuata sul database

```
SELECT [CustomerID],[CompanyName],[ContactName],[ContactTitle],[Address],
       [City],[Region],[PostalCode],[Country],[Phone],[Fax]
FROM [Northwind].[dbo].[Customers]
WHERE [City] = 'Milano'
```

Se invece si vuole inserire un nuovo record all'interno del database non bisogna fare altro che creare l'informazione da inserire utilizzando le strutture definite dal mapping LINQ to SQL e sottomettere il nuovo dato all'oggetto che si occupa di gestire le comunicazioni tra l'applicazione ed il database.

Gli esempi di aggiornamento del database e di cancellazione di record sono stati omessi, ma comunque non si discostano di molto dagli ultimi due casi esaminati.

Listing 5.5: Esempio di inserimento in C#

```
ORMDataContext context = new ORMDataContext();

Customer c = new Customer();
c.CustomerID = "ID180";
c.City = "Roma";
c.ContactName = "Massimo Decimo Meridio";
c.Address = "Via Appia";
c.CompanyName = "colosseo.com";

context.Customers.InsertOnSubmit(c);
context.SubmitChanges();
```

Listing 5.6: Esempio di inserimento in VB

```
Dim context As New ORMDataContext()

Dim c As New Customer
c.CustomerID = "ID180"
c.City = "Roma"
c.ContactName = "Massimo Decimo Meridio"
c.Address = "Via Appia"
c.CompanyName = "colosseo.com"

context.Customers.InsertOnSubmit(c)
context.SubmitChanges()
```

5.4 LINQ to SQL

LINQ to SQL è una delle implementazioni di LINQ che sono state rilasciate con Visual Studio 2008. LINQ to SQL è il modo più semplice per poter lavorare con SQL Server usando un nuovo modello di programmazione in C# 3.0 e Visual Basic 9. In questo modo nel nostro linguaggio .NET preferito scriviamo del codice che si avvicina ad una sintassi SQL rendendo di fatto meno complicato far "parlare" le applicazioni fatte di classi, cicli e quant'altro con SQL Server.
Con LINQ to SQL in sostanza si mappano uno a uno le tabelle di SQL Server con delle classi e grazie al framework messo a disposizione si è in grado di fare le classiche operazioni di Insert, Update, Delete e Query.
LINQ to SQL mette in campo due tipologie principali di funzionalità:

- permette la definizione di un modello ad oggetti basato sul mapping delle tabelle di un database SQL Server;

- un vero e proprio runtime per la gestione dei meccanismi di aggiornamento, persistenza, gestione delle performance e della cache.

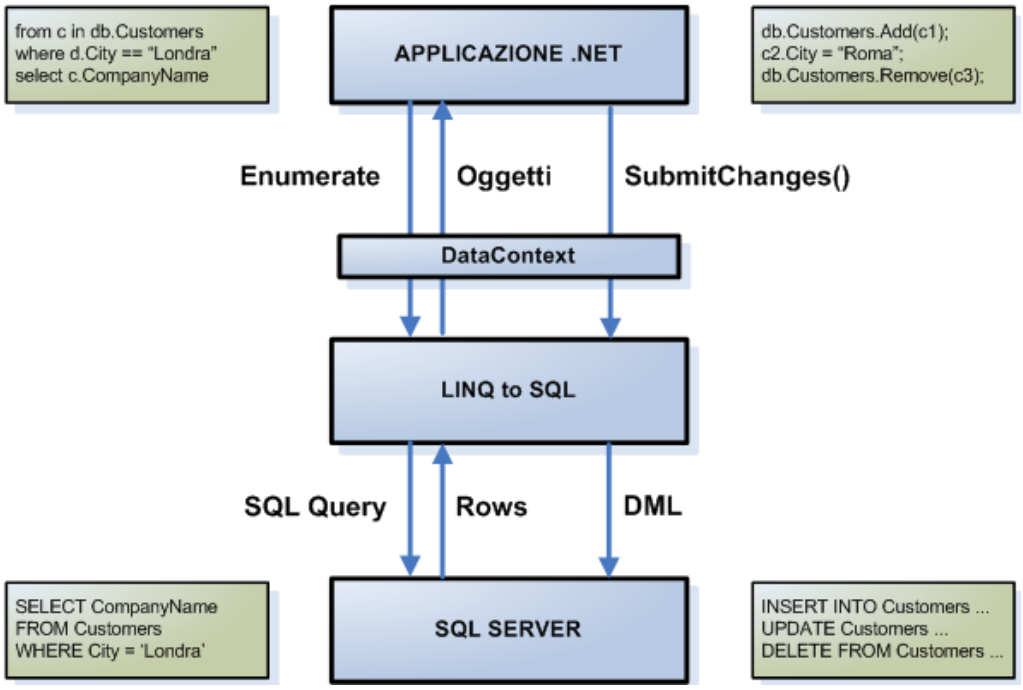

Figura 5.4: Struttura di LINQ to SQL

A supporto di questa nuova tecnologia, Visual Studio 2008 mette in campo l'*Object-Relational Designer*, che consente di effettuare il mapping tra applicazione e database, tramite semplici operazioni visuali come il drag & drop. Gli schemi realizzati sono poi memorizzati con estensione *.dbml*.
Il mapping offerto da LINQ to SQL presenta delle relazioni 1:1 tra tabelle della base di dati e classi che andranno poi a far parte del domain model dell'applicazione. In realtà il mapping "si avvicina molto" al pattern 1 a 1, poiché lo si può arricchire con associazioni, collezioni e meccanismi di ereditarietà basati sulla strategia a *Single Table Hierarchy*.
Per favorire questo tipo di funzionalità, LINQ to SQL si basa sul DataContext: una classe che fornisce il punto di incontro tra entità logiche (mapping) e database (entità del RDBMS).
Il *DataContext* rappresenta il concetto di "contesto di persistenza", concetto che sta alla base di ogni ORM che si rispetti; esso si occupa infatti sia delle connessioni fisiche alla base di dati, facendo da wrapper per un'istanza di una classe che implementa l'interfaccia IDBConnection e gestendo l'apertura e la chiusura della connessione, sia di tutti i meccanismi di persistenza delle entity.
Quando si definisce un mapping con l'O-R Designer di Visual Studio 2008 si otteniene sia un file di mapping (.dbml), sia la creazione di una classe che eredita direttamente

da DataContext (nel namespace *System.Data.Linq*), con proprietà tipizzate relative alle tabelle mappate e metodi relativi ad ogni singola stored procedure definita.
Le classi generate da questa operazione di mapping, sono le cosiddette entity, che possono essere descritte appunto come entità logiche che rappresentano le informazioni che dovranno poi essere gestite dai vari livelli concettuali in cui sono divise le applicazioni multilivello.
Queste entità sono gestite dal runtime di LINQ to SQL, che invece si occupa della connessione fisica tra applicazione e base di dati. Questo permette di effettuare operazioni sul database, secondo il modello di programmazione ad oggetti, senza dover specificare alcuna riga di codice T-SQL, in quanto è proprio il runtime stesso che provvede alla conversione delle operazioni in costrutti sintattici propri del linguaggio con cui solitamente si accede alle informazioni poste nel database.

5.4.1 Il tool SQLMetal

Il .NET Framework 3.5 mette a disposizione un tool a riga di comando denominato *SQLMetal.exe* che consente di creare un file di mapping, in C# oppure VB.NET, con cui accedere ad un database SQL senza doversi preoccupare di stringhe di connessione, oggetti Command e Parameter. Mediante l'applicazione delle opzioni è possibile utilizzare SqlMetal per eseguire diverse azioni, fra cui:

- a partire da un database, generare codice sorgente e attributi di mapping oppure un file di mapping;
- a partire da un database, generare un file DBML (Database Markup Language) intermedio da personalizzare;
- a partire da un file con estensione dbml, generare codice e attributi di mapping oppure un file di mapping.

Eseguendo il tool SQLMetal.exe su tale database si ottiene un file di mapping che deve essere aggiunto al proprio progetto .NET. Fatto questo, bastano pochissime righe di codice per accedere al database con LINQ, leggere i valori in esso contenuti e fare delle modifiche. Questo strumento non può essere utilizzato per poter generare automaticamente i file di Entity da RDBMS diversi da SQL Server in quanto SqlMetal.exe è un tool da riga di comando di supporto a LINQ to SQL, tecnologia totalmente basata sul prodotto della Microsoft, per il quale è decisamente ottimizzato. Ecco perché sono nati altri progetti da terzi di supporto a questa funzionalità, come *Devart dotConnect* [45]: questo sistema fornisce un query provider personalizzato per poter utilizzare i database della Oracle.

5.4.2 L'applicazione Linqer

Poiché la sintassi di LINQ è simile al SQL ma non è del tutto uguale si può pensare, almeno inizialmente, ad utilizzare dei programmi che ne permettano la conversione automatica delle query da sintassi SQL in sintassi LINQ, magari potendo far scegliere anche il linguaggio di riferimento tra C# e VB.NET. ***Linqer*** [37] è un'applicazione realizzata per effettuare ciò.
Il programma richiede, oltre al codice SQL da convertire, la connesione al database SQL Server e che sia già stato realizzato il mapping tra l'applicazione e il database (memorizzati nel file *.dbml*) e il file riguardo la struttura (file *designer.cs*).
Questi file sono generati automaticamente all'interno di Visual Studio quando si sceglie l'inserimento di un componente LINQ to SQL. Purtroppo in Linqer non si possono utilizzare data provider differenti, quindi non vi è la possibilità di poter connettere database differenti a SQL Server.

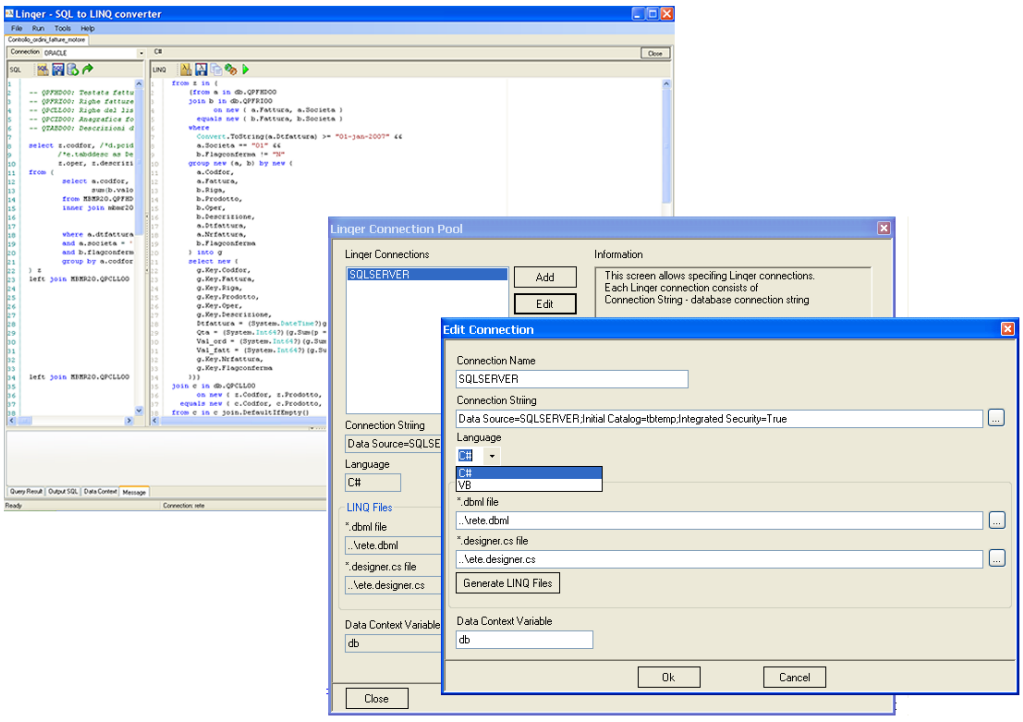

Figura 5.5: Linqer - SQL to LINQ converter

In figura 5.5 si può vedere in alto la conversione della macro "*Controllo_ordini_fatture*" in sintassi LINQ e sotto invece vi è una schermata dell'applicazione riguardante le impostazioni di connessione. Da qui si può notare la possibilità di poter generare il LINQ file direttamente senza ricorrere al tool SQLMetal da linea di comando.
Tale applicazione può essere utile solo per i primi passi in LINQ, non oltre poiché non supporta tutte le funzioni introdotte da tale tecnologia.

5.5 ADO.NET Entity Framework

Gli sviluppatori impiegano spesso ore e ore per decifrare gli schemi di database e scrivere query complesse per recuperare i dati necessari da utilizzare nelle relative applicazioni. ***ADO.NET Entity Framework*** [38] semplifica queste attività e consente agli sviluppatori di concentrarsi sulla logica di business delle relative applicazioni.
I sistemi aziendali utilizzano spesso dati di diverse e molteplici origini che utilizzano schemi e convenzioni di denominazione diverse. In più, queste origini dati utilizzano spesso vari livelli di normalizzazione, con la conseguenza che le informazioni di uno specifico elemento di business vengono sparse in più tabelle e righe. Il risultato è che gli sviluppatori devono scrivere una grossa quantità di logica delle applicazioni per gestire queste complesse relazioni di database.
ADO.NET Entity Framework, che è basato su un Entity Data Model, consente agli sviluppatori di trasformare i dati relazionali presenti negli schemi di database in entità concettuali che possono essere utilizzate direttamente nelle applicazioni. I dati dei clienti contenuti in un'applicazione, ad esempio, possono essere memorizzati in più tabelle di un database. Utilizzando ADO.NET Entity Framework, i progettisti e gli sviluppatori possono definire una singola entità cliente concettuale che astrae esattamente le complesse relazioni necessarie per accedere e aggiornare i dati del cliente da un'applicazione. Questo livello di astrazione isola la logica di accesso ai dati in una serie di entità ben definite che possono essere utilizzate in un'applicazione e l'astrazione consente agli sviluppatori di concentrarsi sullo sviluppo della logica dell'applicazione.
ADO.NET Entity Framework fornisce un'interfaccia di programmazione dei dati che rende:

- *facile comprendere il modello di dati concettuale*: utilizzando Entity Data Model è possibile utilizzare i dati in termini di logica di business nell'applicazione, invece che di schema logico nell'origine dati;
- *facile progettare e sviluppare le applicazioni*: sviluppando applicazioni che allineano logica di business alla logica di accesso ai dati è considerevolmente più semplice per i progettisti pianificare le applicazioni e per gli sviluppatori scrivere il codice;
- *facile gestire le applicazioni*: utilizzando un modello di dati concettuale, gli sviluppatori possono concentrarsi sulla logica di business di un'applicazione, invece che sulla logica di archiviazione dei dati. Inoltre ADO.NET Entity Framework protegge le applicazioni dalle modifiche allo schema di dati sottostante, minimizzando in tal modo i tempi dedicati alla manutenzione.

Poiché Entity Data Model utilizza entità anziché tabelle e righe, gli sviluppatori necessitano di un linguaggio di query che interagisca con questi oggetti. Entity SQL è un nuovo

linguaggio che consente l'esecuzione di query dichiarative basate su set e aggiornamenti di entità e relazioni in Entity Data Model. Entity SQL è in teoria indipendente dal provider di dati ed è quindi possibile riutilizzare le query create con diversi provider di database, il che consente di risparmiare tempo.
La maggior parte degli sviluppatori utilizza linguaggi di programmazione orientata a oggetti quali C# e Visual Basic per scrivere nuovo codice nelle applicazioni aziendali. Questi linguaggi modellano le entità come classi e i relativi funzionamenti come codice, a differenza di ADO.NET che espone i dati come valori. Questo tipo di operazione introduce una mancata corrispondenza tra i dati e l'applicazione. ADO.NET Entity Framework fornisce un livello di servizi oggetto che riduce questo problema. Gli sviluppatori possono utilizzare Object Services per creare query e per restituire, gestire e aggiornare i risultati come oggetti business. ADO.NET Entity Framework genera classi .NET dalle entità di Entity Data Model in uno schema. Queste classi sono classi parziali così che gli sviluppatori possano estenderle con una logica di business personalizzata senza influire sul codice generato. È possibile eseguire query per questi oggetti business utilizzando Entity SQL o LINQ (Language Integrated Query).

5.6 LINQ to Entities

Con l'uscita del Service Pack 1 del .NET Framework 3.5, Microsoft ha introdotto un nuovo ORM di nome *Entity Framework*, un progetto completamente diverso da LINQ to SQL in quanto con questo condivide solo il fatto di essere un ORM e di utilizzare LINQ come linguaggio principale di interrogazione.
LINQ to Entities [39] è un'altra implementazione di LINQ fatta per parlare con l'ADO.NET Entity Framework; sia l' Entity Framework che LINQ to Entities sono attualmente in Beta 3.
Il principio su cui si basa il funzionamento dell'ADO.NET Entity Framework è semplice: non considerare il database come un insieme di tabelle e relazioni ma come un insieme di oggetti entità logiche ed oggetti relazioni. Questo consente agli sviluppatori di lavorare con un maggior livello di astrazione; cioè uno sviluppatore si concentrerà solo sul modello concettuale proprio del modello Entità-Relazione, in maniera indipendente dallo storage sottostante, sia esso SQL Server o un altro database, come Oracle e MySQL. Inoltre il modello ad oggetti usato con Entity Framework è diverso da quello usato dal designer di Visual Studio per LINQ to SQL in quanto permette di lavorare con relazioni molti-a-molti.

Quindi i vantaggi che ne derivato dall'utilizzo dell'Entity Framework sono:

- *motore di memorizzazione dei dati* e *indipendenza dello schema*: attraverso la

creazione di uno strato di astrazione tra il schema logico e lo schema concettuale, l'applicazione non si ha più bisogno di conoscere lo schema di memorizzazione fisica dei dati; modello concettuale più espressivo: i tipi di entità possono ereditare da altri tipi di entità in modo tale da permetterne la creazione di forme più complesse in aggiunta ai tipi standard presenti nei database;

- *accesso a più sistemi database*: poiché l'applicazione non è più dipendente dalla memorizzazione fisica dei dati, lo sviluppatore ha la possibilità di accedere ai dati provenienti da diverse fonti utilizzando un singolo object model;
- *LINQ*: offre il controllo della sintassi delle query al tempo di compilazione.

Entity Framework genera una classe derivata da *ObjectContext* che rappresenta il contenitore di entità nel modello concettuale. Questo contesto dell'oggetto fornisce le funzionalità di registrazione delle modifiche e di gestione di identità, concorrenza e relazioni. Questa classe espone inoltre un metodo SaveChanges che scrive inserimenti, aggiornamenti ed eliminazioni nell'origine dati. Analogamente alle query, queste modifiche vengono eseguite da comandi generati automaticamente dal sistema o da stored procedure specificate dallo sviluppatore.

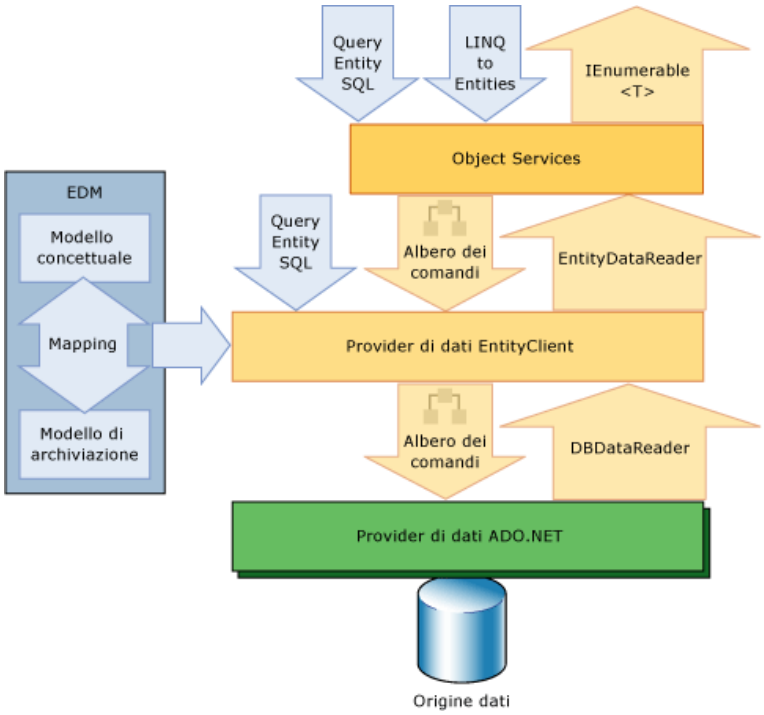

Figura 5.6: Architettura di Entity Framework per l'accesso ai dati

5.7 Le differenze e la miglior scelta tra LINQ to SQL e LINQ to Entities

Perché se esiste già LINQ to SQL, Microsoft ha creato un nuovo ORM? Esistono diverse spiegazioni per questa scelta da parte di Microsoft che hanno motivazioni più o meno valide. LINQ to SQL è pensato per essere il modo più rapido per accedere a SQL Server ed in scenari in cui il modello ad oggetti si mappa 1:1 con le tabelle del database di riferimento. Dato c'è che un mapping 1:1 (o al massimo 1:N se per una tabella si definisce una gerarchia di classi) con le tabelle del database si deve portare dietro anche la tabella di associazione, dovendo quindi gestire un'entità che in un Domain Model disegnato nativamente, e non ricavato dal modello relazionale del database, non ci sarebbe mai stata. Inoltre le classi saranno fortemente legate alle tabelle, tanto che gli oggetti di supporto di LINQ to SQL si chiamo essi stessi "table" (*System.Data.LINQ.Table*). Ovviamente ciò non è auspicabile poiché il modello ad oggetti non è libero di essere quello che dovrebbe, ma il rovescio della medaglia è nel minor tempo speso per lo sviluppo delle applicazioni e per applicazioni la cui vita è breve, e quindi il ciclo dello sviluppo deve essere anch'esso breve, quindi risulta una soluzione sicuramente interessante.
Quindi LINQ to SQL rappresenta la giusta soluzione per realizzare applicazioni *RAD* (*Rapid application development*) o per realizzare applicazioni in cui il mapping più sofisticato di Entity Framework non è necessario: in questo scenario LINQ to SQL rappresenta il modo più rapido di lavorare con LINQ e SQL Server. Mentre per applicazioni in cui non è pensabile avere il Domain Model derivato dal Modello Relazionale del database (nè viceversa) c'è LINQ to Entities, ed in generare l'Entity Framework.
Entity Framework permette di descrivere il proprio modello applicativo pensando al Modello Entità-Associazione ed è possibile mappare poi il modello creato sulle tabelle del database relazionale sottostante (sia esso SQL Server o un altro DBMS di quelli che sono e saranno supportati dall'Entity Framework). L'architettura realizzata si basa sull'utilizzo di tre file XML che descrivono come deve avvenire il mapping (vedi paragrafo successivo). In generale Entity Framework sarà più adatto ad ambienti in cui viene richiesto il supporto a database diversi da SQL Server e in cui l'evoluzione del database stesso avviene spesso ad opera di persone diverse da quelle che scrivono le applicazioni. In questi scenari è tipico avere un elevato numero di tabelle che rappresentano logicamente un'entità o relazione di ereditarietà tra queste.
Comunque in entrambi gli approcci è possibile specificare di poter utilizzare le proprie stored procedure per accedere ai dati presenti nelle tabelle, e questo è sicuramente fondamentale per garantire manutenibilità e flessibilità del database. Nella figura 5.7 sono messi a confronto le diverse caratteristiche offerte da entrambe le tecnologie.

Feature	LINQ to SQL	LINQ to Entities
Language Extensions Support	Y	Y
Language Integrated Database Queries	Y	Y
Many-to-Many (3way Join/Payload relationship)	N	N
Many-to-Many (No payload)	N	Y
Stored Procedures	Y	N (to be added)
Entity Inheritance	N	Y
Single Entity From Multiple Tables	N	Y
Identity Management / CRUD features	Y	Y

Figura 5.7: Architettura di Entity Framework per l'accesso ai dati

Queste sono solo alcune delle caratteristiche che fanno preferire Entity Framework rispetto a LINQ to SQL in quanto permette una migliore gestione della concorrenza e mette a disposizione una serie di driver di terze parti (data provider) già pronti per supportare i database più diffusi.
L'Entity Framework è un ORM molto complesso e la sua struttura è composta da diversi componenti, ma oltre a quanto già scritto non si desidera continuare poiché lo studio dell'infrastruttura software non è l'argomento principale di questo capitolo, ma si voleva dare solo un'ampia panoramica. Adesso vengono esaminate le soluzioni per poter interagire con tale tecnologia ed adattarla ai nostri scopi.

5.8 Realizzazione dei modelli concettuali

Uno schema di progettazione comune e ormai consolidato per la modellazione dei dati prevede la suddivisione del modello di dati in tre parti: un *modello concettuale*, un *modello logico* e un *modello fisico*. Il modello concettuale definisce le entità e le relazioni nel sistema da modellare. Il modello logico per un database relazionale normalizza le entità e le relazioni in tabelle con vincoli di chiave esterna. Il modello fisico gestisce le funzionalità di un determinato motore dei dati specificando dettagli sull'archiviazione come il partizionamento e l'indicizzazione.
Il modello fisico viene ridefinito dagli amministratori del database per migliorare le prestazioni, ma i programmatori che scrivono il codice delle applicazioni tendono a utilizzare solo il modello logico scrivendo query SQL e chiamando stored procedure. I modelli concettuali in genere vengono utilizzati come uno strumento per l'acquisizione e la comunicazione dei requisiti di un'applicazione, spesso come diagrammi inerti visualizzati e discussi nelle fasi iniziali di un progetto e quindi abbandonati. Molti team di sviluppo saltano la fase

di creazione di un modello concettuale e partono direttamente dall'indicazione di tabelle, colonne e chiavi in un database relazionale.
Entity Framework favorisce la realizzazione dei modelli concettuali in quanto consente agli sviluppatori di eseguire query su entità e associazioni del modello concettuale basandosi al tempo stesso su Entity Framework per convertire le operazioni in comandi specifici dell'origine dati. Le applicazioni non sono quindi più vincolate a dipendenze hard-coded su una determinata origine dati. Il modello concettuale, il modello di archiviazione e il mapping tra i due modelli sono espressi in una specifica esterna, nota come *Entity Data Model* (*EDM*) [43]. Il modello di archiviazione e i mapping possono cambiare in base alle esigenze senza richiedere modifiche al modello concettuale, alle classi di dati o al codice dell'applicazione. Poiché i modelli di archiviazione sono specifici del provider, è possibile utilizzare un modello concettuale coerente in varie origini dati.
Un modello EDM viene definito dai tre file modello e di mapping riportati di seguito che includono le estensioni del nome file corrispondenti:

- File *CSDL* (*Conceptual Schema Definition Language*) con estensione *csdl*: definisce il modello concettuale.
- File *SSDL* (*Store Schema Definition Language*) con estensione *ssdl*: definisce il modello di archiviazione, chiamato anche modello logico.
- File *MSL* (*Mapping Specification Language*) con estensione *msl*: definisce il mapping tra il modello di archiviazione e il modello concettuale.

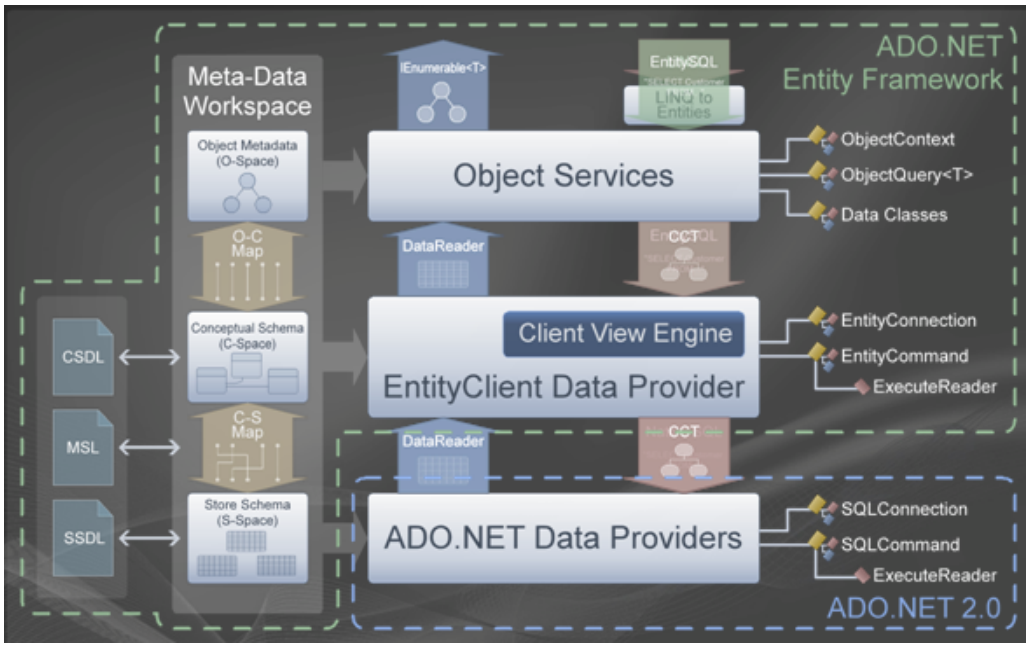

Figura 5.8: Architettura dell'Entity Framework

Nel file SSDL viene inserita la descrizione del database con tutte le sue tabelle, i singoli campi, le relazioni, le chiavi primarie e tutto quanto è necessario per descrivere la struttura del database sottostante. Anche le funzioni e le stored procedure devono essere incluse in questa sezione.
Nel file CSDL viene specificata la struttura delle classi di dominio con tutte le colonne, il tipo di dato che contengono, le relazioni con altre classi, ecc.
Una volta che si ha a disposizione la descrizione dei due mondi da far colloquiare, nel file MSL si inseriscono le informazioni di mapping tra lo schema delle classi e le tabelle del database. Questo è probabilmente quello più complesso soprattutto quando entrano in gioco meccanismi di mapping complessi che non fanno un semplice mapping 1:1 tra tabella e classe (ereditarietà, tipi complessi, ecc.).
Questi file modello e di mapping basati su XML vengono utilizzati da Entity Framework per trasformare le operazioni di creazione, lettura, aggiornamento ed eliminazione di entità e associazioni del modello concettuale in operazioni equivalenti nell'origine dati. Entity Framework è accompagnato da un designer visuale che viene aggiunto a Visual Studio e che copre buona parte delle necessità, lasciando allo sviluppatore il solo scopo di disegnare visualmente le classi e la generazione del codice e del file di mapping. Tuttavia, non tutte le operazioni possono essere svolte dall'editor visuale, quindi conoscere la sintassi del file di mapping risulta fondamentale in alcuni casi.

5.9 LINQ to Oracle

Fino ad adesso è stata data un'ampia introduzione sull'argomento LINQ e sui diversi modi di accesso alle informazioni, ma ora si vuole analizzare come tale tecnologia può essere applicata ai nostri scopi.
Ma perché abbiamo fatto l'introduzione di questo nuovo argomento, specialmente se è già stata data una pratica realizzazione di ciò che richiedeva l'azienda? Durante lo sviluppo delle applicazioni in C# si è sentito un po' il distacco tra la parte object oriented, legata alla realizzazione dell'interfaccia grafica e alla connessione al database, e la parte relativa alla creazione delle query direttamente in SQL, quindi orientata ai dati.
LINQ non è una tecnologia nata durante lo svolgimento di questa tesi, ma è già stata rilasciata dalla Microsoft verso la fine del 2007, prima come beta release per poi diventare versione ufficiale con l'uscita di Visual Studio 2008. Però nel frattempo sono nate diverse soluzioni, anche da terze parti, per effettuare integrazioni non supportate direttamente da LINQ, come l'accesso da parte di LINQ to SQL a database differenti a SQL Server.

Poiché non sono attualmente previste soluzioni per poter accedere ad altri RDBMS, sono nati progetti realizzati da terzi, sia open source che commerciali, che cercano di poter

interfacciare i due mondi. Tra i diversi progetti che sono nati in questo ambito, si sono analizzati quelli che permettono di utilizzare LINQ interfacciandolo su RDBMS Oracle (versione 10g) con lo scopo di poter effettuare le query che si sono viste nel capitolo 4.
I progetti che sono risultati interessanti sono:

- ***DbLINQ Project*** [40]: è un progetto che fornisce il LINQ provider che permette di collegare qualsiasi database a LINQ to SQL attraverso opportune API. Sono supportati i seguenti RDBMS: MySQL, Oracle e PostgreSQL, SQLite, Ingres, Firebird ed, ovviamente, SQL Server.

- ***LINQ To Oracle Project*** [41]: è un progetto che viene distribuito su CodePlex, il sito web dove vengono pubblicati e distribuiti i progetti open source della Microsoft. Tale progetto permette di avere un *query provider* personalizzato per database Oracle (funzionante su Oracle 10g, Oracle XE e Oracle 11).

- ***LINQ to NHibernate*** [42]: è un LINQ provider open source da poter utilizzare con l'ORM NHibernate.

Questi sono solo alcuni progetti nati a riguardo alla necessità di poter utilizzare Oracle come RDBMS per LINQ. Altre soluzioni interessanti sono state realizzate dalla Devart.

La ***Devart*** [44], precedentemente conosciuta come Core Lab, mette a disposizione diversi strumenti commerciali che nell'insieme permettono di poter effettuare un accesso completo ad database, non strettamente SQL Server, e di poter utilizzare entrambe le tecnologie introdotte in precedenza, ovvero LINQ to SQL e Entity Framework. Realizza diversi database tools per SQL Server, Oracle e MySQL date dalle diverse partnership dirette con le aziende per poter soddisfare le esigenze richieste dalle aziende di sviluppo software. Dei diversi prodotti della Devart sono stati utilizzati:

- ***dotConnect for Oracle Data Provider*** [45]: precedentemente chiamato OraDirect.NET, è una soluzione che fornisce la funzionalità di data provider con la possibilità di potersi connettere al database server Oracle, con o senza la Oracle Call Interface (OCI), utilizzando l'architettura ADO.NET. In base al database (sono supportati anche altri database oltre ad Oracle e SQL Server, come MySQL, PostgreSQL e SQLite) esistono versioni personalizzate che permettono di avere funzionalità specifiche al database in questione. Inoltre esiste una versione universale, denominata *UniDirect.NET* [46], che è indipendente dal database utilizzato per poter coprire così tutta la gamma dei prodotti del settore. Si integra con l'ambiente di Visual Studio attraverso un menu fra i diversi tool a disposizione (più una serie di docking windows).

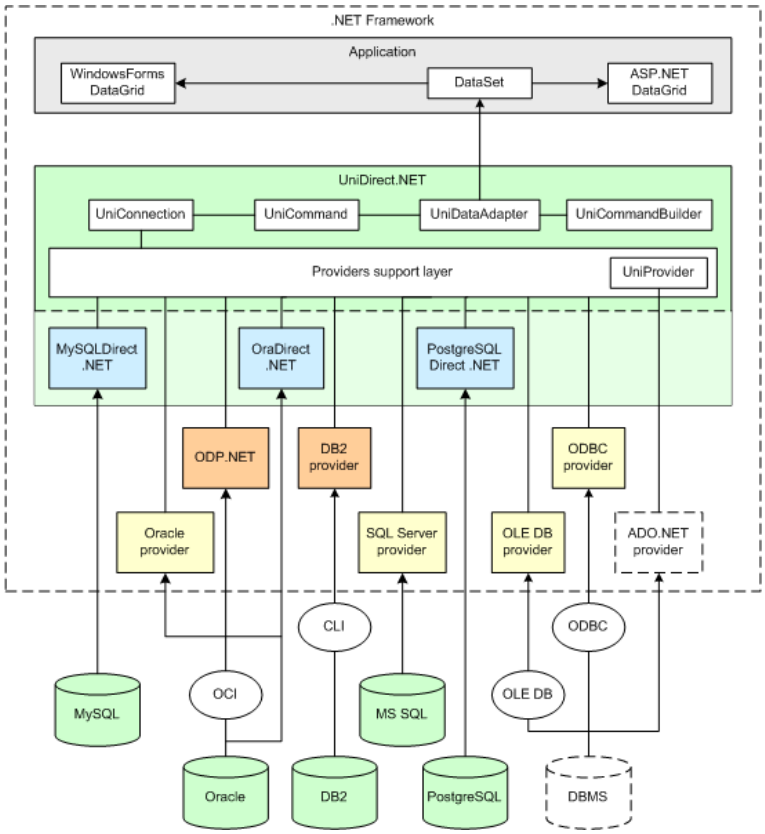

Figura 5.9: L'architettura di UniDirect.NET

- ***Entity Developer for dotConnect*** [47]: è uno strumento che permette di generare modelli e codice per LINQ to SQL e ADO.NET Entity Framework. Permette di progettare un entità model partendo da zero oppure di poter effettuare un reverse engineering da un database esistente, includendo anche le relazioni se esistenti. Tale modello è utilizzato per generare il codice C# o di Visual Basic partendo da template predefiniti, ma personalizzabili in base alle esigenze. Attraverso dotConnect si possono creare i modelli e importarli direttamente nei progetti di Visual Studio. Anche questo prodotto, come il precedente, offre il supporto a molteplici database.

- ***OraDeveloper Tools*** [48]: è un add-in che permette di semplificare lo sviluppo delle applicazioni basate sull'accesso al database Oracle e può essere integrato dentro l'ambiente di Visual Studio e di Delphi. Permette di poter creare diverse connessioni preconfigurate, data adapter o command objects per i seguenti prodotti: *MS OracleClient*, *ODP.NET*, *OLE DB Data Provider*, *dotConnect for Oracle Data Provider* e *Oracle Data Access Components*, dove gli ultimi due sono stati sviluppati dalla Devart. Inoltre esiste una versione denominata *OraDeveloper Studio* che svolge proprio da ambiente di sviluppo, ma più essenziale rispetto a Visual Studio.

Quindi l'insieme delle soluzioni Microsoft (.NET, ADO.NET e LINQ) e l'insieme dei tool messi a disposizione dalla Devart si può realizzare praticamente ***LINQ to Oracle***.

5.9.1 Come realizzare i file di mapping tra applicazioni e database

Il primo esempio che si vuole realizzare è quello di poter implementare un programma già realizzato e verificarne il funzionamento. Il programma che si andrà ad analizzare è "*Movimenti acquisto conto pieno*".
Dall'ambiente di Visual Studio si crea un nuovo progetto dove si aggiunge un nuovo item. Dalla maschera di scelta si seleziona la voce "*Devart LINQ to SQL Model*" e si rinomina il *DataContext* con il nome della macro che si vuole realizzare in LINQ. Da notare che tale elemento ha come estensione *lqml* e differisce dal quella di default di LINQ to SQL in quanto quest'ultima è *dbml*.

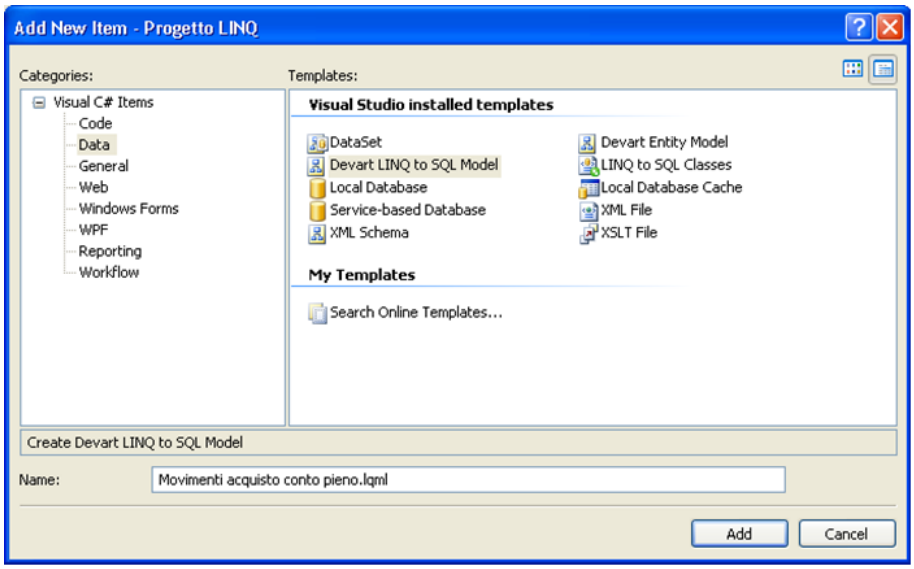

Figura 5.10: Maschera di inserimento di nuovo oggetto presente in Visual Studio

Da qui si apre il tool di creazione dei modelli della Devart denominato *Entity Developer for dotConnect*. Qui si crea la connessione con il database Oracle seguendo un apposito wizard o si carica un modello già creato in precedenza. Se la connessione non era già stata impostata si segue il wizard, denominato "*Database reverse Engineering Wizard*", messo a disposizione dall'applicazione che permette creare il modello da un database esistente e di settare passo dopo passo i parametri di connessione e le tabelle interessate. Già da subito tale strumento permette di poter scegliere il tipo di modello da creare tra LINQ to SQL o Entity Framework.
Si è scelto la prima voce, ovvero LINQ to SQL, e si sono impostati i dati di connessione verso il database Oracle. Da notare che per effettuare la connessione si è utilizzato il *Data*

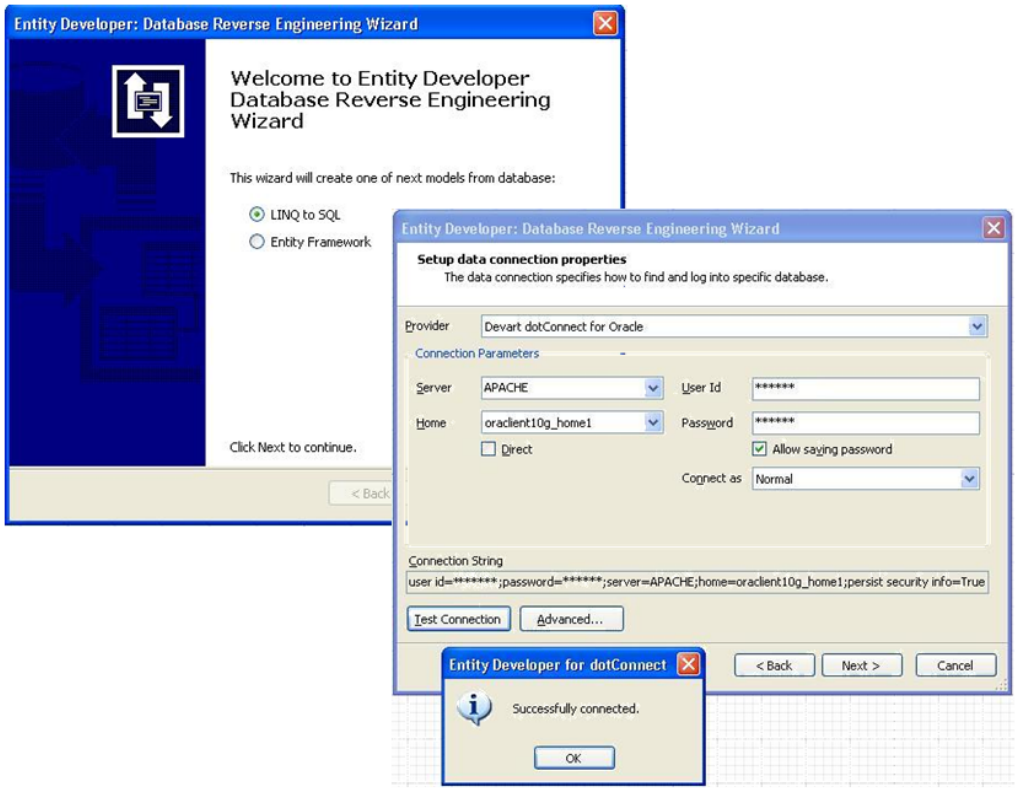

Figura 5.11: Finestre dell'applicazione Entity Developer

Provider dotConnect for Oracle.
Dopodiché si scelgono le tabelle da importare nel progetto corrente ed infine si impostano i nomi del *namespace* e del *DataContext*. Si è voluto nominarli entrambi MACP perché è l'abbreviazione utilizzata nel capitolo 4 per indicare tale macro.
Ora il modello è già pronto ed è automaticamente importato nel progetto di Visual Studio (vedi figura 5.12). Adesso rimane solo da scrivere la query LINQ dentro l'ambiente di sviluppo direttamente in C# o VB.NET. Nella realizzazione delle query LINQ non si è avuto il tempo di poter implementare le query parametriche, quindi si sostituisce tale campo con parametri già impostati.

Anche in LINQ è possibile utilizzare query dinamiche, anche se il relativo supporto non è incluso "di base". Alcuni esempi di utilizzo sono disponibili su MSDN che fornisce la "Dynamic Query Library" [50], ovvero una serie di extension methods con cui è possibile utilizzare una stringa come argomento dell'operatore *Where* [51]. Di seguito vi è un piccolo esempio di come si può implementare una query dinamica.

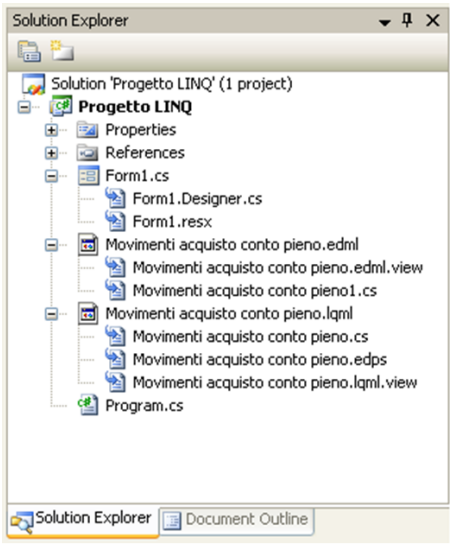

Figura 5.12: Solution Explorer con i modelli LINQ to SQL e Entity Framework

Listing 5.7: Esempio query dinamica in LINQ

```
var query =
    db.Customers.
    Where("City = @0 and Orders.Count >= @1", "Milano", 10).
    OrderBy("CompanyName").
    Select("new(CompanyName as Name, Phone)");
```

5.10 Sviluppi futuri

Le potenzialità di LINQ risultano essere veramente tante e soprattutto, ormai, risulta la scelta ottimale basarsi su di un ORM per gestire l'accesso ai dati e la loro persistenza all'interno delle proprie applicazioni. Nel frattempo sono nate molte applicazioni che permettono di utilizare LINQ e gli argomenti trattati sopra sono solo dei piccoli esempi di cosa si può realizzare con tale ORM: infatti sono stati realizzati diversi data provider per poter interfacciare LINQ con altri ambienti, anche totalmente differenti, come ad esempio *JavaScript*, *LDAP*, *Google*, *RDF Files*, ecc. Qui si può trovare un'ampia lista dei data provider finora realizzati [49].
Delle due versioni di LINQ esaminate in precedenza, mettendo da parte eventuali motivazioni commerciali, Entity Framework è comunque un ORM molto più completo rispetto a LINQ to SQL poiché copre esigenze che con quest'ultimo non possono essere soddisfatte. Parlando di mapping, LINQ to SQL obbliga a modellare le classi di dominio esattamente come le tabelle su cui queste mappano; ad esempio non è possibile avere una classe che contiene dati di due tabelle e viceversa. Entity Framework supera questa limitazione offrendo un mapping pressoché completo e quindi si può definire che sicuramente Microsoft nei prossimi anni punterà soprattutto sullo sviluppo di tale versione.

Conclusioni

A conclusione della tesi posso affermare che l'attività di stage svolta presso la Bosch Rexroth Oil Control S.p.A. è stata significativa in quanto ho potuto vivere in prima persona le vicende aziendali di un grande gruppo con legami internazionali. Lo stage è durato sei mesi, ma solo in un arco di tempo così lungo è stato possibile cogliere alcuni meccanismi aziendali di una struttura societaria così complessa, dato anche dai diversi stabilimenti produttivi distribuiti nel territorio tra Modena e Reggio Emilia.
Tra i tanti aspetti che ho riscontrato in azienda, mi ha colpito positivamente l'ambiente di lavoro dinamico e giovane e l'importanza che viene data al lavoro di squadra. Nessuna attività viene svolta da un unico soggetto ma vi è sempre un team di lavoro che contribuisce allo svolgimento, anche tra diversi reparti, raggiungendo così risultati più efficienti ed efficaci.

Dentro l'area aziendale in cui ero inserito (reparto ISY - Information System) ho avuto la possibilità di poter realizzare le soluzioni che mi sono state richieste come scopo del tirocinio. Il lavoro eseguito è risultato estremamente positivo in quanto riesce a soddisfare una buona parte dei requisiti che erano stati prefissati.
Il database reverse engineering effettuato sui programmi Access è stato essenziale per poter capire in fondo la logica utilizzata dal sistema gestionale ERP Apache, per poi aver la possibilità di poter effettuare il porting su database server Oracle attraverso la riscrittura completa delle query SQL. I risultati ottenuti soddisfano tutti i presupposti stabiliti: tempo di esecuzione, eliminazione delle parti superflue, minor carico di informazioni da essere inviate dal database server al programma richiedente inviando solo i dati necessari, diverse modalità di impiego delle stesse query e maggior controllo sui dati accessibili.
Ma per poter soddisfare le esigenze legate all'esecuzione delle stesse query ed alla sicurezza sugli accessi e sui dati è stato realizzato un programma in C per gli utenti finali in modo tale da poter eseguire facilmente le query realizzate. Tale programma permette di poter essere distribuito facilmente agli utenti e di tenere sotto controllo le query effettivamente eseguite in quanto risiedono in remoto su directory accessibili solo ai responsabili ISY.
Come parte integrativa non attinente allo stage ho effettuato una ricerca su LINQ (Language-Integrated Query) e di come può essere inserito nell'attività svolta, specialmente inte-

grando tale tecnologia con il RDBMS Oracle. Questa parte è nata dall'esigenza di voler indicare un possibile sviluppo del lavoro svolto e di come può essere realizzato in futuro.

Bibliografia

[1] Documenti aziendali della Bosch Rexroth Oil Control S.p.A.

[2] Gruppo Bosch:
http://www.bosch.com/

[3] Gruppo Bosch Italia:
http://www.bosch.it/

[4] Bosch Rexroth AG:
http://www.boschrexroth.com/

[5] Bosch Rexroth Italia
http://www.boschrexroth.it/

[6] Bosch Rexroth Oil Control S.p.A.:
http://www.oilcontrol.com/

[7] Bosch Rexroth Oil Control S.p.A.
http://www.bosch.it/stampa/comunicato.asp?idCom=518

[8] D. Boldrini, *Bosch Rexroth Oil Control S.p.A. "Alla conquista del mondo"* - 2006:
http://www.scieditrice.com/_pdf/ris/0106/Bosch.pdf

[9] Oil Control S.p.A.:
http://www.oilcontrolgroup.com/

[10] La meccanica in Emilia-Romagna:
http://www.investinemiliaromagna.it/wcm/investiner/pagine/Un_sistema_ad_alta_specializzazione/meccanica.htm

[11] PLM (Product Lifecycle Management):
http://it.wikipedia.org/wiki/Gestione_del_ciclo_di_vita_del_prodotto)

[12] MRP (Manufacturing Resource Planning):
Appunti Prof.ssa P. Monari del corso *Sistemi Informativi della Produzione*:

https://www.ing.unimo.it/campusone/VisualizzazioneIngegneria/MaterialeDidattico.asp?IdInsegnamento=7396

[13] C. Elston, *ERP/MRP Presents Unique Challenges*: http://www.devicelink.com/mddi/archive/04/10/021.html

[14] Apache MBM: http://www.mbm.it/

[15] Nicim: http://www.nicim.it/

[16] WHMS: http://webdoc.siemens.it/CP/IS/ProdottiSoluzionieServizi/Lenostresoluzioni/LogisticadiProduzione/SwgestioneWHMS.htm

[17] SAP: http://www.sap.com/

[18] Quarta Blulink: http://www.blulink.com/

[19] J. L. Viescas, *Microsoft Office Access 2003 Inside Out* - 2004

[20] Power Quest SQL Navigator: http://www.quest.com/sql-navigator/

[21] D. Beneventano, S. Bergamaschi e M. Vincini, *Progetto di Basi di Dati Relazionali*

[22] P. Ciaccia e D. Maio, *Lezioni di Basi di Dati*

[23] F. Ricca e P. Tonella, *Reverse Engineering di sistemi software - Limiti e potenzialità*, Mondo Digitale n.3 pp.52-63 - 2006 http://www.mondodigitale.net/Rivista/06_numero_4/Ricca_p._52-63_.pdf

[24] Semantic Designs: http://www.semdesigns.com

[25] Code Visual to Flowchart: http://www.fatesoft.com/s2f/

[26] Source-Navigator: http://sourcenav.sourceforge.net/

[27] Imagix 4D: http://www.imagix.com/

[28] EclipseUML:
http://www.omondo.com/

[29] Codecrawler:
http://www.inf.unisi.ch/faculty/lanza/codecrawler.html

[30] Rigi:
http://www.rigi.csc.uvic.ca/

[31] M. Van Emmerik, *Decompilation And Reverse Engineering*:
http://www.program-transformation.org/Transform/DecompilationAndReverseEngineering

[32] MSDN Building a Drop-Down Filter List for a DataGridView Column Header Cell:
http://msdn.microsoft.com/en-us/library/aa480727.aspx

[33] Aspose:
http://www.aspose.com/

[34] Microsoft LINQ Project:
http://msdn.microsoft.com/en-us/netframework/aa904594.aspx

[35] Sintassi delle query e sintassi dei metodi (LINQ):
http://msdn.microsoft.com/it-it/library/bb397947.aspx

[36] Domain Model:
http://robertocaico.wordpress.com/2008/03/22/domain-model-e-orm/

[37] Linqer:
http://www.sqltolinq.com/

[38] Introduzione a Entity Framework:
http://msdn.microsoft.com/it-it/library/bb399567.aspx

[39] LINQ to Entities:
http://msdn.microsoft.com/it-it/library/bb386964.aspx

[40] DbLinq Project:
http://code.google.com/p/dblinq2007/

[41] LinqToOracle Project:
http://www.codeplex.com/LinqToOracle/

[42] LINQ to NHibernate:
http://www.ayende.com/

[43] Entity Data Model:
http://msdn.microsoft.com/it-it/library/bb387122.aspx

[44] Devart:
http://www.devart.com/

[45] Devart dotConnect for Oracle Data Provider:
http://devart.com/dotconnect/oracle/

[46] Devart UniDirect.NET:
http://devart.com/unidirect/

[47] Devart Entity Developer for dotConnect:
http://devart.com/entitydeveloper/

[48] Devart OraDeveloper Tools:
http://devart.com/oradevtools/

[49] Lista di Provider per LINQ:
http://blogs.dotnethell.it/coach/Lista-di-Provider-per-Linq__13075.aspx

[50] MSDN Dynamic Query Library:
http://msdn.microsoft.com/en-us/bb330936.aspx

[51] Dynamic LINQ di Scott Guthrie:
http://weblogs.asp.net/scottgu/archive/2008/01/07/dynamic-linq-part-1-using-the-linq-dynamic-query-library.aspx

Ringraziamenti

Giunto al termine di questo lavoro desidero ringraziare ed esprimere la mia riconoscenza nei confronti di tutte le persone che, in modi diversi, mi sono state vicine e hanno permesso e incoraggiato sia i miei studi che la realizzazione e stesura di questa tesi. I miei più sentiti ringraziamenti vanno a chi mi ha seguito durante la redazione del lavoro di tesi.
Desidero innanzitutto ringraziare la Prof.ssa Sonia Bergamaschi per i preziosi insegnamenti e le ore dedicate alla tesi ed alla sua costante disponibilità a dirimere i miei dubbi durante la stesura di questo lavoro.
Doverosi ringraziamenti per tutti i colleghi incontrati presso l'azienda Bosch Rexroth Oil Control S.p.A., in particolare tutto il reparto ISY (Information System) di Nonantola per avermi offerto la possibilità di svolgere uno stage su nuove tematiche, ambienti e strumenti all'avanguardia, per aver mostrato fiducia e interesse nelle mie idee e la pazienza riposta nei miei confronti nel rispondere sempre alle mie domande date da un'innata curiosità in tutto ciò che riguarda la sfera informatica e aziendale.
Per ultimi, ma di certo non per importanza, ringrazio la mia famiglia e gli amici che mi sono stati molto vicini in tutti questi anni "da studente", che oltre ad avermi sempre "supportato" mi hanno più di tutto "sopportato".
Il mio primo pensiero, ovviamente, va ai miei genitori che mi hanno sempre motivato e dato la spinta giusta ad andare avanti ed attraverso il loro aiuto e consigli non avrei mai raggiunto questa meta. Un forte ringraziamento a mio fratello Francesco in quanto mi ha sempre dato una mano per tutto quello che riguarda la sfera aziendale e di marketing d'azienda che mi è servito per potermi meglio addentrare e capire le politiche che governano il mondo economico, ma più che altro è stato ed è un amico a cui sono davvero legato e posso sempre fare affidamento.
Rimarrà in me il piacevole ricordo di questi anni di studio che ho trascorso "a tempo pieno", tranne per un periodo di lavoro e di studio all'estero per il progetto Erasmus, in questo dipartimento e per aver trovato quasi sempre professori disponibili al dialogo e a confrontarsi con le idee altrui, qualità non da tutti. Come non ringraziare tutti gli Amici dell'Università e in modo particolare i miei Compagni di corso e di Progetti universitari con i quali ho condiviso più da vicino questi anni di intenso studio (ma anche di piacevoli svaghi). Ora, più di tutti, possono comprendere il mio grado di soddisfazione.

Inoltre un doveroso ringraziamento di tutti gli studenti e amici, conosciuti durante il periodo di studi ad Horsens (Danimarca) presso l'Università Vitus Bering (ora VIA University) con i quali ho vissuto in pieno l'esperienza che ha segnato la mia vita, dove ai normali impegni di studio sono riuscito ad affiancare volentieri piacevolissimi momenti di svago e viaggi in Europa. È stata un'esperienza estremamente formativa in ambito accademico, notando la diversità di insegnamento rispetto all'Italia, ed alla conclusione di tale periodo mi sono reso conto che sono cambiato facendomi crescere come persona diventando più responsabile di me stesso e delle mie azioni.
Desidero ringraziare tutte quelle persone con cui ho iniziato e trascorso i miei studi, con cui ho scambiato qualche pensiero, qualche idea, qualche risata all'interno del dipartimento. In diversi modi hanno contribuito nel mio percorso formativo, aiutandomi a credere in me stesso, suscitando in me nuovi interessi e soprattutto mi hanno suggerito, direttamente o indirettamente, le modalità per poterli raggiungere.

Giampiero