Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea Triennale in Ingegneria Informatica

Analisi dati in SQL sul database dell'azienda LAR S.p.A.

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Simone Boschi

Anno Accademico 2019-2020

Ringraziamenti

A tutta la mia famiglia, per tutti i sacrifici che hanno compiuto e per avermi sempre supportato e "sopportato" in questi miei anni universitari, soprattutto nei momenti più difficili.

Alla mia fidanzata Veronica, per avermi sempre spronato a dare il massimo e per tutte le interrogazioni fino a tarda notte nei giorni precedenti agli esami e alla sua famiglia.
Un sentito ringraziamento all'Ing. Gaetano Bertolo per avermi aiutato nella correzione della tesi.

Ai miei amici, per esserci sempre stati, in ogni istante e per avermi sempre incoraggiato a non mollare mai.

Al Dott. Gabriele Gandolfi, per avermi aiutato all'interno della LAR, per avermi fatto sempre sorridere in tutte le giornate lavorative e per avermi fornito il materiale utile per la stesura della tesi.

All'Ing. Maurizio Gentile, per tutte le giornate passate a lavorare insieme, per tutte le competenze acquisite e per avermi fornito il materiale per la scrittura di questo testo.

A tutto il personale della LAR S.p.A., in particolare Marco, Dario, Alessandro, Giovanni e al mio tutor aziendale Riccardo Monari per avermi aiutato all'interno della azienda.

Al prof. Nicola Bicocchi, mio tutor universitario per il tirocinio.

Alla mia relatrice, Prof.ssa Sonia Bergamaschi, per aver analizzato il mio lavoro e per avermi aiutato nonostante i tempi stretti.

Indice

1. Introduzione

In questo elaborato vengono descritte le attività svolte durante il tirocinio curricolare presso l'impresa LAR S.p.A..

In particolare, vengono analizzate le attività relative all'analisi dei dati in SQL, oggetto principale delle attività svolte.

1.1. LAR S.p.A.

LAR nasce nel 1939 a Milano.

In questi anni LAR produce pettini, occhiali da sole in resina e trousse e si afferma anche nel mercato americano.

Nel 1941, durante la Seconda Guerra Mondiale, la fabbrica viene completamente distrutta e i proprietari decidono di trasferirla a Formigine (MO).

Nel 1963 il prof. Giulio Natta collaboratore di LAR vince il Premio Nobel per la chimica per l'invenzione del Moplen, una plastica innovativa che viene realizzata per la prima volta proprio in LAR.

Il Moplen (PP-H, polipropilene isotattico) è una plastica creata tramite la polimerizzazione del propilene. Il Moplen è una vera e propria rivoluzione nell'industria della plastica. Ha infatti maggiore resistenza meccanica (ovvero lo sforzo massimo che il materiale riesce a supportare prima di una rottura) e ha inoltre bassi costi di produzione. Il Moplen ancora oggi è utilizzato per la produzione, ad esempio, di tubi di scarico o secchi.

Figura 1 - Pubblicità del Moplen

Negli anni '80 LAR sviluppa per prima il sistema automatico di taglio dello sfrido, ovvero l'eliminazione dei residui di lavorazione.

Tutte queste innovazioni portano LAR a imporsi nel mondo della plastica.

Nel 2012 cambia la proprietà e crescono ulteriormente gli investimenti in uno sviluppo tecnologico e in sostenibilità ambientale.
Sono stati installati impianti di produzione di energia elettrica che utilizzano fonti rinnovabili e che ricoprono oltre l'85% del fabbisogno energetico aziendale.
L'investimento in ricerca e sviluppo prosegue tuttora: l'ultima importante innovazione è la creazione di una nuova divisione, chiamata "camera bianca", per la produzione di occhiali e lenti nella quale i processi di trattamento, assemblaggio e controllo sono totalmente automatizzati.

1.2 Attività svolte durante il tirocinio

Nel tirocinio svolto presso LAR ho effettuato diverse attività.
Gli obiettivi sono stati:

- di controllare il corretto funzionamento di software aziendali,
- la configurazione di nuove macchine, apparecchiature e in generale della parte IT per infrastrutture aziendali,
- la gestione di analisi dei dati del database aziendale tramite tecniche di data mining,
- la pianificazione e l'attuazione per progetti di miglioramento relativi alla parte informatica e tutto ciò che ne deriva.

A partire da manuali complessi e altro materiale interno, ho generato dei tutorial per semplificare l'utilizzo dei software aziendali, ad uso dell'utente finale.

In un primo momento, ho effettuato l'analisi tecnica di tutti i macchinari presenti nel reparto produzione e la "camera bianca".
Insieme ad un tecnico, abbiamo analizzato il codice software ed il funzionamento di uno dei bracci robotici della "camera bianca", dedicato all'assemblaggio degli occhiali.

In affiancamento al responsabile IT di LAR, l'Ing. Maurizio Gentile, abbiamo prodotto diverse analisi dei dati, utilizzando query in linguaggio SQL, basati sulla tecnica di *data mining*, richiesti dagli uffici commerciali ed amministrativi.
Il data mining è ormai uno strumento potente e largamente utilizzato per effettuare analisi di mercato, per analizzare gli acquisti dei singoli clienti e per trovare relazioni tra i dati.

Successivamente, ho contribuito al miglioramento del sito internet aziendale, sviluppato con CMS Wordpress.
Ho creato una nuova sezione del sito "Area Personale - Sicurezza", nella quale ogni dipendente, con le proprie credenziali, ha accesso a un'area formativa inerente alla sicurezza. Sono stati caricati filmati relativi al primo soccorso, al servizio antincendio e al RSPP (Responsabile Servizio Prevenzione e Protezione). Si tratta di uno strumento formativo che permette una prima valutazione delle capacità e conoscenze del candidato, con possibilità di autovalutazione tramite quiz.
Successivamente, ho partecipato alla creazione del nuovo sito web, sviluppato con CMS Wordpress.

Per la pianificazione passo per passo di nuovi progetti, delle nuove produzioni e per visionare lo stato di avanzamento di questi, è stato introdotto l'utilizzo dei diagrammi di GANTT.

Ho partecipato alla migrazione dal server di posta in uso ad una nuova soluzione basata su OFFICE 365 (con contestuale cambio da protocollo POP3 a IMAP) e l'aggiornamento hardware e software di computer aziendali.

L'oggetto dell'elaborato finale è relativo all'utilizzo della tecnica di "data mining, che viene descritto nei capitoli successivi.

2. Data mining

2.1. Knowledge Discovery in Database (KDD)

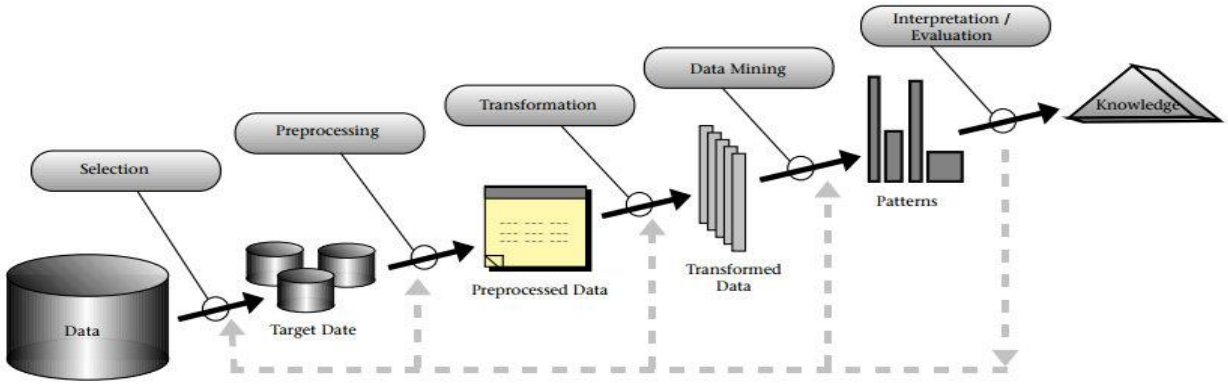

Figura 2 - Processo di Knowledge Discovery in Database

Come si può vedere nella figura sopra riportata, il data mining è solamente una delle fasi per il "Knowledge Discovery in Database" KDD (o processo di estrazione di conoscenza di dati).

Il KDD è un processo che unisce operazioni automatiche a scelte e decisioni umane, ed ha come obiettivo l'estrazione di elementi di conoscenza da insiemi di dati eterogenei e allo stato grezzo.

Le fasi di questo processo sono:

1) **Selection (selezione dei dati):** in un primo momento si definisce l'obiettivo finale, ovvero cosa ci aspettiamo che succeda alla fine dell'intero processo e in base a questo passaggio riusciremo a stabilire il successo o meno del KDD. Fatto questo inizia la vera e propria fase di selezione, dove scegliamo solamente i dati di nostro interesse per perseguire lo scopo finale, creando dei "Target Data".
2) **Preprocessing (pre-processamento dei dati):** rimozione di eventuali inconsistenze, rimozione o riduzione del "rumore", eliminazione dei casi limite, decisione delle strategie per la gestione dei valori nulli e scarto dei dati obsoleti. Scopo finale di questa fase risulta essere "il miglioramento della qualità dei dati selezionati".

3) **Trasformation (trasformazione dei dati):** i dati vengono trasformati e consolidati in formati adatti all'analisi delle tecniche di data mining, riducendo la varietà dei dati e preservando allo stesso tempo la qualità.
4) **Data mining:** utilizzo delle tecniche (elencate nel punto 3) con lo scopo di analizzare i dati e scoprire modelli interessanti o di estrarre conoscenza da questi.
5) **Interpretation/Evaluation (interpretazione/valutazione):** in questa fase si effettua la documentazione e l'interpretazione dei risultati raggiunti. L'interpretazione viene effettuata considerando la conoscenza pregressa e valutando i risultati ottenuti rispetto agli obiettivi prefissati. Nel caso in cui le richieste o gli obiettivi non sono stati raggiunti, si ha la possibilità di tornare alle fasi precedenti.

2.2. Cenni storici

Il data mining nasce negli anni '80 e la sua evoluzione è dovuta da due fattori: l'aumentare delle prestazioni dei calcolatori e la maggiore quantità di dati che abbiamo a disposizione.

La gestione ottimale di quest'ultime è attribuita alle basi di dati, che sono state uno strumento in grado di consentire una maggiore facilità operazionale.

Nel 1960 si assiste ad un primo incremento della mole dei dati immagazzinati dovuti allo sviluppo dei dispositivi informatici. In questa fase i dati sono modellati su base gerarchica o relazionale.

Negli anni successivi, dal 1970 al 1980, si afferma il modello relazionale con lo sviluppo di DBMS (Database Management System) relazionali. I DBMS sono software che si interpongono fra l'utente e il database andando a semplificare la gestione dei dati.

Nell'anno 1994 ha inizio l'evoluzione dei Data Warehouse e dei data mining in chiave moderna.

I Data Warehouse sono sistemi di database, volti ad abilitare e supportare le attività di BI (business intelligence), indipendenti dai sistemi operativi di elaborazione dati. I dati provenienti da fonti diverse ed eterogenee vengono raccolti, compressi e archiviati.

I data mining coniugano le competenze del machine learning (l'abilità di una macchina di compiere compiti nuovi, che non ha mai eseguito, dopo aver appreso le conoscenze necessarie da dati di apprendimento), della statistica e dell'intelligenza artificiale.

Figura 3 - Data Mining

I principali sviluppi del data mining sono la *data visualization* (visualizzazione di dati) e il *social data mining*.

La *data visualization* è una pratica di traduzione delle informazioni in un contesto visivo (mappe o grafici), che semplifica l'identificazione di modelli, tendenze e valori anomali da set di dati.

Con il *social mining* andiamo ad elaborare big data attraverso l'utilizzo da parte dell'utente di social network o delle app.

2.3. Definizione, campi di applicazione e metodi

"Il data mining è l'insieme di tecniche e metodologie che hanno per oggetto l'estrazione di informazioni utili da grandi quantità di dati attraverso metodi automatici o semi-automatici e l'utilizzo scientifico, aziendale, industriale o operativo delle stesse". (https://it.wikipedia.org/wiki/Data_mining)

Ha perciò una duplice valenza:

- l'estrazione di informazione implicita, da dati già strutturati per renderla disponibile e direttamente utilizzabile,
- l'esplorazione ed analisi su grandi quantità di dati al fine di scoprire i pattern (schemi o regolarità) significativi.

I suddetti pattern implicano che i dati sono correlati attraverso una relazione e di conseguenza sono prevedibili - nel caso in cui si ha una mancanza abbiamo una casualità.

Possiamo suddividere le tecniche di data mining (nel seguito indicate semplicemente con data mining) in due macroaree: *predittivo* e *descrittivo*.

Il data mining predittivo acquisisce informazioni sui dati che ha a disposizione effettuando delle ipotesi su valori (sconosciuti e futuri) di variabili di interesse.

Il data mining descrittivo ha come obiettivo di trovare schemi interpretabili che forniscono informazioni sulla struttura dei dati.

L'unica distinzione tra i due tipi di data mining riguarda lo scopo finale, perché i metodi utilizzati sono gli stessi per entrambi ed esistono data mining che sono sia descrittivi che predittivi.

Le principali tecniche di data mining sono:

- **Classificazione**: l'obiettivo di questo metodo è di trovare un profilo descrittivo per ogni classe, che permetta di assegnare oggetti di classe ignota alla classe appropriata.
- **Clustering**: serve per identificare e raggruppare oggetti simili tra di loro, senza aver definito a priori delle classi. Questa omogeneità degli elementi è determinata dal valore degli attributi.
- **Modellazione delle dipendenze**: l'obiettivo di questo metodo è di trovare un modello che descriva dipendenze fra le variabili.
- **Riepilogazione**: metodi che individuano una descrizione compatta dei dati che si vuole analizzare. I risultati finali saranno le informazioni sintetiche sui dati.
- **Regressione**: serve per predire il valore di una variabile sulla base di altre.
- **Individuazione di deviazioni:** permette di scoprire cambiamenti nei dati rispetto a misurazioni eseguite in un momento precedente o valori normati.
- **Regole associative:** metodo utile per l'estrazione e l'identificazione di relazioni nascoste tra i dati. Originariamente proposto nella *market basket*

analysis per la scoperta delle regolarità all'interno delle transizioni registrate nelle vendite nel supermercato.

I principali settori che utilizzano questi metodi di data mining sono: marketing, finanza, scienza, ICT e statistica.

Per fare alcuni esempi:

1) nel campo del marketing si possono utilizzare le tecniche per l'individuazione di tipologie di clientela relative ad un determinato prodotto o verificare in quale luogo un determinato articolo è stato venduto maggiormente. Questi dati serviranno anche in un secondo momento per decidere la distribuzione delle nuove campagne,
2) in ambito finanziario per rilevare frodi e fare previsioni sugli indici azionari,
3) in quello informatico una delle applicazioni fondamentali è la sicurezza con la rilevazione delle anomalie e discordanze tra i dati.

3. Strumenti utilizzati per l'analisi dei dati

3.1. Database aziendale

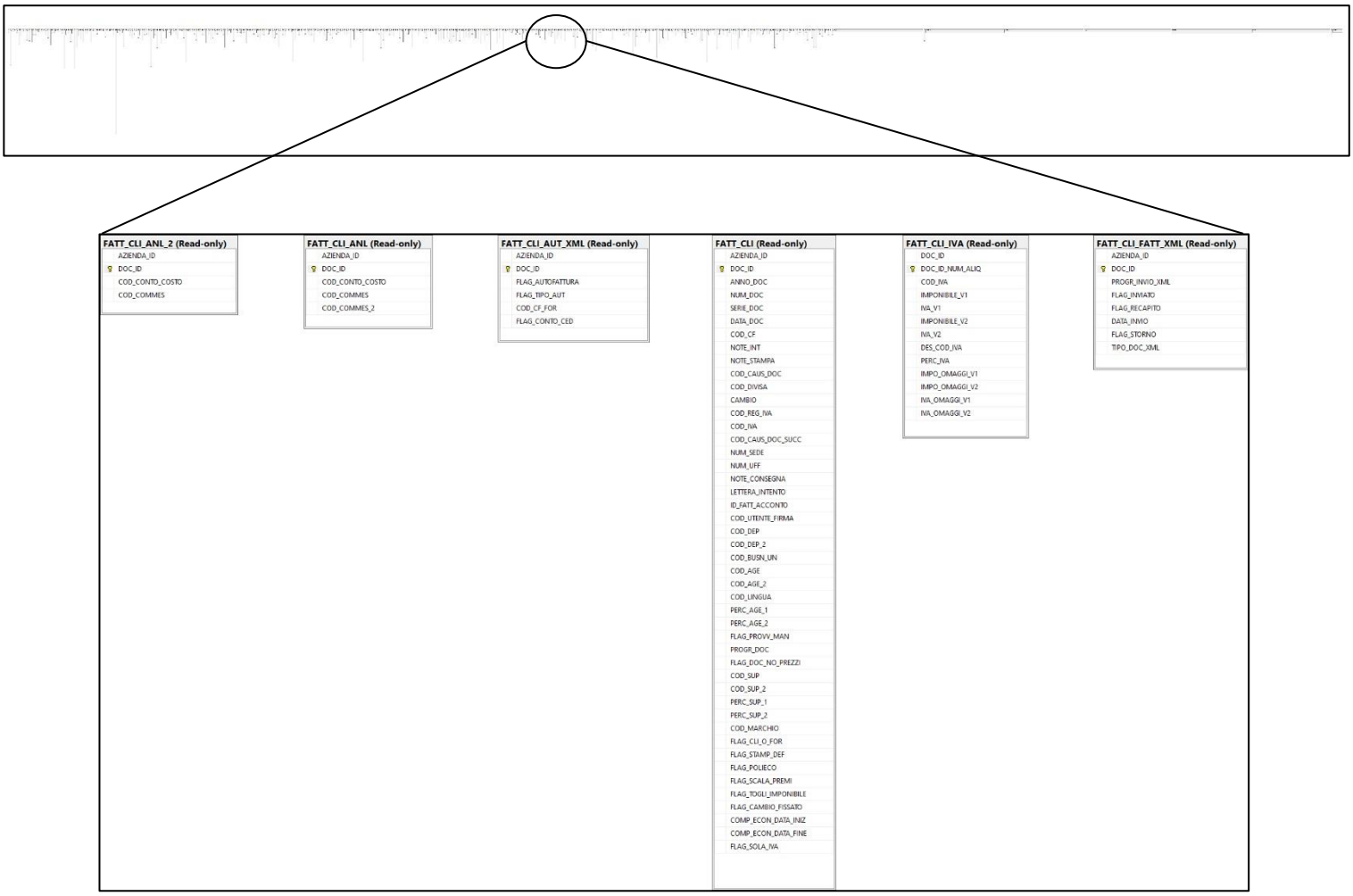

Figura 4 - Database Aziendale

La figura rappresenta il diagramma del database sul quale ho lavorato per effettuare tutte le analisi sui dati.

La struttura è costituita da molteplici entità tutte indipendenti, ovvero senza nessuna relazione.

Ogni tabella è composta da una chiave primaria che è un identificatore auto incrementale.

Per effettuare operazioni di join, necessarie per combinare le tuple di due relazioni, si utilizza un attributo presente in entrambe le entità, come nell'esempio riportato nella pagina successiva.

```
FROM FATT_CLI F
JOIN FATT_CLI_RIGHE  R ON F.DOC_ID = R.DOC_ID
```

In questo caso queste due tabelle “FATT_CLI” e “FATT_CLI_RIGHE” sono relative alle fatture.

La prima contiene tutti attributi generali, come l’importo totale e la data di emissione.

La seconda contiene tutte le righe di una determinata fattura, ovvero tutte le voci, che possono essere relative agli articoli venduti, con le relative quantità e importi.

Quindi, per effettuare un’operazione di join tra queste si utilizza l’attributo “DOC_ID” che, in questo caso, risulta essere l’identificativo della fattura.

Il principale problema di questa struttura del database, prendendo in esame l’esempio in considerazione, è che questo attributo non essendo una Foreign Key, ovvero vincoli che garantiscono l’integrità dei dati e di conseguenza non essendo presente in una tabella distinta, nel caso in cui si voglia eliminare o aggiornare una fattura non si può effettuare un “DELETE/UPDATE CASCADE” ma bisogna effettuare l’operazione di aggiornamento o eliminazione manuale per ogni singola tupla in ogni tabella.

Creando una Foreign Key usando l’opzione “delete cascade” o “update cascade”, automaticamente dopo l’eliminazione o l’aggiornamento della riga di riferimento nella tabella padre, la relativa azione viene propagata alle relative tuple di riferimento.

Un altro problema del mancato utilizzo delle Foreign key è che in ogni tabella, tornando al caso preso in esempio, in cui abbiamo l’attributo “DOC_ID” dobbiamo andare ad inserire le condizioni necessarie al corretto funzionamento del database, come il fatto che sia diverso da NULL.

3.2. SQL Server

SQL Server è un DBMS relazionale o RDBMS (Relational Database Management System) prodotto da Microsoft, ed è in grado di gestire basi di dati di qualsiasi dimensione.

Un RDBMS si basa sul modello relazionale introdotto da Edgar F. Codd nel 1970, con l'espressione presente nel seminario "A Relational Model of Data for Large Shared Data Banks".

Il modello relazionale è basato sul concetto di relazione: "una relazione R su n domini D_1, D_2, …, D_n, non necessariamente distinti, è un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \ldots \times D_n$:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n$$

dove il valore n rappresenta il Grado della relazione, mentre il numero di tuple viene chiamata Cardinalità della relazione". (*Progetto di Basi di Dati Relazionali. Lezioni ed esercizi*)

Ogni RDBMS si basa sul linguaggio SQL (Structured Query Language), quest'ultimo è stato progettato per le seguenti operazioni:

- DDL (Data Definition Language): creare e modificare schemi di database,
- DML (Data Manipulation Language): inserire, modificare e gestire dati memorizzati
- DCL (Data Control Language): creare e gestire strumenti di controllo e accesso ai dati.

SQL nasce nel 1974 grazie a Donald Chamberlin, nei laboratori dell'IBM (International Business Machines Corporation).

Negli anni successivi, si sviluppano nuove versioni e vedendo il successo dei prodotti basati su SQL, diventa lo standard industriale per i software che utilizzano il modello relazionale, la prima a adottarlo come standard è l'ANSI (American National Standards Institute), nel 1986, mentre nel 1987 l'ISO (International Organization for Standardization).

Le query, che sono descritte nel quinto capitolo, sono state sviluppate e testate utilizzando SQL Server.

Per accedere ai dati aziendali senza poter in alcun modo corrompere il database è stato creato un'utente con soli diritti di lettura.

3.3. Forklift

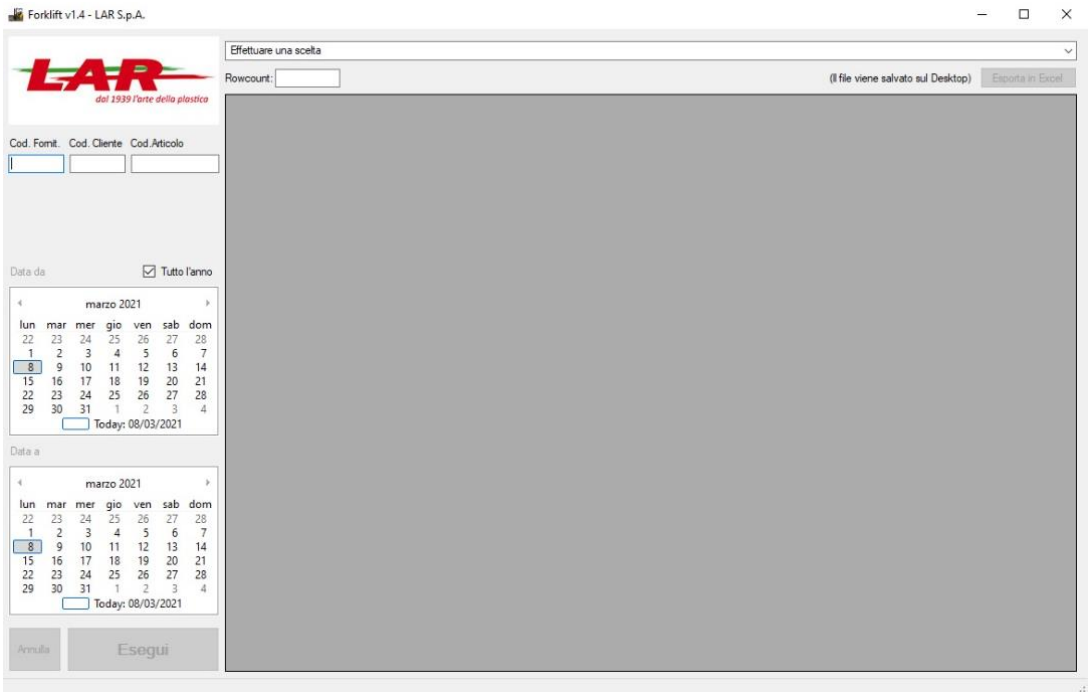

Figura 5 - Schermata iniziale Forklift

Forklift è l'applicazione sviluppata dal responsabile IT di LAR, l'Ing. Maurizio Gentile, utilizzando:

- Visual Studio come ambiente di sviluppo,
- .NET come framework,
- C# come linguaggio di programmazione.

Visual Studio è stato sviluppato per la prima volta da Microsoft nel 1997 ed è un ambiente di sviluppo integrato (Integrated Development Environment o IDE).
Un IDE è un software che supporta i programmatori nello sviluppo e nel debugging del codice del programma, e consiste dei seguenti componenti:

- Editor di codice sorgente,
- Compilatore: dove avviene la traduzione partendo dal codice sorgente,
- Un tool di building automatico,
- Debugger.

Un aspetto importante di Visual Studio è che, dall'introduzione del framework .NET, supporta molteplici linguaggi di programmazione.

Il framework .NET è l'ambiente di esecuzione runtime della omonima piattaforma tecnologica, ed è costituito da Common Language Runtime CLR (fornisce la gestione: dei thread di memoria e processore, delle eccezioni dei programmi e della sicurezza) e librerie di classi.

Il linguaggio C#, inizialmente denominato "Cool", ovvero "C-like Object Oriented Language" (linguaggio orientato agli oggetti simile al C), è stato progettato da Anders Hejsberg, un informatico danese, e dal suo gruppo di lavoro nel 1999.
C# è un linguaggio orientato ad oggetti e consente ai programmatori di compilare molti tipi di applicazioni sicure ed affidabili eseguiti in .NET.

Forklift è l'applicazione che viene distribuita agli utenti e, attraverso questa, sono in grado di eseguire le query sviluppate, selezionandole nel menu a tendina.
Inoltre, è possibile specificare input come: il codice del fornitore, del cliente o dell'articolo (i tre campi di testo presenti sotto il logo della LAR) e la data (selezionabile attraverso i due calendari, oppure selezionando "Tutto l'anno").
Dopo aver specificato i relativi input e aver eseguito la query, l'output prodotto è dato dalle righe interessate (esportabili in Excel, ovvero un documento con estensione xlsx) e il numero di queste.

3.4. Target Cross

Target Cross è l'applicazione utilizzata maggiormente all'interno della LAR, con questa è possibile gestire (modificare, eliminare o aggiungere) le fatture, gli ordini, i DDT, i clienti e i fornitori.

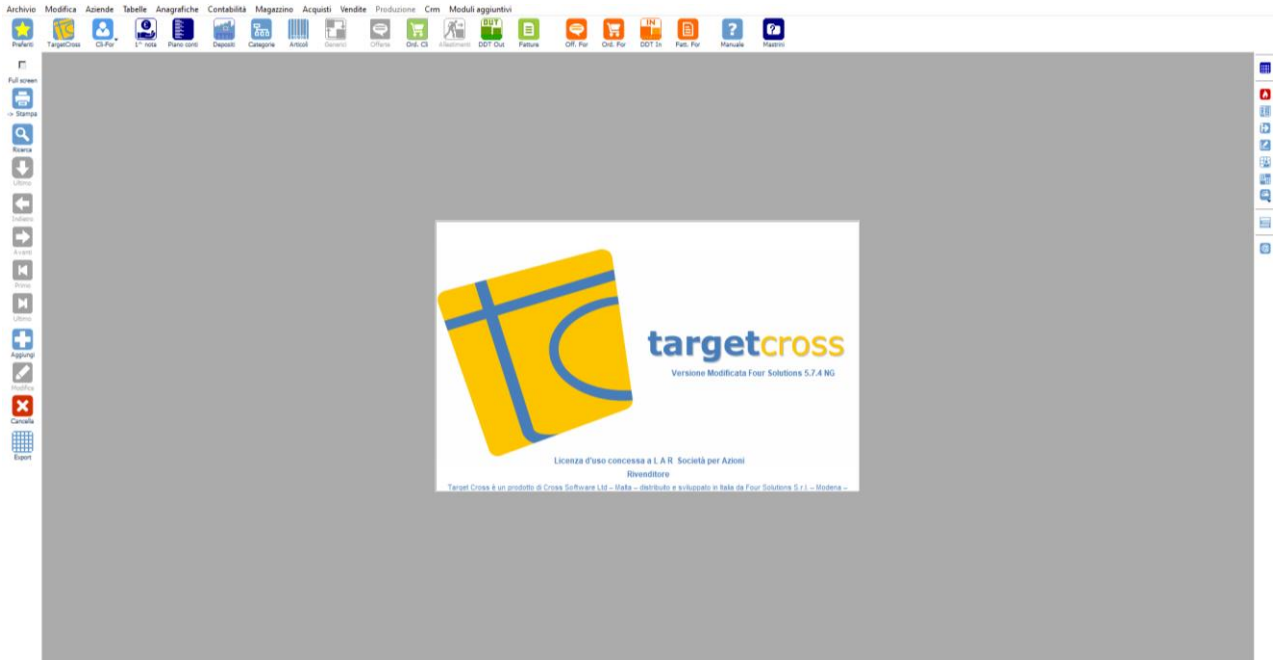

Figura 6 - Schermata principale Target Cross

L'unico requisito per poter utilizzare tutte le funzionalità di Target Cross e per accedere a tutti i dati in esso è di effettuare una connessione al DBMS attraverso l'installazione e la corretta configurazione del driver ODBC (Open DataBase Connectivity).

La connessione data dal ODBC viene effettuata attraverso un'API (Application Programming Interface) standard, che è indipendente dai: linguaggi di programmazione, dai sistemi di database e dal sistema operativo.

Target Cross presenta una sezione per il CRM (Customer Relationship Management), ovvero la gestione delle relazioni con i clienti.

Esistono quattro tipi di CRM:

- CRM operativo: soluzioni per automatizzare i processi di business che prevedono il contatto diretto con il cliente.
- CRM analitico: procedure per migliorare la conoscenza del cliente andando ad analizzare i dati estratti dal CRM operativo.
- CRM collaborativo: metodologie e tecnologie integrate con gli strumenti di comunicazione per gestire il contatto con il cliente.

- CRM strategico: obiettivo di conquistare e mantenere clienti ad alta profittabilità.

L'obiettivo finale del CRM è di rimanere in contatto con i propri clienti in modo continuativo.

I principali vantaggi sono: aumento delle interazioni con i clienti (che comporta un aumento della produttività aziendale), migliore gestione delle vendite, attraverso l'analisi dei dati è possibile effettuare previsioni di vendite e un maggiore grado di fidelizzazione con i clienti stessi.

Questa sezione, relativa al CRM, va a fornire servizi al singolo dipendente della LAR:

- aggiungere eventi all'agenda personale, che vengono salvati nel piano settimanale, dove è possibile visionare tutti gli appuntamenti dei colleghi,
- effettuare campagne di marketing e monitorare tutti gli eventi associati, permettendo di mandare dei testi creati e di gestire delle liste di clienti,
- inviare e gestire le e-mail, con la possibilità di utilizzare tutte le informazioni presenti e registrati nell'applicazione, come i contatti dei clienti o fornitori.

Un'altra sezione presente in Target Cross, inerente all'oggetto della mia tesi, è quella del Data Mining.

In questa parte, è possibile effettuare le query in due modi: manualmente oppure selezionando da un elenco le colonne necessarie.

La query manuale è scritta direttamente dal programmatore e inserita nell'apposito campo, flaggando l'opzione "Crea la SELECT manuale".

Una query segue lo standard SQL, nel quale sono presenti tre campi:

- **Select:** campo nel quale inseriamo i campi che verranno visualizzati in output,
- **From:** campo nel quale specifichiamo le tabelle utilizzate dalla query,
- **Where:** campo nel quale definiamo tutti le condizioni che devono essere soddisfatte.

Il secondo tipo di query viene prodotta seguendo la stessa sintassi standard, ma a differenza della precedente, si deve soltanto selezionare da un elenco le colonne da

inserire nel campo select e creare, in un’apposita sezione, i filtri necessari (riconducibili alle condizioni del campo where).

4. Analisi dei dati in LAR

4.1. Descrizione generale

Le query prodotte durante il mio tirocinio curricolare presso LAR, che vengono analizzati nello specifico nel capitolo successivo, sono relative a diverse aree di attività presenti nell'azienda:

- **commerciale,**
- **amministrativa,**
- **produzione,**
- **logistica.**

AREA COMMERCIALE

Sono stati prodotti vari esperimenti di data mining volti, ad esempio, ad analizzare le vendite dei vari articoli, principalmente in termini di quantità, con la possibilità di selezionare determinati periodi temporali.

Questa query può essere utile per effettuare indagini di mercato, dal momento che è possibile a livello visivo e di dati, capire quale prodotto è acquistato maggiormente, e di conseguenza, quale articolo prediligere per effettuare campagne pubblicitarie e per aumentare il numero di clienti.

Un altro esempio utilizzato in ambito commerciale per effettuare analisi di mercato (anche queste specificando dei periodi temporali), può essere relativo ai clienti che hanno acquistato un determinato articolo nel periodo.

Utilizzando questo, è possibile scoprire in un determinato momento dove, in termini di provincia e paese, è stato venduto maggiormente un singolo prodotto o tutti.

Un altro utilizzo di questa query è la creazione di e-mail personalizzate per ogni cliente; attraverso i dati estratti, possono essere inviati dei "reminder", ovvero dei promemoria, per esempio, ricordando che nello stesso periodo dell'anno scorso sono stati acquistati determinati prodotti, con le relative quantità e se si ha la volontà di procedere con lo stesso ordine.

Un'altra query prodotta è relativa alle fatture per variazione del prezzo dell'articolo, questo coinvolge sia l'area commerciale sia quella amministrativa, quest'ultima verrà analizzata successivamente.
Nel primo settore, può essere utilizzata per analizzare gli andamenti delle vendite del prodotto nel caso in cui questo abbia delle maggiorazioni di prezzo.

Infine, l'ultima query prodotta è relativa ai clienti basso-acquistanti (o clienti persi), ovvero quelli che dopo un determinato anno o periodo hanno smesso di acquistare prodotti dell'azienda.
In questo caso, attraverso l'estrazione dal database di tutti i dati è possibile effettuare azioni con lo scopo di far tornare questi soggetti dei clienti.

AREA AMMINISTRATIVA

Come specificato precedentemente la query relativa alle fatture per variazione del prezzo dell'articolo può essere utilizzato anche in ambito amministrativo, dal momento che, sapendo il prezzo del prodotto, consente una ricerca più veloce, senza dover cercare ogni singola fattura inserita su Target Cross.

Dalla query "Previsione Esborsi" si ottengono, specificando un periodo di riferimento, tutti gli esborsi dell'azienda, ovvero i pagamenti da effettuare ai fornitori.
Può essere utilizzato a livello economico, per previsioni future e per analisi sull'andamento dell'impresa.

Infine, l'ultima analisi per quest'area è: "Totale fatturato in periodo per cliente", questo permette di analizzare per ogni cliente, date le fatture emesse, l'importo che entrerà nelle casse aziendali.

AREA PRODUZIONE

Considerando la query volta ad analizzare i clienti che hanno acquistato un determinato articolo nel periodo.

Come è stato spiegato nell'"area commerciale" attraverso l'utilizzo di questa query possiamo creare delle e-mail.
Queste hanno una doppia funzione da una parte quella commerciale analizzata precedentemente, mentre la seconda riguarda quest'area, dal momento che inviando questi reminder prima rispetto al periodo di riferimento, nel caso in cui il destinatario sia intenzionato a procedere con l'ordine si può organizzare la produzione con anticipo.

La stessa funzione specificata precedentemente, ovvero la possibilità di organizzare la produzione in modo efficiente, può essere ricondotta alle query: "quantità prodotta ad otto settimane da oggi" e "product order DKT settimanale/mensile".
Il primo ha una visione generale di tutti i prodotti indipendentemente dal cliente, mentre il secondo va ad analizzare solamente gli ordini di un determinato cliente (DKT).

AREA LOGISTICA

Le ultime due query analizzate nell'area produttiva, ovvero "quantità prodotta ad otto settimane da oggi" e "product order DKT settimanale/mensile", possono essere utilizzate anche dall'area logistica, per organizzare tutta la gestione di queste, come ad esempio lo stoccaggio dei vari prodotti all'interno dei magazzini presenti nell'azienda.

4.2. Tecniche utilizzate

CASE Statement

Valuta un elenco di condizioni e ritorna un'espressione di risultato.

L'espressione CASE può essere sviluppata secondo due formati:

1) CASE semplice: confronta l'espressione di input con l'espressione presente in ogni clausola WHEN per determinare l'equivalenza, quando si verifica questa condizione viene restituito il risultato nella clausola THEN.

```
CASE input_expression
      WHEN when_expression THEN result_expression [ ...n ]
      [ ELSE else_result_expression ]
END
```

2) CASE avanzato: risultato ottenuto valutando un set di espressioni booleane.

```
CASE
      WHEN Boolean_expression THEN result_expression [ ...n ]
      [ ELSE else_result_expression ]
END
```

REPLACE()

La funzione REPLACE() serve a sostituire tutte le occorrenze di una determinata substring all'interno di una stringa, con una nuova substring.

```
REPLACE ( string_expression , string_pattern , string_replacement )
```

CAST()

La funzione CAST() serve per effettuare conversioni di un'espressione da un tipo ad un altro.

```
CAST ( expression AS data_type [ ( lenght ) ] )
```

SUM()

La funzione SUM() restituisce la somma di tutti i valori (ALL) o dei valori distinti (DISTINCT) dell'espressione, ma solo per colonne numeriche.

```
SUM ( [ ( ALL | DISTINCT ) ] expression )
```

OVER([PARTITION BY])

Utile per il partizionamento ed eventuale ordinamento di un set di righe prima dell'applicazione della funzione finestra associata.

Utilizzando la clausola OVER e PARTITION BY raggruppiamo i dati secondo un set di righe specificato dall'utente, in un secondo momento, una funzione calcola un valore per ogni riga.

```
OVER ( [  <PARTITION BY clause> ]
       [ <ORDER BY clause> ]
      [ <ROW or RANGE clause> ]
                )
```

I campi opzionali sono:

- **PARTITION BY:** suddivide il set di risultati della query in partizioni.
- **ORDER BY:** definisce l'ordine logico delle righe all'interno di ogni partizione del set di risultati.
- **ROWS | RANGE:** limita le righe all'interno della partizione specificando i punti iniziali e finali.

Questa clausola può essere ricondotta al GROUP BY, dal momento che tutte e due vanno a raggruppare le righe, ma la differenza sostanziale è che l'OVER va a mantenere tutte le righe che andiamo a processare, mentre il GROUP BY va a raggruppare delle righe in base agli attributi che specifichiamo.

id	firstname	lastname	Mark
1	arun	prasanth	40
2	ann	antony	45
3	sruthy	abc	41
6	new	abc	47
1	arun	prasanth	45
1	arun	prasanth	49
2	ann	antony	49

Figura 7 - Tabella di esempio

Figura 8 - Risultato Query con OVER

```
marksum firstname
-------------------
134     arun
134     arun
134     arun
94      ann
94      ann
41      sruthy
47      new
```

```
marksum  firstname
----------------
94       ann
134      arun
47       new
41       sruthy
```

Figura 9 - Risultato Query con GROUP BY

DATEADD()

La funzione DATEADD() serve ad aggiungere a un valore date in input, un numero con la relativa "unità di misura" (year, month, day, week, …).

```
DATEADD ( datepart , number , date )
```

Gli argomenti di questa funzione sono:

- **datepart:** Parte di date a cui DATEADD() aggiunge il valore number.
- **number:** Valore intero con segno.
- **Date.**

WITH

WITH indica un set di risultati temporaneo, denominato Common Table Expression CTE (espressione di tabella comune).

Le righe che descriveranno questa CTE saranno selezionate attraverso la CTE_query_definition.

```
WITH <common_table_expression> expression_name [ ( column_name [ ,...n ] ) ]
                        AS ( CTE_query_definition )
```

Gli argomenti sono:

- **Expression_name**: identificatore della CTE.
- **Column_name:** serve a specificare un nome di colonna nella CTE. Il numero dei column_name deve essere uguale a quello degli attributi presenti nel campo SELECT della CTE_query_definition.

- **CTE_query_definition:** specifica un'istruzione SELECT che ha l'obiettivo di popolare con i propri risultati la CTE.

MAX()

Restituisce il valore massimo dell'espressione.

```
MAX ( [ ( ALL | DISTINCT ) ] expression )
```

JOIN

Il JOIN serve per combinare le tuple di due o più relazioni, in base a una colonna correlata tra di loro.

Le possibili tipologie sono:

- **INNER JOIN**: è il join di default e attraverso questo tipo vengono selezionati sono i valori corrispondenti in entrambe le tabelle (in base alla condizione inserita).
- **LEFT [OUTER] JOIN**: si mantengono anche le tuple della first_table che non hanno corrispondenza nella second_table, queste vengono anche chiamate "tuple di dangling".
- **RIGHT [OUTER] JOIN**: si mantengono mantenute anche le tuple della second_table che non hanno corrispondenza nella first_table.
- **FULL [OUTER] JOIN**: si mantengono tutte le tuple di dangling di ogni table che non hanno corrispondenza con l'altra tabella.
- **CROSS JOIN**: può essere definito anche come prodotto cartesiano e produce un set di risultati che è dato dal numero di righe della first_table moltiplicato per quelle della second_table (se non vengono inserite clausole nel WHERE).
- **SELF JOIN**: un join dove la first_table è unita a se stessa, di conseguenza di dovrà utilizzare gli alias.

```
FROM first_table < join_type> second_table [ ON ( join_condition ) ]
```

DATEPART()

La funzione DATEPART() restituisce un intero che rappresenta una parte della data fornita in input.

```
DATEADD ( datepart , date )
```

ISNUMERIC()

Determina se l'espressione fornita alla funzione è un tipo numerico valido.

```
ISNUMERIC ( expression )
```

GETDATE()

Restituisce il valore della data in cui viene chiamata la funzione.

```
GETDATE ()
```

5. Query per l'analisi dei dati svolte

Attraverso l'utilizzo degli strumenti e delle tecniche, descritti nei precedenti capitoli, è stato possibile realizzare le seguenti query per effettuare analisi sui dati presenti nel database aziendale.

Questi vengono analizzati specificando l'obiettivo (o lo scopo finale) e il campo d'applicazione, ovvero in che area aziendale possiamo collocarli, ed elencando gli input, gli output attesi, le tabelle utilizzate e le funzioni/statement.

5.1. Query 1 - Clienti che hanno acquistato un determinato articolo nel periodo

```
SELECT DISTINCT
        R.COD_ART AS 'Codice articolo',
        REPLACE(A.DES_ART, CHAR(164), '') AS 'Descrizione articolo',
        CASE WHEN LEN(C.LIV_6_CAT) > 0 THEN C.DESC_L6
        WHEN LEN(C.LIV_5_CAT) > 0 THEN C.DESC_L5
        WHEN LEN(C.LIV_4_CAT) > 0 THEN C.DESC_L4
        WHEN LEN(C.LIV_3_CAT) > 0 THEN C.DESC_L3
        WHEN LEN(C.LIV_2_CAT) > 0 THEN C.DESC_L2
        WHEN LEN(C.LIV_1_CAT) > 0 THEN C.DESC_L1 ELSE ''
        END AS 'Categoria articolo',
        CF2.COD_AGE1_CLI AS 'Codice agente',
        AG.NOME_AGE AS 'Descrizione agente',
        O.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        CF.TEL_CF AS 'Telefono',
        CF.E_MAIL_CF AS 'Email',
        CF.PROVINCIA_CF AS 'Provincia'
        CF.STATO_CF AS 'Stato',
        CAST(SUM(R.QUANT_RIGA) OVER(PARTITION BY (R.COD_ART + O.COD_CF))
        AS INT) AS 'Quantità ordinata'

FROM ORD_CLI O
        LEFT JOIN ORD_CLI_RIGHE R ON O.DOC_ID = R.DOC_ID
        LEFT JOIN ART_ANA A ON A.COD_ART = R.COD_ART
        LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT
        LEFT JOIN CF_CLI_4B CF2 ON CF2.COD_CF = O.COD_CF AND CF2.AZIENDA_ID =
        O.AZIENDA_ID
        LEFT JOIN AGENTI AG ON AG.COD_AGE = CF2.COD_AGE1_CLI AND
        AG.AZIENDA_ID = O.AZIENDA_ID
```

```
        JOIN CF ON CF.COD_CF = O.COD_CF

WHERE O.AZIENDA_ID = 1
        AND O.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM + "'
        -- fromDateYDM e toDateYDM sono due variabili relative alla data

        if (itemCode != string.Empty) -- this is taken by Visual Studio code and there is a
                AND R.COD_ART LIKE '" + itemCode + "%' -- check with an if statement
        AND A.COD_CAT NOT LIKE 'VIRT%'  -- excludes virtual item
        AND A.COD_CAT NOT LIKE 'ZDG%' -- exclude other non-items

ORDER BY O.COD_CF
```

Obiettivo: Trovare tutti i clienti che hanno acquistato un determinato prodotto in un determinato periodo, oppure lasciando vuoto il campo articolo trovare, dato un periodo, tutti i clienti che hanno acquistato dei prodotti e quali di questi.

Campo d'applicazione: Commerciale e produzione.

Input: Data da, Data a e [Cod.Articolo].

Output: Codice articolo, Descrizione articolo, Categoria articolo, Codice agente, Descrizione agente, Codice Cliente, Descrizione Cliente, Telefono, Email, Provincia, Stato e Quantità ordinata.

Tabelle di partenza: ORD_CLI (ordini effettuati dai clienti), ORD_CLI_RIGHE (tutte le singole voci presenti negli ordini dei clienti), ART_ANA (anagrafica degli articoli con tutte le relative informazioni dettagliate), CAT_MERCE (categoria della merce), CF_CLI_4B (tutti i dati del personale della LAR), AGENTI (i responsabili che hanno gestito gli ordini) e CF (tutti i dati dei clienti e dei fornitori).

Funzioni/Statement utilizzati:

- CASE Statement: per determinare in quale attributo si trova la categoria.
- REPLACE(): per diminuire, fino ad un massimo di 164B la descrizione dell'articolo.
- CAST(): conversione del valore dato dalla SUM in INT.
- SUM(): somma delle quantità delle righe con lo stesso prodotto.
- OVER([PARTITION BY]): determina il partizionamento delle righe rispetto al codice dell'articolo COD_ART e il codice del cliente COD_CF.

5.2. Query 2 - Totale fatturato in periodo per cliente

```
SELECT DISTINCT
        F.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        CF.P_IVA_CF AS 'Partiva IVA',
        CF.TEL_CF AS 'Telefono',
        CF.E_MAIL_CF AS 'Email',
        CF.INDI_CF as 'Indirizzo',
        CF.CAP_CF as 'Cap',
        CF.COMUNE_CF as 'Comune',
        CF.PROVINCIA_CF as 'Provincia',
        CF.STATO_CF as 'Stato',
        CAST(SUM(T.TOTALE_V1) OVER(PARTITION BY F.COD_CF) AS DECIMAL(10, 3))
        AS 'Totale fatturato'

FROM FATT_CLI F
        LEFT JOIN FATT_CLI_TOT T ON T.DOC_ID = F.DOC_ID
        JOIN CF ON CF.COD_CF = F.COD_CF

WHERE F.AZIENDA_ID = 1
        AND F.COD_CAUS_DOC LIKE 'FAT[_]_[_]_' -- the _ is the placeholder for a single char,
        -- we use[_] to escape it as we want to search for FATT_singlechar_C
        AND F.DATA_DOC BETWEEN ''' + fromDateYDM + ''' AND ''' + toDateYDM + '''

ORDER BY 'Totale fatturato' DESC
```

Obiettivo: Conoscenza dell'importo totale, ovvero la somma delle fatture, di ogni singolo cliente dato un periodo scelto a priori dall'utente.

Campo d'applicazione: Amministrativo.

Input: Data da e Data a.

Output: Codice cliente, Descrizione cliente, P.IVA, Telefono, E-mail, Indirizzo, Cap, Comune, Provincia, Stato e Totale fatturato.

Tabelle di partenza: FATT_CLI (fatture dei clienti), FATT_CLI_TOT (fatture totali dei clienti) e CF (tutti i dati dei clienti).

Funzioni/Statement utilizzati: CAST(), SUM() e OVER ([PARTITION BY]).

5.3. Query 3 - Quantità prodotta ad 8 settimane da oggi

```
SELECT DISTINCT
        CF.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        OP.COD_ART AS 'Codice articolo LAR',
        AC.COD_SECONDARIO_ART AS 'Codice articolo DKT',
        OP.DES_PROD AS 'Descrizione articolo',
        OP.UM AS 'Unità di misura',
        SUM(OP.QUANT_DA_PROD) OVER(PARTITION BY OP.COD_ART) AS 'Qta prodotta
        a + 8 weeks',
        DATEADD(WEEK, 8, GETDATE()) AS 'Data di fine prevista'

FROM ORP_EFF OP
        JOIN CF ON CF.COD_CF = OP.COD_CF
        LEFT JOIN ART_CODICI AC ON AC.COD_ART = OP.COD_ART
        LEFT JOIN ART_ANA A ON A.COD_ART = AC.COD_ART

WHERE OP.AZIENDA_ID = 1
        AND OP.COD_CF = '000009472'
        AND OP.DATA_FINE_PREVISTA <= DATEADD(WEEK, 8, GETDATE())

ORDER BY 4
```

Obiettivo: Verificare la quantità per ogni articolo da produrre o già prodotta, relativa agli ordini con scadenza entro le due mensilità dalla data in cui si esegue questa query.

Campo d'applicazione: Logistica e produzione.

Input: -

Output: Codice cliente, Descrizione cliente, Codice articolo LAR, Codice articolo DKT, Descrizione articolo, Unità di misura, Q.ta prodotta a + 8 weeks e Data di fine prevista.

Tabelle di partenza: ORP_EFF (ordini di produzione effettivi), CF, ART_CODICI (codici degli articoli) e ART_ANA (tutti gli attributi degli articoli).

Funzioni/Statement utilizzati:

- SUM(),
- OVER ([PARTITION BY]).

- DATEADD(): con questa funzione andiamo a sommare le 8 settimane al giorno in cui si esegue questa query.
- GETDATE(): restituisce il valore della data in cui si esegue la query.

5.4. Query 4 - Clienti basso-acquistanti

```
WITH CTE_DateUltimeFatture(COD_CF, UltimaDataAcquisto) AS
(
        SELECT DISTINCT
                F.COD_CF,
                MAX(F.DATA_DOC) OVER(PARTITION BY F.COD_CF) AS
                UltimaDataAcquisto
        FROM FATT_CLI F
        WHERE F.AZIENDA_ID = 1
                AND F.COD_CAUS_DOC LIKE 'FAT[_]_[_]C' -- the _ is the placeholder for a
                single char, we use[_] to escape it as we want to search for FATT_singlechar_C
                AND F.DATA_DOC < '" + toDateYDM + "'
)

SELECT
        F.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        CF.P_IVA_CF AS 'Partiva IVA',
        CF.TEL_CF AS 'Telefono',
        CF.E_MAIL_CF AS 'Email',
        R.COD_ART AS 'Codice articolo',
        REPLACE(A.DES_ART, CHAR(164), '') AS 'Descrizione articolo',
        C.LIV_1_CAT AS 'Categoria',
        CAST(R.PREZZO_LORDO_VU1 AS DECIMAL(10, 3)) AS 'Prezzo d''acquisto',
        F.DATA_DOC AS 'Ultima data d''acquisto'
 FROM FATT_CLI F
        JOIN CTE_DateUltimeFatture DUF ON DUF.COD_CF = F.COD_CF AND
        DUF.UltimaDataAcquisto = F.DATA_DOC
        JOIN CF ON CF.COD_CF = F.COD_CF
        LEFT JOIN FATT_CLI_RIGHE R ON F.DOC_ID = R.DOC_ID
        LEFT JOIN ART_ANA A ON A.COD_ART = R.COD_ART
        LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT

WHERE F.AZIENDA_ID = 1

        if (itemCode != string.Empty)
                AND R.COD_ART LIKE '" + itemCode + "%'

        AND A.COD_CAT NOT LIKE 'VIRT%'   --excludes virtual items
        AND A.COD_CAT NOT LIKE 'ZDG%'    --exclude other non-items

ORDER BY F.DATA_DOC, F.COD_CF ASC
```

Obiettivo: Verificare i clienti persi, valutando la data in cui hanno acquistato per l'ultima volta un prodotto dall'azienda.

Campo d'applicazione: Commerciale.

Input: Data a e [Cod.Articolo]

Output: Codice cliente, Descrizione cliente, P.IVA, Telefono, E-mail, Codice articolo, Descrizione articolo, Categoria, Prezzo d'acquisto e Ultima data d'acquisto.

Tabelle di partenza: FATT_CLI, CTE_DataUltimeFatture (tabella temporanea che ha come scopo di estrarre per ogni cliente la data dell'ultima fattura), CF, FATT_CLI_RIGHE, ART_ANA e CAT_MERCE.

Funzioni/Statement utilizzati:

- WITH: utilizzato per la creazione di una tabella temporanea dove ad ogni codice del cliente si associa l'ultima data dell'ultima fattura prima di un valore limite temporale "toDateYDM".
- REPLACE()
- CAST Statement
- MAX(): funzione inserita con lo scopo di trovare la data relativa all'ultima fattura emessa per ogni cliente.

5.5. Query 5 - Quantità venduta per articolo

```
WITH CTE_Sum(CodiceArticolo, DescrizioneArticolo, Categoria, Qty) AS
(
	SELECT DISTINCT
		R.COD_ART AS CodiceArticolo,
		REPLACE(A.DES_ART, CHAR(164), '') AS DescrizioneArticolo,
		CASE WHEN LEN(C.LIV_6_CAT) > 0 THEN C.LIV_6_CAT
		WHEN LEN(C.LIV_5_CAT) > 0 THEN C.LIV_5_CAT
		WHEN LEN(C.LIV_4_CAT) > 0 THEN C.LIV_4_CAT
		WHEN LEN(C.LIV_3_CAT) > 0 THEN C.LIV_3_CAT
		WHEN LEN(C.LIV_2_CAT) > 0 THEN C.LIV_2_CAT
		WHEN LEN(C.LIV_1_CAT) > 0 THEN C.LIV_1_CAT ELSE ''
		END AS Categoria,
		CAST(SUM(R.QUANT_RIGA) OVER(PARTITION BY R.COD_ART) AS INT)
		AS Qty

	FROM FATT_CLI_RIGHE R
		JOIN ART_ANA A ON A.COD_ART = R.COD_ART
		LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT

	WHERE A.COD_CAT NOT LIKE 'VIRT%' --excludes virtual items
		AND A.COD_CAT NOT LIKE 'ZDG%' --exclude other non-items
		AND R.DATA_DOC BETWEEN ''' + fromDateYDM + ''' AND ''' + toDateYDM +
		''' -- The BETWEEN operator is inclusive: begin and end values are included.

		if (itemCode != string.Empty)
			AND R.COD_ART LIKE ''' + itemCode + '%'
)


SELECT
	S.CodiceArticolo AS 'Codice articolo',
	S.DescrizioneArticolo AS 'Descrizione articolo',
	S.Categoria AS 'Categoria',
	S.Qty AS 'Quantità'

FROM CTE_Sum S

ORDER BY S.Qty DESC
```

Obiettivo: Verificare la quantità venduta di un articolo o di tutti in un determinato periodo, potendo in un successivo momento effettuare analisi di mercato e di vendita.

Campo d'applicazione: Commerciale.

Input: Data da, Data a e [Cod.Articolo].

Output: Codice Articolo, Descrizione Articolo, Categoria e Qty.

Tabelle di partenza: CTE_Sum (tabella temporanea al cui interno per ogni articolo si sommano le quantità vendute in un determinato periodo), FATT_CLI_RIGHE, ART_ANA e CAT_MERCE.

Funzioni/Statement utilizzati: WITH, REPLACE(), CASE Statement, CAST(), SUM() e OVER ([PARTITION BY]).

5.6. Query 6 - Previsione esborsi

```
SELECT
	CF.COD_CF AS 'Codice fornitore',
	 CF.RAG_SOC_CF AS 'Descrizione fornitore',
	O.NUM_DOC AS 'Numero ordine',
	SUM(R.IMPORTO_V1) AS 'Totale imponibile',
	RS.DATA_CONS_RIGA AS 'Data consegna',
	C.DES_PAGA AS 'Modalità di pagamento',
	'' AS 'Scadenza pagamento fattura'


 FROM ORD_FOR O
	JOIN CF ON CF.COD_CF = O.COD_CF AND CF.FLAG_FOR = 1
	LEFT JOIN ORD_FOR_RIGHE R ON R.DOC_ID = O.DOC_ID
	LEFT JOIN ORD_FOR_PAG P ON P.DOC_ID = O.DOC_ID
	LEFT JOIN CONDPA C ON C.COD_PAGA = P.COD_PAGA
	LEFT JOIN ORD_FOR_RIGHE_SPEC RS ON RS.DOC_RIGA_ID = R.DOC_RIGA_ID


WHERE O.AZIENDA_ID = 1
	AND RS.DATA_CONS_RIGA BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM
	+ "' -- yyyyddmm - The BETWEEN operator is inclusive: begin and end values are
	included.


GROUP BY CF.COD_CF, CF.RAG_SOC_CF, O.NUM_DOC, C.DES_PAGA,
RS.DATA_CONS_RIGA, R.IMPORTO_V1


ORDER BY O.NUM_DOC
```

Obiettivo: Analizzare dato un periodo di tempo scelto dall'utente gli esborsi aziendali.

Campo d'applicazione: Amministrativo.

Input: Data da e Data a

Output: Codice fornitore, Descrizione fornitore, Numero ordine, Totale imponibile, Data consegna, Modalità di pagamento e Scadenza pagamento fattura.

Tabelle di partenza: ORD_FOR (ordini ai fornitori), CF, ORD_FOR_RIGHE (tutte le singole voci presenti negli ordini ai fornitori), ORD_FOR_PAG (pagamenti ai fornitori), CONDPA (condizioni di pagamento) e ORD_FOR_RIGHE_SPEC (come ORD_FOR_RIGHE ma specifiche).
Funzioni/Statement utilizzati: SUM().

5.7. Query 7 - Product Order DKT settimanale/mensile

```
SELECT
        CF.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        O.DATA_DOC AS 'Data ordine',
        O.NUM_DOC AS 'Numero ordine LAR',
        OS.NUM_ORDINE_CLIENTE AS 'Numero ordine PO DKT',
        R.COD_ART AS 'Codice articolo LAR',
        AC.COD_SECONDARIO_ART AS 'Codice articolo DKT',
        CAST(R.PREZZO_NETTO_VU1 AS DECIMAL(10, 3)) AS 'Importo EXW',
        R.QUANT_RIGA AS 'Quantità',
        R.UM AS 'UM',
        CAST(SUM(R.PREZZO_NETTO_VU1*R.QUANT_RIGA) OVER(PARTITION BY
        R.COD_ART) AS DECIMAL(10,3)) AS 'Importo totale mensile',
        RS.DATA_CONS_RICH_RIGA AS 'Data consegna',
        DATEPART(MONTH, RS.DATA_CONS_RICH_RIGA) AS 'Mese'

        --DATEPART(WEEK, RS.DATA_CONS_RICH_RIGA) AS 'Settimana consegna'
        -- if we want this query weekly and not monthly

FROM ORD_CLI O
        JOIN CF ON CF.COD_CF = O.COD_CF
        LEFT JOIN ORD_CLI_RIGHE R ON R.DOC_ID = O.DOC_ID
        LEFT JOIN ORD_CLI_RIGHE_SPEC RS ON RS.DOC_RIGA_ID = R.DOC_RIGA_ID
        LEFT JOIN ORD_CLI_SPEC OS ON OS.DOC_ID = O.DOC_ID
        LEFT JOIN ART_CODICI AC ON AC.COD_ART = R.COD_ART
        LEFT JOIN ART_ANA A ON A.COD_ART = AC.COD_ART

WHERE O.COD_CF = '000009472'
        AND R.COD_ART != 'V1' -- excludes unwanted items
        AND O.DATA_DOC >= '" + fromDateYDM + "'   -- used for DataOrdine
        AND RS.DATA_CONS_RICH_RIGA <= '" + toDateYDM + "' -- used for DataConsegna
        AND 1 = ISNUMERIC(OS.NUM_ORDINE_CLIENTE)  -- excludes
        unwanted/incomplete/canceled orders which are usually prefixed by !!, --, **, or similar

ORDER BY RS.DATA_CONS_RICH_RIGA
```

Obiettivo: Determinare e analizzare il numero di prodotti ordinati mensilmente o settimanalmente da parte del cliente DKT. Utile per effettuare previsioni a livello produttivo e anche logistico.

Campo d'applicazione: Produzione e logistica.

Input: Data da e Data a.

Output: Codice cliente, Descrizione cliente, Data ordine, Numero ordine LAR, Numero ordine PO DKT, Codice articolo LAR, Codice articolo DKT, Importo EXW, Quantità, UM, Importo totale mensile, Data consegna e Mese (o settimana) consegna.

Tabelle di partenza: ORD_CLI, ORD_CLI_RIGHE, ORD_CLI_RIGHE_SPEC, ORD_CLI_SPEC, ART_CODICI E ART_ANA.

Funzioni/Statement utilizzati:

- CAST().
- SUM().
- OVER ([PARTITION BY]).
- DATEPART(): per ritornare solamente il mese o la settimana.
- ISNUMERIC(): per escludere ordini non voluti, incompleti o cancellati che hanno prefissi non numerici.

5.8. Query 8 - Fatture per variazione prezzo articolo

```
SELECT
        R.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        F.NUM_DOC AS 'Numero fattura',
        R.DOC_ID AS 'N. completo',
        R.NUM_RIGA AS 'Riga fattura',
        R.COD_ART AS 'Codice articolo',
        R.DES_RIGA AS 'Descrizione articolo',
        CAST(R.PREZZO_LORDO_VU1 AS DECIMAL(10,3)) AS 'Prezzo unitario',
        R.UM_BASE AS 'UM base',
        R.QUANT_UM_BASE AS 'Quantità base',
        R.UM AS 'UM',
        R.QUANT_RIGA AS 'Quantità',
        CAST(R.IMPORTO_V1 AS DECIMAL(10,2)) AS 'Importo'

FROM FATT_CLI F
        JOIN FATT_CLI_RIGHE R ON F.DOC_ID = R.DOC_ID
        LEFT JOIN CF ON CF.COD_CF = F.COD_CF

WHERE F.AZIENDA_ID = 1
        AND F.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM + "' -- The
        BETWEEN operator is inclusive: begin and end values are included.

        if (customerCode != string.Empty)
                AND F.COD_CF = '" + customerCode + "'

        if (itemCode != string.Empty)
                AND R.COD_ART LIKE '" + itemCode + "%'

ORDER BY R.COD_CF, F.NUM_DOC, R.NUM_RIGA
```

Obiettivo: Visualizzare le possibili variazioni del prezzo di un articolo, per tutti i clienti o per uno singolo.

Questa query può essere effettuata per analizzare il numero di quantità vendute, in base al prezzo per un determinato articolo.

Campo d'applicazione: Amministrativo e commerciale.

Input: Data da, Data da, [Cod.Articolo] e [Cod. Cliente].

Output: Codice cliente, Descrizione cliente, Numero fattura, N. completo, Riga fattura, Codice articolo, Descrizione articolo, Prezzo unitario, UM base, Quantità base, UM, Quantità e Importo.

Tabelle di partenza: FATT_CLI, FATT_CLI_RIGHE e CF.

Funzioni/Statement utilizzati: CAST().

6. Conclusioni

Nell'ambito della presente tesi è stato descritto l'oggetto principale del mio tirocinio curricolare svolto in LAR, ovvero il data mining.

Dopo un'introduzione sull'azienda e sulle attività svolte, è stata presentata l'evoluzione storica di questo insieme di tecniche e metodologie (analizzando parallelamente il progresso delle basi di dati), fornendo anche la definizione e le principali tecniche. In seguito, è stato descritto il processo Knowledge Discovery in Database" KDD (o processo di estrazione di conoscenza di dati), dove una delle fasi è il data mining.

Successivamente, è stata presentata la seconda parte della tesi, ovvero l'applicazione diretta del data mining.
Sono stati illustrati gli strumenti utilizzati e che sono serviti per lo sviluppo dell'oggetto della tesi, analizzando anche una problematica del database aziendale, ovvero la mancanza delle relazioni tra le entità, descrivendo tutti gli effetti negativi che ne derivano.
Successivamente, sono state definite le tecniche utilizzate all'interno delle query e sono state descritte le aree aziendali che sono state interessate dalla creazione dei data mining, analizzando nello specifico l'efficacia e l'applicazione.
L'ultimo argomento trattato all'interno della mia tesi riguarda l'illustrazione delle query prodotte, effettuando anche una descrizione specifica, attraverso l'elencazione degli obiettivi, degli input/output, dei campi di applicazione, delle tabelle di partenza e, infine, delle funzioni e degli Statement utilizzati.

La produzione delle query per l'analisi dei dati analizzate è stata richiesta direttamente dagli utenti per migliorare la ricerca, ad esempio, delle fatture o per effettuare analisi di mercato in modo diretto senza dover ricercare manualmente i dati necessari.

Di conseguenza, gli effetti immediati sono stati: maggiore facilità nell'ottenimento delle dei dati e, anche, una diminuzione delle risorse temporali richieste per ottenere lo stesso risultato senza i data mining.

"Oggi solo l'1% dei dati raccolti viene utilizzato dalle imprese, che potrebbero invece ottenere vantaggi a partire dal "machine learning", dalle macchine cioè che perfezionano la loro resa "imparando" dai dati via via raccolti e analizzati." (https://www.economyup.it/innovazione/cos-e-l-industria-40-e-perche-e-importante-saperla-affrontare/)

Una possibile soluzione che può sfociare in uno scenario futuro aziendale può riguardare l'introduzione del Manufacturing Execution System MES (sistema di esecuzione della produzione).

Il MES è un sistema in grado di gestire gli ordini e le risorse necessarie, di eseguire le fasi di produzione (controllando l'avanzamento), di tracciare ogni prodotto e di raccogliere dati e monitorare costantemente l'andamento produttivo, per analizzare le performance.
Un altro compito fondamentale del MES è quello di produrre dei dati fondamentali per tutto il reparto produzione, questi in un secondo momento possono essere gestiti dai data mining, sia per effettuare analisi predittive che per creare grafici.

Queste innovazioni tecnologiche, come il MES, hanno un impatto verso quattro direttrici di sviluppo: il settore che si occupa del passaggio da digitale a reale (stampanti 3D, robotica, …), l'interazione tra uomo e macchina, big data e, infine, analytics, dove ricaviamo valore ai dati raccolti.
Servono, anche, a rendere la produzione aziendale automatizzata e interconnessa e a portare l'azienda verso l'"Industry 4.0".

7. Bibliografia

- *Lezione 07_LAB2019 corso Basi di Dati.*
- *Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini, "Progetto di Basi di Dati Relazionali. Lezioni ed esercizi", Pitagora Editrice Bologna, 2007.*

- https://www.lar.it/, *Lar S.p.A.*, visionato il 26/02/2021.
- https://www.confindustriaemilia.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/91270, Articolo: "*Lar, 80 anni di tradizione e innovazione da festeggiare in Accademia*", visionato il 26/02/2021.
- https://it.wikipedia.org, *Wikipedia*, visionato il 27/02/2021.
- https://www.ictsecuritymagazine.com/articoli/il-data-mining/, Articolo: "*Il data mining*", visionato il 04/03/2021.
- https://www.sqlshack.com/delete-cascade-and-update-cascade-in-sql-server-foreign-key/, Articolo: "*DELETE CASCADE and UPDATE CASCADE in SQL Server foreign key*", visionato il 09/03/2021.
- https://it.phhsnews.com/what-is-microsoft1800, Articolo: "*Che cos'è Microsoft .NET Framework e Perché è installato sul mio PC?*", visionato il 12/03/2021.
- https://giordanoblog.altervista.org/storia-del-linguaggio/, Articolo "*Storia del Linguaggio*", visionato il 12/03/2021.
- https://docs.microsoft.com/, *Microsoft Docs*, visionato il 15/03/2021.
- https://qastack.it/programming/2404565/sql-server-difference-between-partition-by-and-group-by, *SQL Server: differenza tra PARTITION BY e GROUP BY*, visionato il 15/03/2021.