UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI SCIENZE STATISTICHE

CORSO DI LAUREA IN STATISTICA E TECNOLOGIE INFORMATICHE

RELAZIONE DI LAUREA

Documentazione ed analisi del database "VisionSQL"; stage presso un'azienda che sviluppa applicazioni gestionali.

Documentation and analysis of the "VisionSQL" database; stage at a company that develops data management software.

Relatore:Prof. MASSIMO MELUCCI

Laureando: MAURO BRUNAZZO

Matricola: 554276

ANNO ACCADEMICO: 2008/2009

Un giorno le macchine riusciranno a risolvere tutti i problemi,
ma mai nessuna di esse potrà porne uno.

Albert Einstein

INDICE ANALITICO

1. Introduzione

Questa tesi è il risultato di un'esperienza di stage effettuato presso un'azienda che produce e commercia un software gestionale per piccole e medie imprese.
La scelta di intraprendere un percorso di stage piuttosto che addentrarmi in un progetto di approfondimento nasce dall'esigenza di dover vedere applicate alla realtà aziendale le tecniche e le metodologie acquisite nei corsi frequentati all'università. Durante gli studi, infatti, si apprendono diversi concetti importanti ma molto spesso non si riesce ad avere una visione di come questi possano essere effettivamente utilizzati e maneggiati in ambito lavorativo. Da qui, quindi, il desiderio di provare un'esperienza che mi permettesse di immergermi in un contesto diverso da quello scolastico.
La relazione, suddivisa in due parti, inizia con la presentazione dell'azienda ospitante, della sua organizzazione interna e del software che essa commercializza riportando anche l'elenco delle attività svolte e pianificate con il tutor aziendale all'avvio dello stage. Segue la prima parte in cui analizzo in dettaglio alcune nuove conoscenze che ho avuto la possibilità di acquisire nel periodo di permanenza in azienda. Nella seconda parte, invece, trovano spazio alcuni test e analisi su database in cui entrano in gioco applicazioni statistiche a dati rilevati tramite strumenti informatici; infine vengono riportate alcune esperienze significative vissute nell'ambito aziendale che sono state un momento di crescita personale seguite da un resoconto finale sullo stage.
Anticipo già la mia piena soddisfazione per il lavoro svolto, un'esperienza che mi ha permesso di scontrarmi con molti concetti trattati durante i corsi all'università e di ampliare le conoscenze su alcuni importanti elementi presenti nelle basi di dati.

2. L'azienda e il progetto

2.1 Presentazione dell'azienda

L'azienda presso la quale ho effettuato lo stage si chiama "Vision software" ed ha sede a Pernumia, in provincia di Padova.
"Vision software" è un azienda di recente creazione, che si occupa di informatica e più precisamente di quella parte dell'informatica dedicata alle applicazioni gestionali.
I prodotti che "Vision" offre sono soluzioni applicative in ambiente Windows per la gestione contabile, finanziaria, commerciale, di magazzino o di produzione per l'industria e l'azienda di distribuzione; soluzioni rivolte ad aziende che intendono impostare sul sistema informatico non solo la semplice gestione amministrativa o di magazzino ma tutta una serie di operazioni più o meno legate all'attività aziendale che di norma non vengono implementate dai classici gestionali.
Si fa riferimento, per citare un esempio, a quei software che gestiscono solo le operazioni di magazzino, articoli e documenti e non quegli aspetti che risultano collegati alla situazione contabile quali i cespiti, i ratei e i risconti, la contabilità analitica; servizi, quindi, che non possono essere controllati internamente ma che devono essere richiesti all'esterno.
Il software gestionale su cui la ditta investe è denominato "Vision" e nasce come prodotto *internazionale* grazie alla funzionalità "MYLAB Make Your Language By Yourself", che consente di creare con la massima facilità l'intera interfaccia operativa del prodotto nella lingua desiderata.
Su "Vision", poi, sono stati costruiti una serie di programmi verticali specifici per diverse attività tra cui:

- "Vision Energy": E' il software per la gestione contabile, finanziaria, commerciale e di magazzino per aziende di distribuzione all'ingrosso di prodotti petroliferi e gas

- "Vision Fresh": E' rivolto alle aziende che operano nel settore ortofrutticolo e ittico.
- "Vision Trasport": E' il software gestionale per autotrasporti di merce e persone
- "Vision Pref": E' il software gestionale adatto ad aziende di produzione di prefabbricati per l'edilizia.
- "Vision Mobile": E' il software gestionale per *Pc palmari* e aiuta tutti coloro che hanno la necessità di operare in mobilità per gestire documenti di vendita o altro.
- "Vision Cantieri": E' un software gestionale per tenere sotto controllo i costi dei cantieri e gestione generale dell'impresa
- "Vision OnWeb": Serve a tutte le aziende che vogliano vendere i propri prodotti in un nuovo mercato, il grande mercato del World Wide Web; a tutte le aziende che desiderino gestire in modo più efficiente la propria forza vendita dando ai propri agenti uno strumento in più che gli consenta di avere tutte le informazioni di loro interesse ovunque si trovino.
- "Vision SQL": E' un software completo di alto livello per la gestione delle risorse, dei dipendenti, della contabilità, del magazzino, della produzione, ecc.

2.2 Presentazione del progetto

L'attività di stage che ho svolto può essere divisa in due macroargomenti principali: disegno degli schemi concettuali e logici e test ed analisi delle prestazioni della base di dati e di quelle gestionali.
Prima dell'avvio del progetto è stata fatta, insieme al tutor aziendale, una pianificazione dei lavori che sarebbero andati a svolgersi settimana per settimana e posso dire che

questa scaletta è stata seguita alla lettera. Riporto di seguito una presentazione dello stage con tutti i temi che sono stati trattati rimandando ai successivi capitoli per una spiegazione dettagliata degli stessi.

Il progetto da me svolto è stato denominato "documentazione ed analisi database VisionSQL". Tale lavoro parte dal software gestionale VisionSQL, della linea "Vision", già realizzato da Vision Software utilizzando come ambiente di sviluppo Microsoft Visual FoxPro e Microsoft SQL Server come sistema di gestione di basi di dati.

Come già anticipato, pur rimanendo nell' ambito del database su cui poggia il software applicativo, lo scopo dello stage è stato duplice :

1. **realizzare i diagrammi** che visualizzano le relazioni tra le tabelle del database.
2. **misurare le performance delle query** inviate dal gestionale al database, in particolare valutare i tempi di esecuzione in presenza e in assenza di diagrammi.

Relativamente al primo punto serve specificare che Microsoft SQL Server include specifici strumenti atti a disegnare diagrammi in cui sono collegate (tramite chiavi) le varie tabelle del database.

Il database di VisionSQL da me analizzato disponeva già di triggers e stored procedure, ovvero procedure che definiscono tali interdipendenze.

Scopo dello stage è stato quello di riportare in forma grafica tali regole, così che fossero più facilmente consultabili da personale tecnico che non avesse però adeguate conoscenze .

Relativamente al secondo punto si precisa che il software VisionSQL esegue numerosissime interrogazioni al database, spesso tramite query che coinvolgono diverse tabelle messe in relazione.

Scopo dello stage è stato :

- individuare, tramite precise misurazioni, lo stato dell'arte delle prestazioni
- eseguire una esaustiva serie di test in ambienti diversi e con database diversi

Infine sono stati effettuati ulteriori test per valutare la differenza in termini di prestazioni di due diversi software gestionali.

Nelle pagine che seguono sono presentati gli strumenti utilizzati oltre ai concetti con i quali sono venuto a contatto grazie a questa importante esperienza di stage.

PARTE PRIMA

All'interno del database

3. SQL Server e i diagrammi di database

3.1 Microsoft SQL Server 2005

Microsoft SQL Server è un DBMS, ossia un sistema di gestione di basi di dati, prodotto da Microsoft. Dalla prima versione risalente al 1988 fino a quella rilasciata nell'anno 1999, questo strumento è stato utilizzato principalmente per basi di dati medio-piccole; a partire dalla versione distribuita nell'anno 2000, invece, è stato usato anche per la gestione di database di grandi dimensioni. Di recente, più precisamente nel mese di agosto 2008, Microsoft ha rilasciato una nuova release denominata "SQL Server 2008". Durante l'esperienza di stage in azienda ho avuto modo di utilizzare "SQL Server 2005" del quale sono disponibili varie sottoversioni:

A PAGAMENTO:

- Enterprise edition
- Workgroup edition
- Standard edition

GRATUITE:

- Express edition

La sostanziale differenza fra le versioni a pagamento consiste nel supporto che queste offrono in termini di ram e processore. In particolare "SQL Server Workgroup edition"

è in grado di sfruttare per l'elaborazione fino a quattro Gigabyte di ram e un solo processore, la versione "Standard" supporta fino a quattro processori e non pone limiti sulla ram mentre quella "Enterprise" non ha limitazioni né per quanto riguarda i processori né per la ram.

Per lo svolgimento della prima parte del progetto relativa alla costruzione di diagrammi di database ho utilizzato "SQL Server 2005 Express edition" che risulta essere 'bloccato' in quanto può sfruttare una sola cpu, un Gygabyte di ram e supporta database con dimensioni massime fino a 4 Gigabyte.
Nella seconda fase dello stage, ossia quando ho iniziato ad effettuare test su database, ho utilizzato la versione di SQL Server denominata "Enterprise" in quanto ho operato con database di dimensioni nell'ordine delle decine di Gigabyte che non sarebbero stati supportati dalla versione "Express".
"SQL Server" include anche una console di amministrazione e gestione grafica chiamata "SQL Server Management Studio Express" ; questo programma ci consente di interagire con tutta la fase di progettazione, gestione e messa in sicurezza del database. Vista l'importanza e il largo uso che ne ho fatto durante lo stage, questa console verrà presentata con maggior dettaglio nel prossimo sottoparagrafo.

3.1.1 Management studio

Assieme a SQL Server, viene fornito uno strumento completo per poter operare completamente in maniera visuale con il DBMS: il Management Studio; con esso siamo in grado di gestire tutti gli oggetti dei nostri database e di interagire con questi, ma non solo.
SQL Server Manegement Studio è un programma che, connettendosi a SQL Server, consente all'amministratore di gestire il database. Una volta avviata l'applicazione viene subito richiesta una credenziale di accesso per poter accedere al sistema come mostrato in figura 3.1.

Figura 3.1. Schermata di Login.

Da qui si può scegliere se effettuare l'accesso tramite il nostro attuale utente di windows, oppure tramite un utente presente in SQL Server, che potrebbe essere, ad esempio, l'utente SA (System Administrator).

Una volta entrati, nella sezione sinistra della finestra, ci viene presentato un pannello contenente tutti i database presenti, e per ognuno di questi è presente un corposo menù ad albero che elenca gli oggetti. Troveremo qui le tabelle, le viste, le stored procedure. In ogni tabella avremo elencati i campi, le chiavi, gli indici e così via per tutti gli oggetti.

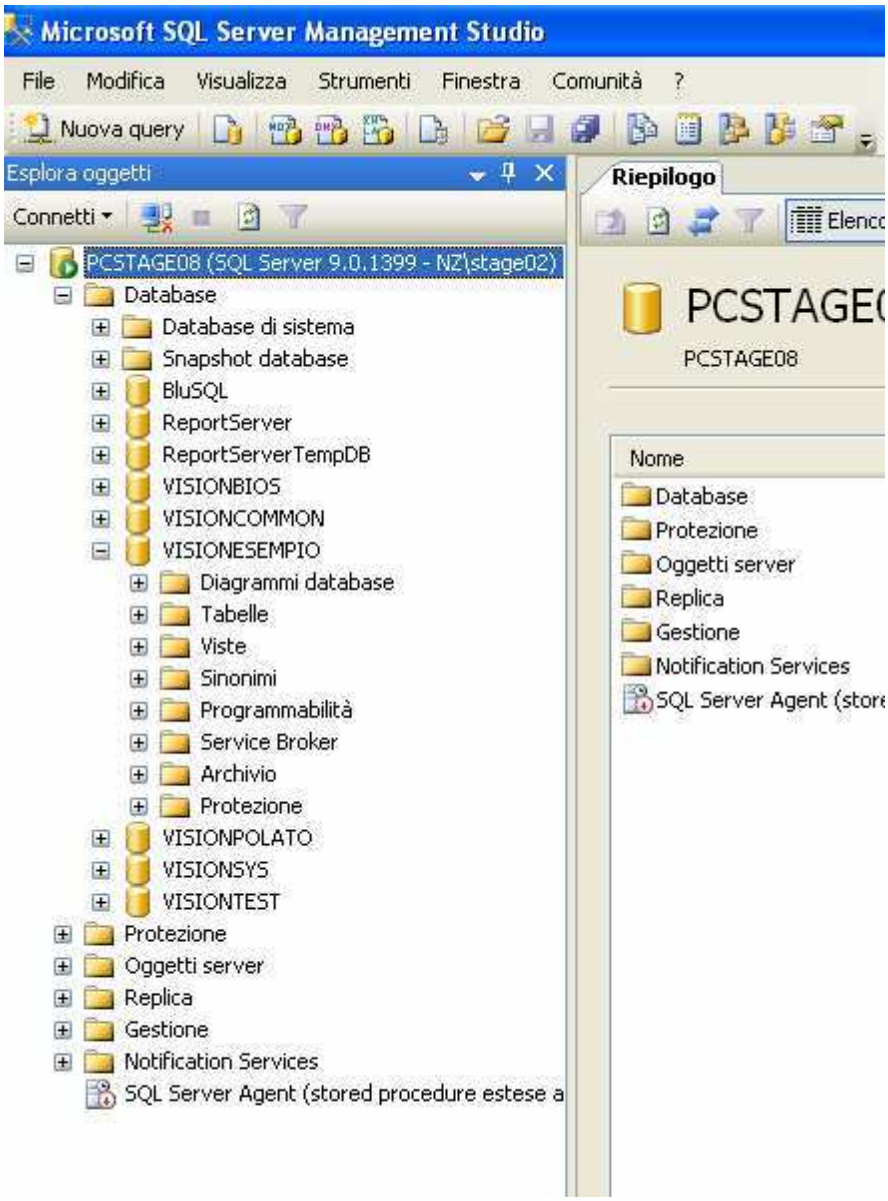

Figura 3.2. Sezione esplora oggetti.

3.2 Diagrammi di database

Quando si devono effettuare delle modifiche a tabelle in una base di dati può risultare difficoltoso avere un'idea immediata degli effetti che potrebbero scatenarsi a seguito di determinate azioni. Questo può essere ancora più complicato quando maneggiamo database composti da molte tabelle, dove riuscire a stabilire e ricordare a memoria tutte le varie dipendenze risulta pressoché impossibile.
Tuttavia, in moltissimi casi, è assai utile essere a conoscenza delle relazioni a cui partecipa una tabella specialmente se si devono apportare modifiche agli attributi o alle chiavi di questa.
Ecco allora che ci vengono in aiuto i diagrammi di database, strumenti tramite i quali risulta possibile rappresentare in forma grafica tutti i legami tra le tabelle della base di dati comportando per l'utente la possibilità di avere un'immediata visione grafica della struttura del database e delle relazioni in esso presenti.

"Un diagramma di database è una rappresentazione grafica dello schema del database. Lo schema è la struttura del database e descrive alcune cose come il nome delle colonne, i tipi di dati, le relazioni fra le tabelle e molti altri componenti della struttura del database."

[Fonte: Mastering Microsoft SQL server 2005, Gunderloy,L.Jorden,W.Tscharz,
SYBEX editrice]

3.2.1 Non si seguono gli schemi...ma la teoria aiuta!

Il lavoro che mi sono trovato a svolgere durante l'esperienza di stage è risultato in qualche modo opposto al procedimento appreso nei corsi frequentati all'università. In tutti gli insegnamenti di basi di dati, infatti, viene fin da subito fatta chiarezza sulle fasi da seguire per lo sviluppo di un database. Queste sequenze vedono come primo passo

la progettazione concettuale in cui viene fatto un esame della realtà di interesse e ne viene prodotta una rappresentazione grafica mediante un diagramma E/R. Il passo successivo consiste invece nella progettazione logica dove viene eseguita la traduzione dello schema concettuale nel modello di rappresentazione dei dati adottato dal sistema di gestione di basi di dati a disposizione. Infine troviamo la progettazione fisica in cui lo schema logico viene completato con la specifica dei parametri fisici di memorizzazione dei dati.

Il lavoro che ho dovuto svolgere io, come già anticipato, è stato contrario a questa sequenziale procedura. Infatti, fin da subito, mi sono trovato davanti ad un database già costruito con tabelle e vincoli d'integrità referenziale ben definiti. Quello che mi è stato chiesto è stato, partendo dalla base di dati disponibile, costruire più diagrammi di database sulle tabelle ritenute di maggior importanza e indicatemi dal responsabile. In qualche modo, quindi, posso dire che sono partito dallo schema logico del database e ho dovuto ricostruire lo schema concettuale, anche se il diagramma di database non può di certo essere considerato uno schema concettuale. Mi permetto comunque di dire che se un database di tale entità è stato progettato e costruito sicuramente ci deve essere stato un progetto, uno schema o comunque una traccia da seguire; forse quella traccia non è più disponibile o forse le ragioni sono altre fatto sta che mi è stato chiesto di fare una mappatura del database "a posteriori".

All'inizio ho proceduto studiando tutte le chiavi primarie e le chiavi esterne presenti nelle tabelle e analizzabili tramite lo strumento "SQL Server Management Studio"; da qui, seguendo una tecnica appresa durante i corsi di basi di dati, sono riuscito a individuare con non poca pazienza e attenzione tutte le relazioni presenti e ad avere un'idea generale dei collegamenti fondamentali.

Il secondo passo consisteva appunto nel disegno dei diagrammi illustranti le dipendenze in cui erano coinvolte le principali tabelle del database. Per fare ciò mi sono state di fondamentale aiuto le conoscenze acquisite riguardo i possibili tipi di relazioni fra tabelle e come queste relazioni vengano poi implementate nella costruzione del database; tramite queste nozioni sono riuscito a risalire a tutti i collegamenti presenti e

ho quindi potuto procedere ad una rappresentazione grafica delle varie relazioni. Devo dire che questa fase anche se delicata, mi ha molto entusiasmato perché ho potuto avere un riscontro di come nella realtà aziendale fossero applicate le regole studiate durante i corsi all'università.

Completata la costruzione dei diagrammi rappresentanti le relazioni a cui partecipavano le fondamentali tabelle del database mi è stato chiesto di effettuare uno studio dei trigger e delle stored procedure ancorati alle tabelle principali e trovare un modo grafico per rappresentare nel diagramma, oltre ai vincoli di integrità referenziale , anche tutti gli effetti che potevano verificarsi a seguito di operazioni quali aggiornamento, modifica o inserimento dati nelle tabelle da me analizzate . Nello svolgere questo lavoro mi sono imbattuto nel linguaggio Transact-SQL proprio di Microsoft SQL server. Con l'aiuto della documentazione disponibile nel sito microsoft e dei manuali di SQL server 2005 presenti in azienda sono riuscito a capire la logica di funzionamento di questo linguaggio e i costrutti fondamentali in esso presenti; durante questa fase ho incontrato il concetto di trigger,cursore e stored procedure che erano stati accennati in qualche corso di basi di dati frequentato all'università ma di certo non si era mai avuto un approfondimento sul tema e tanto meno mai si era entrati nel lato tecnico della questione. Da qui, quindi, l'esigenza di trovare della documentazione a riguardo per saperne di più su questi costrutti importanti e riuscire quindi a capire la logica che stava alla base di molti trigger analizzati. Questo lavoro mi è piaciuto molto perché mi ha permesso innanzitutto di ampliare le mie conoscenze sugli oggetti presenti in un database ma anche di acquisire una consapevolezza di quanto importante sia l'utilizzo di alcuni strumenti per la costruzione e il mantenimento di una base di dati efficiente e consistente.

L'ultimo compito in termini di rappresentazioni grafiche che mi è stato assegnato ha visto la costruzione di un diagramma che illustrasse le dipendenze presenti fra tutte le centosettantatre tabelle del database; non nascondo che si è trattato di un lavoro impegnativo e delicato ma il risultato ottenuto è stato molto soddisfacente sia da parte mia sia da parte dell'azienda.

Questo diagramma, oltre a quelli sulle tabelle considerate di maggior importanza, saranno utilizzati innanzitutto per uso interno da parte della stessa ditta per esempio nel reparto di programmazione per avere una visione chiara e immediata degli effetti che si potrebbero verificare a seguito di eventuali modifiche a tabelle ma anche per uso da parte del personale tecnico dell'azienda che magari non è a conoscenza di certi strumenti utilizzati nell'ambito dei database ma può avere la necessità di sapere le relazioni a cui partecipa una tabella e gli effetti che alcune azioni sulla stessa possono avere per l'intera la base di dati.
Un'altra cosa che verrà fatta con i diagrammi di database da me costruiti sarà di includerli nel pacchetto software fornito dall'azienda ai suoi partner con la possibilità per questi ultimi di avere una rappresentazione chiara e rapida della struttura del database che sta alla base del software gestionale. Ci tengo a precisare che questa decisione da parte dell'azienda è avvenuta a seguito della valutazione di precise misurazioni da me effettuate circa la "bontà" dei diagrammi; per ulteriori chiarimenti rimando alla sezione relativa alla seconda parte dello stage in cui tratterò ampliamente l'argomento.
Riporto invece nei successivi paragrafi una descrizione dettagliata degli elementi appena presentati cercando, per quanto possibile, di dare una spiegazione teorica e un'illustrazione pratica di questi.

4. Costrutti importanti nelle basi di dati

4.1 Trigger

4.1.1 Basi di dati attive

Una base di dati si dice attiva quando dispone di un sottosistema integrato per definire e gestire regole di produzione (dette anche regole attive). Le regole seguono il cosiddetto paradigma *Evento-Condizone-Azione* (ECA) : ciascuna regola reagisce ad alcuni eventi (normalmente modifiche della base di dati), valuta una condizione e, in base al valore di verità della condizione, esegue una reazione.

"Un evento è qualcosa che accade, o si verifica, che è di interesse e che può essere mappato in un istante di tempo. "

"Un condizione è un controllo che viene eseguito quando la regola è considerata e prima che l'azione sia eseguita."

"Un'azione è una sequenza di operazioni che viene eseguita quando la regola è considerata e la sua condizione risulta essere vera."

[FONTI: http://www.disi.unige.it/person/CataniaB/teach/BDer-02-03/attivi.ppt

Atzeni, Ceri, Paraboschi, Torlone, "*Basi di dati - seconda edizione*", McGraw-Hill.]

Più nello specifico, ragionando in termini di database:

- gli eventi sono primitive per la manipolazione dei dati in SQL (insert, delete, update);

- la condizione (che può talvolta mancare) è un predicato booleano, espresso in SQL;
- l'azione è una sequenza di primitive SQL generiche, talvolta arricchite da linguaggio di programmazione integrato disponibile nell'ambito di uno specifico prodotto.

L'esecuzione delle regole avviene sotto il controllo di un sotto-sistema autonomo, detto processore delle regole (rule engine), che tiene traccia degli eventi e manda in esecuzione le regole in base a proprie politiche; in questo modo, si determina un alternarsi tra l'esecuzione delle transazioni, lanciate dagli utenti, e delle regole, lanciate dal sistema; si dice che il sistema risultante ha un comportamento reattivo, che si differenzia dal tipico comportamento passivo di una base di dati priva di regole attive. Quando una base di dati ha un comportamento reattivo, una parte dell'applicazione normalmente codificata mediante i programmi può essere espressa tramite regole attive. Le regole attive possono, ad esempio, gestire vincoli di integrità, calcolare dati derivati e gestire eccezioni, oltre a codificare vere e proprie "regole aziendali". Questo fenomeno aggiunge all'indipendenza delle basi di dati una nuova dimensione, detta indipendenza della conoscenza; la conoscenza di tipo reattivo viene sottratta ai programmi applicativi e codificata sotto forma di regole attive. Il vantaggio introdotto da questa nuova dimensione di indipendenza è che la conoscenza relativa al comportamento reattivo viene definita una volta per tutte sotto forma di regole, che fanno parte dello schema e vengono condivise da tutte le applicazioni, invece che essere replicata in tutti i programmi applicativi; modifiche alle elaborazioni di tipo reattivo possono essere gestite semplicemente cambiando le regole attive, senza dover modificare le applicazioni.

Quasi tutte le basi di dati possono essere anche considerate basi di dati attive, in quanto mettono a disposizione semplici regole, dette trigger.

4.1.2 Comportamento dei trigger

La creazione dei trigger fa parte del Data Definition Language (DDL); i trigger possono anche essere cancellati e in taluni sistemi attivati e disattivati dinamicamente.
In genere i trigger fanno riferimento ad una tabella, detta *target*, in quanto rispondono ad eventi relativi a tale tabella.
Il paradigma ECA (Evento-Condizione-Azione) ha un comportamento semplice e intuitivo: quando si verifica l'evento, se la condizione è soddisfatta, allora viene svolta l'azione. Si dice che un trigger è *attivato* da uno dei suoi eventi, viene *valutato* durante la verifica della sua condizione e viene e *eseguito* quando, a seguito della sua valutazione, viene eseguita la parte azione. Tuttavia, vi sono differenze significative nel modo in cui i sistemi definiscono attivazione, valutazione ed esecuzione dei trigger.
I trigger relazionali hanno due livelli di granularità, detti di tupla (*row-level*) o di primitiva (*statement-level*). Nel primo caso, l'attivazione avviene per ogni tupla coinvolta nell'operazione; si ha cioè un comportamento orientato alle singole istanze. Nel secondo caso, l'attivazione avviene una sola volta per ogni primitiva SQL facendo riferimento a tutte le tuple coinvolte dalla primitiva, con un comportamento orientato agli insiemi.
Inoltre i trigger possono avere la modalità *immediata* oppure *differita*. Quando i trigger hanno modalità immediata, la loro valutazione in genere avviene immediatamente dopo l'evento che li ha attivati (opzione *after*) ; più raramente, la valutazione del trigger deve precedere logicamente l'evento cui si riferisce (opzione *before*). Invece la valutazione differita dei trigger avviene alla fine della transazione, a seguito di un comando commit-work.
E' possibile che i trigger si riattivino l'uno con l'altro; ciò accade quando l'azione di un trigger è anche l'evento di un altro trigger. In tal caso, si dice che i trigger sono in *cascata*. E' anche possibile che i trigger si attivino l'un l'altro in modo indefinito, generando situazioni di non terminazione

Riporto nella tabella seguente un schema riassuntivo sulle modalità di esecuzione dei trigger:

Tabella 4.1: Modalità di esecuzione di un trigger.

	STATEMENT	ROW
BEFORE	**Trigger before statement**: Il trigger è eseguito un'unica volta prima dell'esecuzione del comando che lo attiva	**Trigger before row**: Il trigger è eseguito prima di modificare ogni tupla coinvolta dall'esecuzione del comando che attiva il trigger
AFTER	**Trigger after statement**: Il trigger è eseguito un'unica volta dopo l'esecuzione del comando che lo attiva	**Trigger after row**: Il trigger è eseguito dopo aver modificato ogni tupla coinvolta dall'esecuzione del comando che attiva il trigger

4.1.3 Creazione di un trigger in SQL Server 2005

Quando si crea un trigger, è necessario specificare le informazioni seguenti:

- Il nome.
- La tabella in cui viene definito il trigger.
- Quando verrà attivato il trigger.
- Le istruzioni di modifica dei dati che attivano il trigger. Le opzioni valide sono INSERT, UPDATE o DELETE.
- Le istruzioni di programmazione che eseguono l'azione del trigger.

Riporto di seguito la sintassi di riferimento utilizzata da SQL Server 2005 per la creazione di un trigger:

```
----------------------------------------------------------------------------------------------------
      CREATE TRIGGER [Utente].[NomeDatabase]
      ON
         [Utente].[NomeTabella]
      {INSTEAD OF|AFTER|BEFORE} {INSERT|UPDATE|DELETE}
       AS
         BEGIN

           [elenco istruzioni in T–sql]

         END
----------------------------------------------------------------------------------------------------
```

CREATE TRIGGER è l'istruzione che fisicamente crea il trigger, la parola chiave ON, invece, ci indica la tabella sulla quale viene ancorato; segue la regola di attivazione, quindi l'elenco delle istruzioni da eseguire racchiuse fra le parole chiave BEGIN ed END.

Nota: Nella mia esperienza in azienda ho incontrato molto spesso i trigger "INSTEAD OF"; questi indicano al DBMS che il trigger deve essere eseguito al posto dell'evento. È possibile definire un trigger INSTEAD OF per eseguire il controllo degli errori o dei valori in una o più colonne, quindi effettuare ulteriori azioni prima di inserire il record.

4.1.4 Tabelle di transizione

Nelle istruzioni di trigger vengono molto spesso utilizzate due tabelle speciali, ovvero la tabella “deleted” e la tabella “inserted”. In SQL Server queste tabelle sono create e gestite automaticamente. È possibile utilizzare queste tabelle temporanee residenti in memoria per verificare gli effetti di determinate modifiche apportate ai dati e impostare le condizioni per le azioni dei trigger DML.
Le tabelle “inserted” e “deleted” vengono principalmente utilizzate per eseguire le operazioni seguenti:

- Estendere l'integrità referenziale tra le tabelle.
- Verificare la presenza di errori ed eseguire le azioni appropriate sulla base dell'errore rilevato.
- Individuare le differenze tra lo stato di una tabella prima della modifica dei dati e lo stato della tabella stessa dopo la modifica, per eseguire le azioni appropriate sulla base di tali differenze.

Nella tabella “deleted” vengono archiviate copie delle righe interessate dall'esecuzione delle istruzioni DELETE e UPDATE. Durante l'esecuzione di un'istruzione DELETE o UPDATE, le righe vengono eliminate dalla tabella di trigger e trasferite nella tabella “deleted”**.** In genere, la tabella “deleted” e la tabella di trigger non hanno righe in comune.
Nella tabella “inserted” vengono archiviate copie delle righe interessate dall'esecuzione delle istruzioni INSERT e UPDATE. Durante una transazione INSERT o UPDATE, le nuove righe vengono aggiunte sia alla tabella “inserted” che alla tabella di trigger. Le righe della tabella “inserted” sono copie delle nuove righe della tabella di trigger.
Una transazione UPDATE corrisponde a un'eliminazione seguita da un inserimento. Per prima cosa le righe precedenti tale operazione vengono copiate nella tabella “deleted” e quindi le nuove righe vengono copiate nella tabella “inserted”.

Quando un'istruzione INSERT, UPDATE o DELETE fa riferimento a una tabella con trigger “instead of”, il motore di database esegue una chiamata al trigger invece di eseguire azioni sulle tabelle.

4.1.5 Applicazioni alle basi di dati attive

I trigger sono utilizzati per diversi scopi nella progettazione di un database, e principalmente:

1. per mantenere l'integrità referenziale tra le varie tabelle
2. per mantenere l'integrità dei dati della singola tabella
3. per monitorare i campi di una tabella ed eventualmente generare eventi ad hoc
4. per gestire i dati replicati

Queste applicazioni più classiche delle regole attive sono *interne* alla base di dati: il gestore delle regole attive opera come sottosistema del DBMS per implementare alcune sue funzionalità; in tal caso, i trigger sono generati dal sistema e talvolta non visibili agli utenti.
Altre regole, classificate come *esterne*, esprimono conoscenza di tipo applicativo che sfugge a schemi rigidi predefiniti. Queste regole vengono anche denominate *regole aziendali* in quanto esprimono le strategie di una azienda nel perseguire i propri scopi primari. Nel caso delle regole aziendali non esistono però tecniche fisse per derivare le regole a partire dalle specifiche, quindi ciascun problema applicativo deve essere affrontato separatamente.

4.1.6 Regole aziendali

Le regole aziendali esprimono le strategie di un'azienda nel perseguire i propri scopi primari. Esempi sono le regole che descrivono acquisto e cessione di titoli in base alle fluttuazioni del mercato, le regole per la gestione di una rete di trasporti, oppure anche le regole per la gestione del magazzino in base alle variazione delle quantità giacenti. Le regole attive sono perfettamente adatte per specificare e implementare vincoli e reazioni "generici".

4.1.7 Il linguaggio Transact-SQL

A volte abbreviato con T-SQL, Transact-SQL è l'estensione proprietaria del linguaggio SQL sviluppata da Microsoft e fornita insieme a Microsoft SQL Server.
In particolare, Transact-SQL espande le prestazioni di SQL aggiungendo, tra l'altro:

- Funzioni per controllo di flusso
- Possibilità di definire variabili locali.
- Varie funzioni per la manipolazione di stringhe, date, espressioni matematiche.
- Miglioramento delle istruzioni DELETE e UPDATE.

Nei trigger da me analizzati in azienda ho incontrato molti costrutti del linguaggio T-SQL; riporto di seguito una panoramica sugli elementi fondamentali del linguaggio.

Tabella 4.2: Costrutti fondamentali del linguaggio T-SQL

ELEMENTO DEL LINGUAGGIO	DESCRIZIONE
BACKUP	Effettua un backup del database, del log di transazioni o di uno o più file o di gruppi di file.
BEGIN	Inizia un blocco di dichiarazioni
BREAK	Esce da un ciclo WHILE
CASE	Esamina un elenco di condizioni e restituisce una tra più

	espressioni di risultato possibili.
CONTINUE	Riavvia un ciclo WHILE
Dichiarazioni DCL	Comprendono le dichiarazioni SQL standard GRANT e REVOKE
Dichiarazioni DDL	Comprendono le dichiarazioni CREATE, ALTER, e DROP , con estensioni per la creazione e la gestione di trigger e procedure memorizzate scritte in Transact-SQL
Dichiarazioni di controllo cursore	Comprendono le dichiarazioni DECLARE CURSOR, OPEN, FETCH, CLOSE e DEALLOCATE
Dichiarazioni di transazioni	Comprendono le dichiarazioni BEGIN TRANSACTION, END TRANSACTION, COMMIT e ROLLBACK
Dichiarazioni DML	Comprendono le dichiarazioni SQL standard INSERT, UPDATE e DELETE
DECLARE	Definisce variabili Transact-SQL
END	Termina un blocco di dichiarazioni
EXEC	Esegue una procedura memorizzata.
GOTO	Trasferisce incondizionatamente il controllo a un'etichetta di dichiarazione.
IF…ELSE	Esamina una o più condizioni, fornendo dichiarazioni da eseguire quando una condizione viene valutata come TRUE.
PRINT	Restituisce un messaggio al client.
RESTORE	Ripristina da un backup un database, in log di transazioni o uno o più file o gruppi di file.
RETURN	Termina la procedura corrente, restituendo il controllo al modulo chiamante.
SELECT [INTO]	Comprende la dichiarazione SQL standard SELECT con una clausola opzionale INTO aggiunta per consentire la memorizzazione di dati selezionati in variabili di linguaggio Transact-SQL dichiarate.

SHUTDOWN	Ferma immediatamente il motore DBMS.
TRUNCATE TABLE	Svuota una tabella di tutti i dati, ma senza attivare i trigger definiti sulla tabella.
USE	Indica il database che verrà utilizzato da tutto il codice Transact-SQL successivo.
WAITFOR	Ritarda l'esecuzione di dichiarazione
WHILE	Ripete le dichiarazioni per tutto il tempo per cui una condizione specifica è TRUE

4.2 Cursori

Un linguaggio di programmazione procedurale specifica una sequenza esplicita di passaggi da seguire per completare un'operazione. Nelle applicazioni che utilizzano database relazionali, gli sviluppatori utilizzano diverse tecniche per comunicare con il la base di dati, tra cui naturalmente l'invio di dichiarazioni SQL al database per l'elaborazione e la gestione dei risultati.
I linguaggi di programmazione procedurali sono progettati per la gestione di un record alla volta. Questo risulta essere un problema quando, insieme al linguaggio di programmazione, viene usato SQL perché in genere le query SQL restituiscono set di risultati contenenti più righe di dati. Per riuscire ad operare mediate un sistema "tupla per tupla" è stato introdotto il concetto di *cursore*, uno strumento che, definito su una generica interrogazione, permette ad un programma di accedere alle righe di una tabella una per volta.
I passi fondamentali per poter utilizzare correttamente un cursore sono:

- Dichiarazione
- Apertura
- Lettura del record puntato
- Chiusura

4.2.1 Dichiarazione di un cursore

Un cursore deve essere dichiarato prima che sia possibile fare riferimento ad esso in altre istruzioni SQL; in questa fase, inoltre, si associa il cursore ad una particolare interrogazione sulla base di dati. Al momento della dichiarazione è necessario specificare, oltre al nome del cursore stesso, gli attributi e la tabella sulla quale esso opererà.

```
DECLARE nomeCursore CURSOR
FOR SELECT  nomeCampo1,nomeCampo2,....nomeCampoN
FROM nomeTabella
```

4.2.2 Apertura di un cursore

Prima di poter essere utilizzato, il cursore deve essere aperto. L'istruzione di apertura apre il cursore e lo popola con un insieme di tuple prese dalla tabella alla quale esso è ancorato; la sintassi è molto semplice:

```
OPEN nomeCursore
```

4.2.3 Lettura di un record

Ogni qualvolta il programma richiede una nuova riga di dati, si utilizza semplicemente un comando *FETCH* sul cursore; questo equivale alla lettura del record successivo dalla tabella risultato. Il cursore, infatti, è semplicemente un puntatore nel set di risultati;

ogni qualvolta viene emesso un comando *FETCH*, il cursore avanza di una riga e successivamente restituisce la stessa al programma chiamante (il programma che ha inviato il comando *FETCH*). Se non sono presenti altre tuple nella tabella risultato, ossia se il cursore è avanzato oltre l'ultima riga, al programma chiamante viene restituito un codice volto a indicare tale situazione. Un altro dettaglio gestito dal comando *FETCH* è la mappatura delle colonne ottenute su variabili del linguaggio di programmazione. Questa operazione viene effettuata con la clausola INTO come di seguito riportato:

```
FETCH nomeCursore
[INTO Var1,var2,.....,varN]
```

4.2.4 Chiusura del cursore

E' sempre necessario chiudere il cursore quando il programma non ne ha più necessità perché in tal modo si liberano le risorse utilizzate da questo. La dichiarazione di chiusura è di semplice utilizzo:

```
CLOSE nomeCursore
```

In Transact-SQL la memoria utilizzata da un cursore non viene completamente liberata al momento della sua chiusura; il linguaggio T-SQL fornisce un'ulteriore istruzione che permette di liberare completamente la memoria usata da un cursore; il comando necessario a svolgere tale operazione è il seguente:

```
DEALLOCATE nomeCursore
```

4.3 Stored procedure

4.3.1 Le query memorizzate

Quando inviamo una query a server SQL, di fatto quello che viene mandato è un batch di comandi SQL. Oltre a questo meccanismo esistono altre possibilità di comunicazione client-server; un esempio è fornito dalle stored procedure, ossia sequenze di comandi SQL memorizzati sul server, che sono così disponibili per futuri utilizzi in modo rapido ed efficace. Prima di analizzare nel dettaglio le stored procedure ritengo sia necessario fare chiarezza sul concetto di batch SQL.

4.3.2 I batch SQL

Un batch è sempre formato da uno o più comandi SQL che vengono inviati ed eseguiti sul server SQL. Questo implica una sincronizzazione tra client e server; ogni qualvolta il client invia un batch con delle istruzioni è richiesto il suo riconoscimento da parte del server, il parsing dei comandi inviati, la loro esecuzione e il ritorno di un codice che identifichi lo stato (positivo o negativo) dell'esecuzione del comando inviato.
Queste operazioni vengono eseguite ogni qualvolta si invia un batch SQL, indipendentemente dal numero di comandi coinvolti; proprio per questo motivo è meglio accodare molte più istruzioni SQL all'interno di un singolo batch per diminuire il carico di lavoro sul server.
Volendo portare un esempio pratico, se vengono lanciati mille comandi INSERT, ognuno in batch singoli il carico di lavoro sarà di gran lunga maggiore rispetto a quello che si avrebbe inviando un unico batch con all'interno le mille istruzioni INSERT.

4.3.3 Le procedure memorizzate

Le stored procedure rappresentano il "cuore" della programmazione Transact SQL. Presenti fin dalle prime versioni di SQL Server sono gruppi di istruzioni SQL compattati in un modulo e memorizzati nel Server per un successivo utilizzo.

Racchiudere il codice SQL all'interno di procedure memorizzate porta a dei grossi vantaggi rispetto ai batch di codice SQL tradizionale tra cui:

1. Aumento nella velocità di esecuzione delle istruzioni SQL e quindi delle performance generali delle applicazioni.
2. Aumento della leggibilità e della portabilità del codice.
3. Maggior livello di sicurezza per il database.

In merito alla velocità di esecuzione delle istruzioni SQL, c'è da dire che in un sistema client-server un parametro molto importante da tenere sempre monitorato è il traffico di rete. Consideriamo per un attimo un database privo di stored procedure, in cui lavorano qualche migliaio di utenti che giornalmente effettuano la medesima interrogazione alla base di dati; questo si traduce nell'invio di migliaia di richieste uguali al server SQL il che comporta un notevole traffico di rete e il rischio di creare congestioni nella rete stessa ottenendo come risultato un rallentamento generale dei tempi di esecuzione delle interrogazioni.
Con l'utilizzo delle stored procedure, invece, è possibile memorizzare il codice SQL della query nel server e attribuirgli un nome, permettendo così agli utenti lato client di richiamare l'esecuzione dell'interrogazione semplicemente invocando il nome della procedura. Come è facile intuire, in questo modo, il traffico di rete viene notevolmente ridotto e di conseguenza diminuiscono anche i tempi effettivi di esecuzione. Un altro fattore che va ad influire sul tempo di risposta è rappresentato dalla compilazione della query che vede la fase di controllo della correttezza del codice e ricerca della strategia migliore per l'esecuzione dell'interrogazione da parte del DBMS. In un database che non utilizza stored procedure queste fasi devono essere ripetute ad ogni esecuzione della query; con l'uso delle procedure memorizzate, invece, la compilazione dell'interrogazione avviene nel momento della sua memorizzazione e alle chiamate successive la query viene semplicemente eseguita. Anche questo, ovviamente, contribuisce ad un miglioramento generale delle performance del database.

Le procedure possono essere create sia per uso permanente che temporaneo ed inoltre possono essere avviate in modo automatico quando viene avviato SQL Server.
Una stored procedure può accogliere una quantità di istruzioni SQL fino alla dimensione massima di centoventotto Megabyte; inoltre alla procedura può essere passato un numero di parametri massimo pari a duemilacento.
SQL Server stesso possiede una serie di procedure dette "di sistema" che vengono generate al momento della sua installazione e sono necessarie ad eseguire una serie fondamentale di compiti che vanno dalla creazione dei database alla loro manutenzione (utenti, permessi, repliche, backup, restore, ecc...).
In questo capitolo, tuttavia, tratterò solamente le stored procedure che vengono create dagli utenti di SQL Server.

4.3.4 Come creare le procedure

Le istruzioni per la creazione di stored procedure fanno parte del linguaggio DDL e di seguito riporto la sintassi di dichiarazione.

```
CREATE PROCEDURE [Utente].[NomeProcedura]
    @nomeParametro1 tipo = ValoreDefault,
    ..............,
    @nomeParametroN tipo = ValoreDefault
    AS
        BEGIN
         [elenco istruzioni in T–sql]
        END
```

Per creare una procedura memorizzata, quindi, è sufficiente eseguire in un batch l'istruzione CREATE PROCEDURE dichiarando i parametri necessari ed infine aggiungere le istruzioni Transact SQL costituenti il corpo vero e proprio della procedura.
Le procedure possono richiamare ed essere richiamate da altre procedure e così via fino ad un livello di nidificazione pari a trentadue; questo limite è imposto da SQL Server per impedire errori di overflow.

4.3.5 Eseguire una procedura

Esistono due differenti modi per richiamare una stored procedure; l'uso di uno piuttosto dell'altro varia in funzione della modalità con cui vengono passati i parametri, che può essere esplicita oppure implicita. In particolare, nella modalità implicita il nome del parametro di input non viene specificato ma vengono passati solo i valori numerici che sono da intendere riferiti ad ogni parametro nell'ordine riportato in fase di dichiarazione.
Nella modalità esplicita, invece, il nome del parametro di input viene specificato e passato senza che l'ordine di chiamata nella procedura sia rilevante.

4.3.6 Utilizzo dei parametri

La caratteristica più importante delle stored procedure è che sono delle procedure in grado di ricevere e restituire parametri. In questo modo abbiamo la possibilità di riciclare il nostro codice e parametrizzarlo con tutti i vantaggi del caso.
Durante la fase di dichiarazione dei parametri di una procedura è possibile assegnare a questi dei valori di default; se ad un parametro non viene associato alcun valore di default, al momento dell'esecuzione verrà chiesto di passare un valore per quel parametro e, in caso il valore non venga fornito, SQL Server visualizzerà un messaggio di errore.

Le stored procedure possono avere parametri di tre tipi differenti:

- IN: Sono argomenti in ingresso alla procedura.
- OUT: Sono parametri restituiti dalla stored procedure.
- INOUT: Combinano le caratteristiche precedenti, ossia vengono forniti dei parametri in ingresso e si memorizzano dei valori in uscita dalla stored procedure.

Al momento della dichiarazione dei parametri, se questi vengono specificati senza l'aggiunta di nessuna parola chiave volta a indicarne la tipologia, allora sono da intendere come parametri di input; nel caso invece si desideri indicare che un parametro è di output occorre far seguire alla sua dichiarazione la parola OUT. Infine, quando un parametro è sia di input che di output, è sufficiente aggiungere la parola chiave INOUT immediatamente dopo averlo dichiarato.

PARTE SECONDA

Test e analisi su database

5. L'inferenza statistica a supporto dell'informatica

Dopo che ho completato la fase di disegno dei diagrammi il tutor aziendale mi chiese di essere rassicurato sul fatto che, avendo inserito nel database anche una serie di diagrammi, il funzionamento della base di dati stessa non ne risultasse rallentato o in qualche modo compromesso. Così mi è stato richiesto innanzitutto di trovare un modo per riuscire ad esportare i diagrammi da un database ad un altro che ovviamente avesse lo stesso schema ma fosse popolato con differenti dati. Dopodiché ho iniziato una serie di test su alcuni database al fine di testarne le prestazioni valutate come tempo di esecuzione di alcune query. I tempi di risposta sono stati rilevati nella stessa base di dati sia in presenza che in assenza dei diagrammi di database da me precedentemente creati.
La fase finale dello stage ha visto la rilevazione dei tempi di risposta nell'eseguire determinate operazioni in due differenti gestionali prodotti e commercializzati dall'azienda "Vision software house". In particolare il confronto è stato effettuato fra il software "VisionERP" e il suo futuro successore "VisionSQL"; occorre specificare che la ditta di Pernumia desidera, entro breve tempo, spingere i propri clienti a migrare da "VisionERP" a "VisionSQL". I test che ho eseguito hanno avuto come fine proprio quello di verificare le migliorie in termini di prestazioni del nuovo programma gestionale rispetto al precedente. Il lavoro svolto durante la seconda parte dello stage ha pienamente soddisfatto il mio desiderio di poter utilizzare nella realtà aziendale delle tecniche statistiche applicate all'informatica. In particolare ho avuto la possibilità di ricorrere ad alcuni test statistici studiati all'università per verificare l'ipotesi che i tempi medi di esecuzione in due differenti gruppi di osservazioni potessero essere considerati uguali a meno di variazioni casuali.
Rimando all'appendice 8.2.1 per visionare i comandi utilizzati per l'esportazione dei diagrammi mentre riporto nelle pagine che seguono la descrizione dei test da me effettuati.

5.1 Analisi delle prestazioni di alcune query

Come già anticipato i test eseguiti consistevano nel misurare il tempo di esecuzione di determinate query in alcuni database inserendo o meno i diagrammi.

5.1.1 Metodologia di lavoro

Considerando un generico database ho deciso di effettuare per ogni query trenta misurazioni in assenza dei diagrammi e altre trenta in presenza dei diagrammi. La mia idea era quella di applicare, per ogni interrogazione in esame, un test "t di Student a due campioni" al fine di indagare sulla possibile diversità in media dei tempi di risposta nel database con e senza i diagrammi. Mi sono preposto inoltre l'obiettivo di costruire per ogni query un intervallo di confidenza per la media del tempo di esecuzione dell'interrogazione sia quando nel database erano presenti i diagrammi sia nel caso in cui questi non c'erano.

5.1.2 Problemi e soluzioni

Una delle ipotesi alla base del test "t a due campioni" è quella di assumere le osservazioni all'interno dei due campioni i.i.d (indipendenti e identicamente distribuite).
Da un'analisi preliminare sui primi tempi di esecuzione misurati mi sono subito accorto che SQL Server adotta un meccanismo di "caching", ossia effettua un salvataggio in una memoria temporanea di alcune informazioni utili per l'esecuzione della query cosicché se interrogazione stessa viene lanciata una seconda volta consecutiva, i tempi di risposta risultano essere notevolmente ridotti grazie all'utilizzo di questa area di memoria.
Un primo problema che mi si è dunque presentato consisteva nel fatto che rilevazioni successive non potevano di certo essere considerate indipendenti fra loro per il motivo a cui ho appena accennato.

Con l'obiettivo di superare questo scoglio e di riuscire il più possibile a "ricreare" lo stesso ambiente ad ogni esecuzione della query ho proceduto, dopo un'attenta documentazione sui manuali di SQL Server, eseguendo, prima di lanciare l'interrogazione stessa, due istruzioni in linguaggio SQL che permettono proprio di "svuotare" la memoria cache. In questo modo non ci sono motivi per dubitare circa l'indipendenza delle misurazioni.
Un'altra ipotesi su cui si fonda il test "t a due campioni" vede la normalità della distribuzione dei tempi di risposta in ambedue le popolazioni (ossia quella con diagrammi e quella senza diagrammi).
E' possibile dimostrare tuttavia che il test e le procedure rimangono valide anche se la distribuzione all'interno dei due gruppi non è normale purché siano verificate le seguenti condizioni:

- Ambedue le numerosità campionarie siano "sufficientemente grandi".
- Le osservazioni all'interno dei gruppi siano indipendenti e identicamente distribuite.
- Ambedue le popolazioni abbiano media e varianza finite.
- Le osservazioni di un gruppo siano indipendenti dalle osservazioni dell'altro gruppo.

In merito al primo punto una semplice regola per stabilire quando una numerosità campionaria può essere ritenuta "sufficientemente grande" suggerisce che i dati presenti nel campione devono essere in numero maggiore o uguale a trenta se la distribuzione dei dati è, almeno approssimativamente, simmetrica.
Nel mio caso ho effettuato una serie di trenta rilevazioni per gruppo. Essendo verificate tutte le condizioni sopra riportate posso arrivare a concludere che, anche non avendo indagato sulla normalità dei dati, i risultati ottenuti dal test "t a due campioni" rimangono validi seppur in maniera approssimata.

Riporto di seguito i risultati ottenuti dall'analisi di una query presa come esempio rimandando all'appendice 8.3 per visualizzare tutti i dettagli algebrici che hanno condotto a tali conclusioni.

5.1.3 Risultati delle analisi statistiche

→ Query in esame:

```
-------------------------------------------------------------------------------
SELECT * FROM DocRig
         WHERE DocRig.DataDoc = convert(smalldatetime,'07/07/2004')
-------------------------------------------------------------------------------
```

→ Intervallo di confidenza di livello 0,95 per la media del tempo di risposta risulta:

Senza diagrammi [in sec.]	Con diagrammi [in sec.]
[1,6813 ; 1,8240]	[1,7225 ; 1,8669]

Commento:

L'intervallo di confidenza ci indica che con probabilità del 95% la media del tempo di esecuzione della query nel database senza i diagrammi si trova (in secondi) fra 1,6813 e 1,8240. La media dello stesso tempo di esecuzione nella versione con diagrammi è compresa invece sempre con probabilità del 95% fra 1,7225 e 1,8669 secondi.

→ Verifica dell'ipotesi che la media dei tempi di risposta in presenza o in assenza dei diagrammi di database sia uguale.

Livello di significatività osservato = 0,40.

Commento:

Il livello di significatività osservato costituisce una misura di quanto l'ipotesi formulata è plausibile sulla base dei dati.

I risultati ottenuti suggeriscono che non ci sono ragioni per dubitare circa la differenza dei tempi medi di risposta della query nel database con e senza i diagrammi.
Mi preme sottolineare il fatto che in alcune interrogazioni analizzate il test "t di Student a due campioni" suggeriva che era presente una sostanziale differenza fra le due medie confrontate. Da indagini successive e soprattutto effettuando lo stesso test con regione di rifiuto unilaterale i risultati emersi hanno indicato che il tempo di risposta nel database con diagrammi poteva essere considerato minore di quello in cui i diagrammi non erano presenti.
Questa conclusione è stata molto confortante a supporto dell'ipotesi che l'aggiunta di diagrammi non influisca negativamente sulle prestazioni del database.

5.2 Analisi dei tempi di risposta di due gestionali.

Questa fase dello stage ha avuto come obiettivo la valutazione della differenza del tempo impiegato per l'esecuzione di determinate operazioni in due diversi gestionali e operando con differenti database. I software analizzati sono stati "VisionERP" e "VisionSQL" ricordando l'intenzione dell'azienda di Pernumia di condurre i propri clienti al passaggio da "VisionERP" a "VisionSQL" entro breve termine. Quello che ci si attendeva a monte dei test, quindi, era che i tempi di esecuzione della maggioranza delle procedure in "VisionSQL" fossero minori di quelli impiegati dal suo 'rivale' "VisionERP".

5.2.1 Metodologia di lavoro

All'inizio il mio tutor aziendale mi presentò una serie di operazioni "standard" che vengono eseguite frequentemente dagli utenti che operano con il gestionale e sulle quali mi è stato richiesto di effettuare un'attenta analisi.
Le misurazioni sui tempi di esecuzione sono state fatte in diversi ambienti di lavoro; più precisamente una prima batteria di test è stata eseguita lavorando in "locale" ossia installando il server SQL e il software gestionale sulla stessa macchina. Nella seconda

fase, invece, ho lavorato con un'architettura client/server in cui il server e il programma client risiedevano in due differenti computer. La motivazione di provare ambienti di lavoro diversi è da ricercare nel differente modo di elaborare i risultati dei due gestionali.
In particolare, davanti ad un problema di filtraggio dei dati, ossia quando l'utente desidera ottenere una selezione personalizzata di tutti i dati possibili, il gestionale "VisionERP" è progettato in modo da ricevere tutti i possibili dati dal server SQL e di procedere poi lui stesso con il compito di operarne la selezione. Nella stessa situazione descritta, invece, il software "VisionSQL" lascia che sia il server ad elaborare i dati e al gestionale vengono inviati poi solamente i risultati già pronti.
La differenza sostanziale fra i due programmi, volendo quindi riassumere, sta nel fatto che in "VisionERP" è il client che deve effettuare l'elaborazione dei risultati mentre in "VisionSQL" il carico applicativo risiede sul Server.
Da qui è nata dunque l'esigenza di effettuare più prove oltre che in locale, anche in un ambiente di rete, per poter avere un riscontro sulle differenti prestazioni dei due software.
Il metodo di lavoro è stato analogo a quello utilizzato per effettuare le misurazioni sui tempi di risposta delle query; sono state effettuate trenta misurazioni per ciascuna operazione in ogni gestionale. Anche i problemi incontrati sono gli stessi descritti nella precedente sezione "Analisi delle prestazione di alcune query"; rimando quindi a tale parte nel caso fossero necessari ulteriori chiarimenti.

5.2.2 Risultati delle analisi statistiche

Le analisi e i test eseguiti ricalcano la procedura utilizzata nel caso dello studio delle query. In particolare, una volta ottenute tutte le rilevazioni, si è proceduto con l'applicazione del test "t di Student a due campioni" utilizzato per verificare l'ipotesi che i tempi medi di risposta nel gestionale "VisionSQL" potessero essere considerati minori di quelli ottenuti con il programma "VisionERP".

La conclusione alla quale si è giunti alla fine dei lavori è che in alcune operazioni analizzate “VisionSQL” si dimostrava il più rapido nell’esecuzione, mentre in altre è “VisionERP” che sembrava essere il più veloce. I risultati ottenuti, hanno evidenziato quindi la necessità di operare delle migliorie in alcune parti del gestionale “VisionSQL” prima di poter procedere alla sua commercializzazione. Le analisi da me condotte sono state presentate al responsabile del prodotto e al reparto programmazione durante una riunione aziendale; i punti di maggior carenza rilevati in “VisionSQL” saranno al più presto analizzati e migliorati dai programmatori dell’azienda.

6. Esperienze vissute in azienda

Riporto di seguito alcune esperienze che ho vissuto durante il periodo di permanenza in azienda e che a mio avviso sono state molto utili e significative.

6.1 Fare i backup dei database non basta

In data 13 Marzo 2009 ho potuto assistere ad un evento che nella "normalità" scolastica penso mai sarebbe potuto accadere. Uno dei programmi gestionali su cui l'ambiente di sviluppo, l'amministrazione e altri reparti dell'azienda lavora è memorizzato su un server collocato all'interno della stessa ditta. Questo server contiene, come già accennato, il software applicativo oltre al database al quale questo si interfaccia.
Ogni sera, ad un'ora stabilita, viene lanciata una procedura automatizzata che si occupa di effettuare un backup dell'intero database e questa copia viene memorizzata nello stesso server aziendale dove si trova anche il database e il software gestionale. Il "computer centrale" è composto da due hard disk: uno in cui c'è caricato il sistema operativo e un altro riservato alla memorizzazione dei dati.
La mattina del 13 Marzo 2009 il server in questione ha iniziato a presentare alcune anomalie nel funzionamento; nel giro di poche ore lo stesso è risultato guasto e inutilizzabile. Subito è intervenuta la società che ha installato il server e che ne garantisce l'assistenza; questa, dopo aver prelevato l'elaboratore dall'azienda, ha effettuato alcuni test e ha diagnosticato la rottura di un disco rigido. Il timore al momento di questa comunicazione era che il disco danneggiato fosse quello in cui risiedevano tutti i dati ma fortunatamente accertamenti successivi hanno stabilito che l'hard disk contenente le informazioni aziendali era integro e funzionante mentre era stato irrimediabilmente danneggiato il disco di sistema.
La società di assistenza ha provveduto quindi a sostituire l'hard disk guasto e procedere alla riconfigurazione del server.

Questo fatto, però, ha comportato per l'azienda presso cui ho effettuato lo stage l'impossibilità di operare con uno dei suoi gestionali e di avere a disposizione i dati finchè non fosse stata completata la procedura di riparazione del server.
Questa esperienza, seppur vissuta da "spettatore", mi è stata utile perchè mi ha insegnato che sicuramente i backup sono molto importanti per avere delle copie del database ma queste "copie di sicurezza" dovrebbero essere custodite in posti sicuri e diversi da quelli in cui è memorizzato il database stesso per evitare il rischio di perdere tutti i dati disponibili in caso di gusto della macchina.

6.2 Un videocast con dei partner aziendali

In data 10 Marzo 2009 ho avuto la possibilità di assistere ad un videocast presso l'azienda dove ho effettuato la stage; questo incontro prevedeva la partecipazione di sette utenti e lo scopo era quello di fornire loro informazioni circa le novità principali introdotte dal recente rilascio di una service pack del software "Vision ERP" prodotto e commercializzato dall'azienda "Vision software house".
I partner che hanno partecipato a questo meeting provenivano da tutta Italia; più nello specifico c'erano due utenti dalla Sicilia, due dalla Campania, due dalla Puglia e uno dal Friuli Venezia Giulia.
E' stato possibile realizzare il videocast sfruttando un software "ad hoc" per la trasmissione di informazioni audio-video sul web denominato "skype" utilizzato insieme ad un altro programma che ha permesso di condividere il desktop del computer nell'azienda di Pernumia con tutti gli utenti che partecipavano alla videoconferenza.
Nelle fasi iniziali dell'incontro si è provveduto ad effettuare il collegamento con i partner; in prima battuta questo è sembrato un po' difficoltoso a causa di problemi di connessione ma ad un secondo tentativo tutto si è sistemato e il collegamento è riuscito. Da quel momento l'azienda e i partecipanti alla conferenza potevano conversare via internet e condividere lo stesso desktop. E' così risultato che i sette utenti avevano la

possibilità di vedere in tempo reale tutte le operazioni che venivano svolte sul computer della ditta di Pernumia.
Senza soffermarmi sul contenuto dell'incontro, molto centrato sull'aspetto economico, posso dire che questa esperienza mi ha permesso di vedere l'applicazione in un contesto aziendale di tecnologie usate per la comunicazione a distanza che mai avevo avuto l'occasione di vedere sperimentate nella pratica.

7. Conclusioni

Giunti al termine di questa relazione mi sembra doveroso tirare le somme sull'esperienza svolta. In particolare l'analisi finale è da condurre comparando le mie aspettative iniziali al lavoro svolto durante la permanenza in azienda.
In quest'ottica, ribadisco che il mio desiderio prima di iniziare il progetto era quello di poter vedere applicati ad un contesto lavorativo strumenti studiati in ambito universitario; in particolare, dato il corso di laurea intrapreso, mi interessava poter usare dei procedimenti statistici per trarre conclusioni su dati rilevati mediante strumenti informatici.
Durante lo stage ho avuto modo di scontrarmi con diverse problematiche presenti nell'ambito delle basi di dati; alcuni di questi erano già noti grazie alla frequenza dei corsi all'università, altri invece risultavano un po' oscuri. Tuttavia, grazie alle conoscenze di base possedute, è stato sufficiente un breve lavoro di documentazione e analisi di esempi per riuscire a comprendere e operare anche con questi nuovi elementi.
La fase di test, poi, mi ha permesso di mettere a frutto alcune conoscenze di inferenza statistica ma anche di scontrarmi con delle problematiche emerse durante le analisi e di dover quindi attivarmi per trovare delle soluzioni.
Una considerazione finale è rivolta anche al mio tutor aziendale che si è sempre dimostrato molto attento e disponibile nei miei confronti, pur dovendo gestire i mille impegni e problemi che quotidianamente interessano un'azienda.
Il giudizio complessivo sull'esperienza di stage, considerando anche l'arricchimento teorico e tecnico che se ne è ottenuto, è quindi molto buono. Il periodo di permanenza in "Vision Software House" mi ha permesso di acquisire delle conoscenze e delle abilità operative che mi auguro possa ulteriormente applicare e raffinare in un futuro contesto lavorativo.

8. Appendice

8.1 Disegno dei diagrammi di database

Il disegno diagrammi di database è stato effettuato mediante lo strumento “SQL Server Management”.

Per qualsiasi database è possibile creare il numero desiderato di diagrammi. Si possono inoltre disegnare diagrammi diversi per rappresentare differenti parti della base di dati o per sottolineare alcuni aspetti della progettazione. Ad esempio, è possibile creare un diagramma di grandi dimensioni che mostri tutte le tabelle e le rispettive relazioni e una rappresentazione più ridotta che mostri una o alcune tabelle in particolare.

Ciascun diagramma creato viene archiviato nel database associato. In particolare una volta avviata la console “Management Studio” e selezionato il database di interesse è possibile vedere la cartella “Database diagrams” che, se espansa, visualizza tutti i diagrammi disponibili per la base di dati in oggetto (figura 8.1).

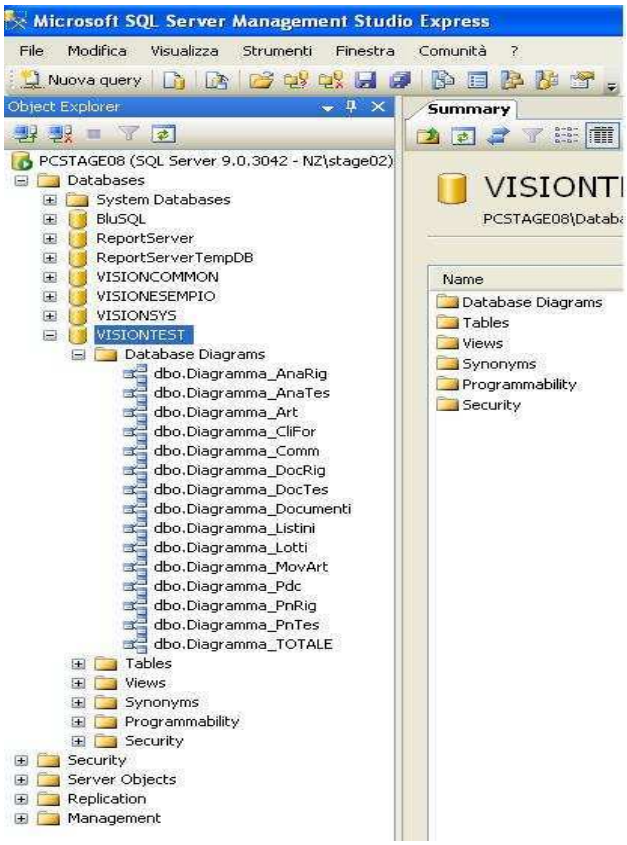

Figura 8.1: Visualizzare i diagrammi di database con Management Studio.

Creare un nuovo diagramma

Per procedere alla creazione di un nuovo diagramma è sufficiente fare click con il tasto destro sopra alla cartella Database diagrams, quindi selezionare “New Database Diagram”. Verrà presentata a video una schermata in cui è possibile selezionare le tabelle da inserire nel diagramma.

RAPPRESENTARE I VINCOLI DI INTEGRITA’ REFERENZIALE

Per visualizzare la presenza di un vincolo di integrità referenziale nel diagramma di database è sufficiente inserire le tabelle coinvolte in tale relazione. Infatti, se vengono inserite nel diagramma due tabelle che presentano un legame di foreign key, “Management studio” disegna automaticamente una linea continua di colore grigio volta a rappresentare tale vincolo. Facendo doppio click sopra alla linea creatasi se ne possono vedere alcune proprietà per esempio gli attributi che sono messi in

corrispondenza, il nome della relazione, le eventuali politiche da adottare in caso di modifica o cancellazione, ecc.

Prima di procedere con il disegno dei diagrammi ho effettuato una "mappatura" del database in particolare mi sono trascritto tutti i legami di foreign key presenti in modo che poi risultasse semplice individuare le tabelle con vincoli di integrità referenziale da inserire nel diagramma.

" Gli endpoint della linea indicano se la relazione è di tipo uno-a-uno o uno-a-molti. Quando una relazione presenta una chiave su un endpoint e un simbolo di infinito sull'altro, rappresenta una relazione uno-a-molti. Quando una relazione presenta una chiave su ciascun endpoint, rappresenta una relazione uno-a-uno."

[FONTE: http://technet.microsoft.com/it-it/library/ms188251(SQL.90).aspx]

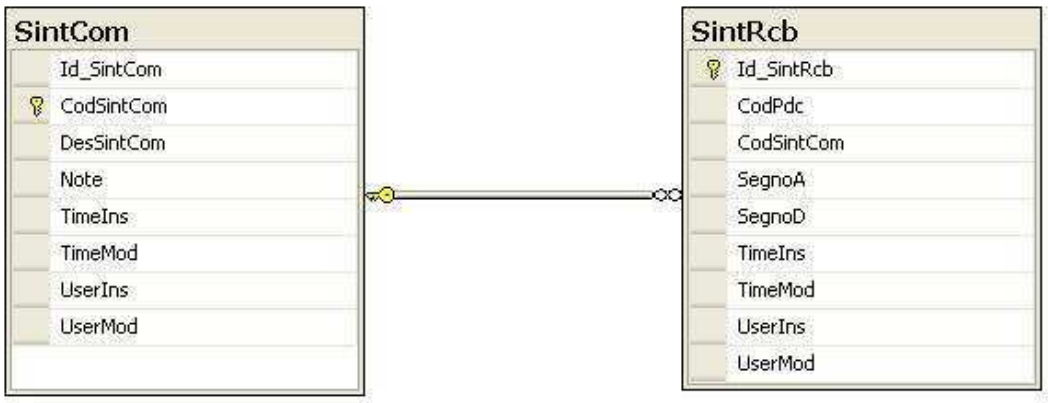

Figura 8.2: Relazione uno-a-molti fra due tabelle.

RAPPRESENTARE LE AZIONI DEI TRIGGER

Trovare un modo per rappresentare con delle linee i controlli sui campi delle tabelle effettuati da un trigger di cancellazione e/o modifica ha richiesto la ricerca di

documentazione in quanto sui manuali presenti in azienda non veniva accennato alla gestione di queste relazioni.

Dando uno sguardo alla documentazione in linea fornita da microsoft su SQL server 2005 mi sono accorto che la linea di congiunzione fra due tabelle poteva avere due stili diversi con differente significato:

“ Stile linea : La linea stessa indica se nel Sistema di gestione di database (DBMS, Database Management System) viene attivata l'integrità referenziale per la relazione quando vengono aggiunti nuovi dati alla tabella chiave esterna. Se la linea è continua, nel sistema DBMS l'integrità referenziale per la relazione verrà attivata quando vengono aggiunte o modificate alcune righe nella tabella chiave esterna. Se la linea è tratteggiata, nel sistema DBMS l'integrità referenziale per la relazione non verrà attivata quando vengono aggiunte o modificate alcune righe nella tabella chiave esterna. “

[FONTE: http://technet.microsoft.com/it-it/library/ms188251(SQL.90).aspx]

Da qui, quindi, l’idea di rappresentare tutti i controlli sulle tabelle effettuati dai trigger mediante linee tratteggiate a indicare il fatto che in quel caso non viene attivato il vincolo di integrità referenziale dal DBMS, ma viene semplicemente messa in evidenza una relazione esistente fra due o più attributi di tabelle diverse.

Per fare in modo che venga visualizzata la linea tratteggiata nella relazione fra due tabelle è necessario individuare gli attributi che devono partecipare alla relazione quindi cliccare sopra ad uno di essi e trascinarlo fino all’altro. Si presenterà automaticamente a video una schermata in cui è possibile inserire un nome per la relazione e impostare la non attivazione del vincolo di integrità referenziale per il legame in oggetto.

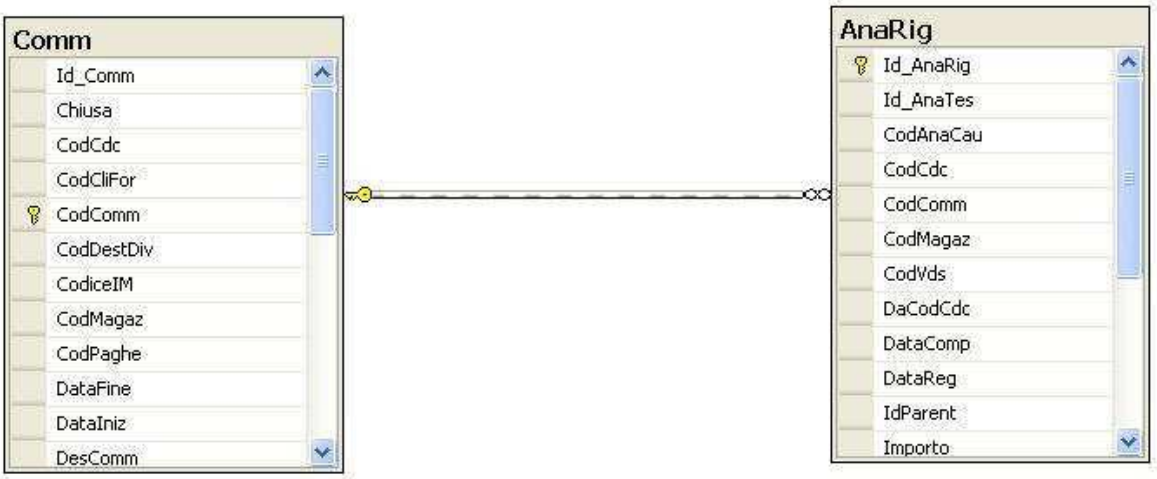

Figura 8.3: Relazione fra due tabelle imposta da un trigger.

8.2 Esportare ed importare i diagrammi di database

8.2.1 Esportare i diagrammi

In SQL server 2005 i diagrammi di database creati vengono memorizzati nella tabella di sistema *"sysdiagrams"*. Eseguendo una query di selezione sulla tabella in questione si può notare che ogni tupla del risultato contiene una corrispondenza con un diagramma di database creato.

Per riuscire ad esportare i riferimenti ai diagrammi occorre utilizzare un'utility denominata "bcp" e fornita insieme a SQL Server; questa è uno strumento richiamabile dalla riga di comando che permette la copia di massa di dati tra un database di SQL Server 2005 e un file di dati in un formato specificato dall'utente.

Accedendo al prompt dei comandi (start->programmi->accessori->prompt dei comandi) per effettuare l'esportazione dei diagrammi è sufficiente eseguire:

```
-------------------------------------------------------------
bcp "SELECT * FROM NomeDatabase.dbo.sysdiagrams" queryout
Percorso_file -T –n
-------------------------------------------------------------
```

Dove:

Bcp = E' nome dell'utility richiamata

*"SELECT * FROM Nomedatabase.dbo.sysdiagrams"* = indica in quale tabella ricercare i dati

queryout = esegue la copia da una query. Questo argomento deve essere specificato solo nel caso di operazioni di copia di massa di dati da una query.

Percorso_file= Percorso completo del file di dati. Quando si esegue un'esportazione di massa di dati da SQL Server, il file di dati include i dati copiati dalla tabella . Il percorso può essere composto da 1 a 255 caratteri. Il file di dati può contenere un massimo di 2.147.483.647 righe.

-T = Specifica che l'utilità bcp si connette a SQL Server con una connessione trusted (sicura) utilizzando la protezione integrata. Non è necessario specificare le credenziali di protezione dell'utente di rete, ovvero *login* e *password.*

-n = Esegue l'operazione di copia di massa utilizzando i tipi di dati nativi del database. Con questa opzione non viene visualizzata una richiesta per ogni campo, ma vengono utilizzati i valori nativi.

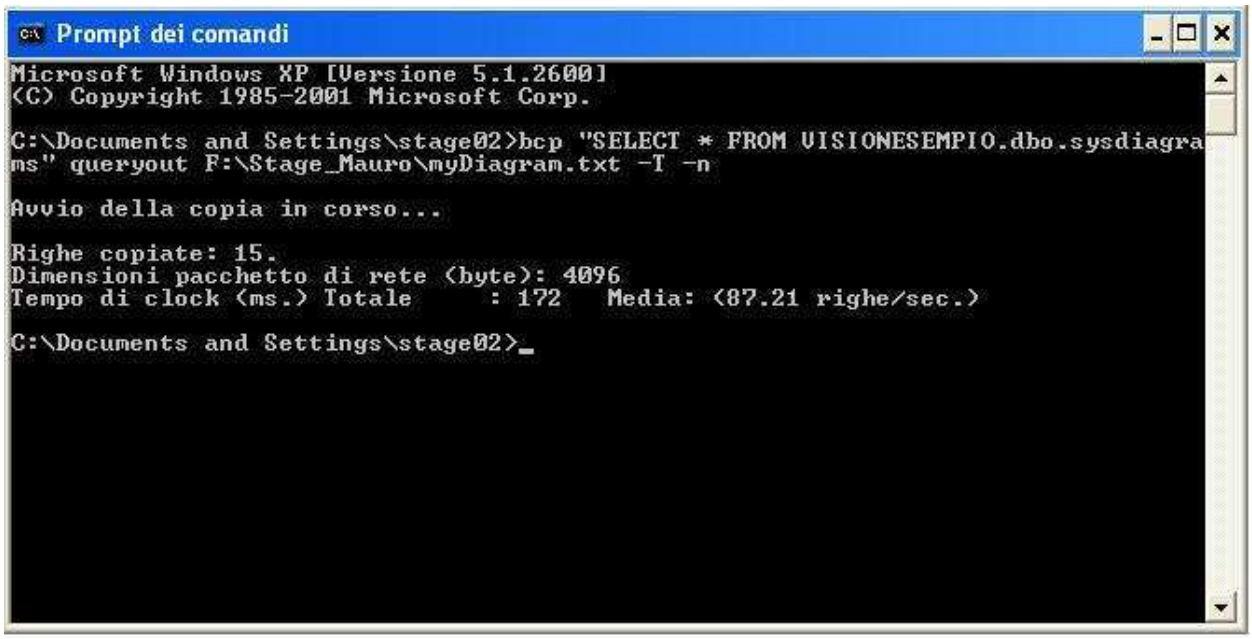

Figura 8.4: Esecuzione del comando per esportare i diagrammi di database.

8.2.2 Importare i diagrammi

Per importare i diagrammi viene fatto uso nuovamente dell'utilità bcp da riga di comando. Questa volta l'istruzione da eseguire è la seguente:

```
bcp NomeDatabase.dbo.sysdiagrams in Persorso_file -T –n
```

Dove:

Bcp = E' nome dell'utility richiamata

`NomeDatabase.dbo.sysdiagrams`= indica la tabella sulla quale importare i dati.

in = esegue la copia da un file in una tabella del database.

Percorso_file = Percorso completo del file di dati. Quando si esegue un'importazione di massa di dati in SQL Server, il file di dati

include i dati da copiare nella tabella o nella vista specificata. Il percorso può essere composto da 1 a 255 caratteri. Il file di dati può contenere un massimo di 2.147.483.647 righe.

-T = Specifica che l'utilità bcp si connette a SQL Server con una connessione trusted (sicura) utilizzando la protezione integrata. Non è necessario specificare le credenziali di protezione dell'utente di rete, ovvero *login* e *password.*

-n = Esegue l'operazione di copia di massa utilizzando i tipi di dati nativi del database. Con questa opzione non viene visualizzata una richiesta per ogni campo, ma vengono utilizzati i valori nativi.

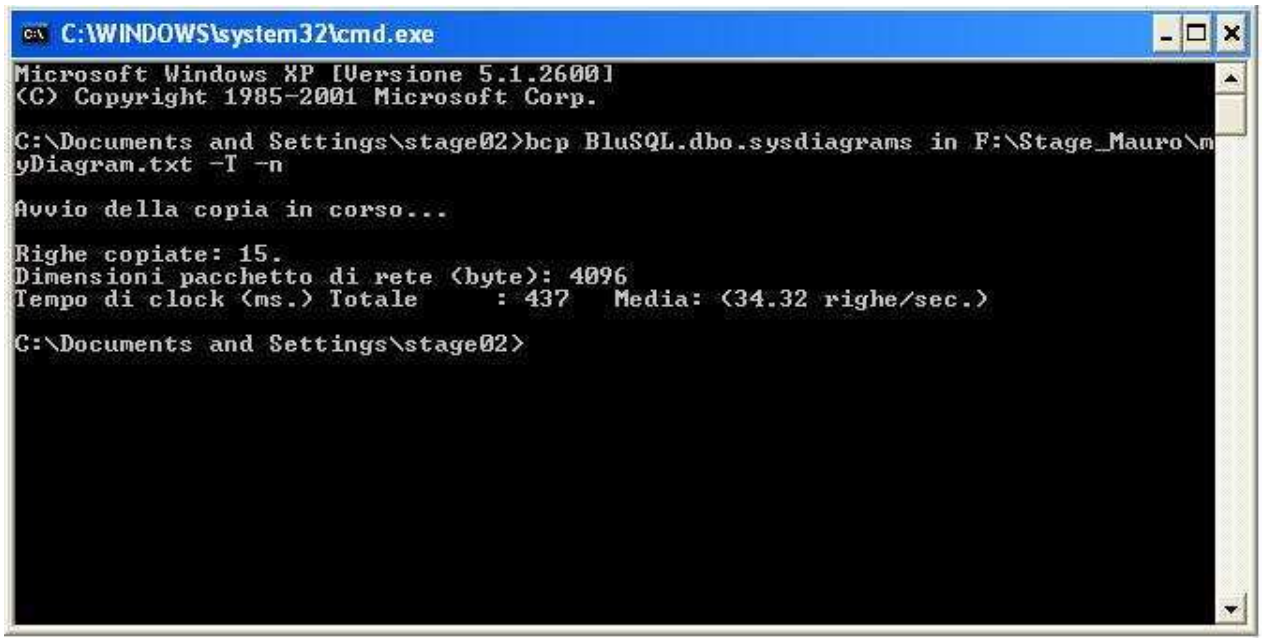

Figura 8.5: Esecuzione del comando per importare i diagrammi di database

[FONTI: http://groups.google.it/group/microsoft.public.it.sql/msg/2aa9843f9d9c6571

http://msdn.microsoft.com/it-it/library/ms162802(SQL.90).aspx]

8.3 Analisi statistiche su una query

```
--------------------------------------------------------------------------------
SELECT * FROM DocRig
         WHERE DocRig.DataDoc = convert(smalldatetime,'07/07/2004')
--------------------------------------------------------------------------------
```

- Analisi tempi di risposta <u>in assenza</u> dei diagrammi di database:

Prova	Tempi esecuzione (y_i)	scarti dalla media $(y_i - \overline{y})$	scarti dalla media al quadrato $(y_i - \overline{y})^2$
1	1,84	0,0873	0,0076
2	1,75	-0,0027	0,0000
3	1,96	0,2073	0,0430
4	1,87	0,1173	0,0138
5	2,22	0,4673	0,2184
6	1,66	-0,0927	0,0086
7	1,9	0,1473	0,0217
8	1,68	-0,0727	0,0053
9	1,68	-0,0727	0,0053
10	1,76	0,0073	0,0001
11	1,47	-0,2827	0,0799
12	1,93	0,1773	0,0314
13	1,67	-0,0827	0,0068
14	1,57	-0,1827	0,0334
15	1,72	-0,0327	0,0011
16	1,62	-0,1327	0,0176
17	1,7	-0,0527	0,0028
18	1,48	-0,2727	0,0743
19	1,81	0,0573	0,0033
20	1,64	-0,1127	0,0127
21	1,63	-0,1227	0,0150
22	1,67	-0,0827	0,0068
23	1,64	-0,1127	0,0127
24	1,72	-0,0327	0,0011
25	1,78	0,0273	0,0007
26	1,63	-0,1227	0,0150
27	1,67	-0,0827	0,0068
28	1,61	-0,1427	0,0204
29	2,34	0,5873	0,3450
30	1,96	0,2073	0,0430

Media campionaria:

n = n° osservazioni del campione

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = 1{,}7527$$

Varianza campionaria:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} (y_i - \bar{y})^2 = 0{,}0363$$

Scarto quadratico medio campionario:

$$s = \sqrt{s^2} = 0{,}1906$$

Ricerca del percentile di una distribuzione "t di Student":

$$t_{0{,}975}(29) = 2{,}05$$

Intervallo di confidenza per la media campionaria di livello 0.95:

$$\left[\bar{y} - \frac{s * t_{1-\frac{\alpha}{2}}(n-1)}{\sqrt{n}}; \bar{y} + \frac{s * t_{1-\frac{\alpha}{2}}(n-1)}{\sqrt{n}}\right] = [1.6813;1.8240]$$

- Analisi tempi di risposta <u>con</u> i diagrammi di database:

Prova	Tempi esecuzione (x_i)	scarti dalla media $(x_i - \bar{x})$	scarti dalla media al quadrato $(x_i - \bar{x})^2$
1	1,91	0,1153	0,0133
2	1,67	-0,1247	0,0155
3	1,35	-0,4447	0,1977
4	1,78	-0,0147	0,0002
5	1,9	0,1053	0,0111
6	1,67	-0,1247	0,0155
7	1,97	0,1753	0,0307
8	1,68	-0,1147	0,0131
9	1,75	-0,0447	0,0020
10	1,83	0,0353	0,0012
11	2,1	0,3053	0,0932
12	1,93	0,1353	0,0183
13	1,82	0,0253	0,0006
14	1,48	-0,3147	0,0990
15	1,77	-0,0247	0,0006
16	1,52	-0,2747	0,0754
17	1,94	0,1453	0,0211
18	1,92	0,1253	0,0157
19	1,79	-0,0047	0,0000
20	1,5	-0,2947	0,0868
21	1,74	-0,0547	0,0030
22	1,79	-0,0047	0,0000
23	1,76	-0,0347	0,0012
24	2,06	0,2653	0,0704
25	2,05	0,2553	0,0652
26	1,99	0,1953	0,0382
27	1,53	-0,2647	0,0700
28	1,9	0,1053	0,0111
29	2,09	0,2953	0,0872
30	1,65	-0,1447	0,0209

Media campionaria:

m = n° osservazioni del campione

$$\bar{y} = \frac{1}{m}\sum_{i=1}^{n} x_i = 1{,}7947$$

Varianza campionaria:

$$s^2 = \frac{1}{m-1}\sum_{i=1}^{n} (x_i - \bar{x})^2 = 0{,}0372$$

Scarto quadratico medio campionario:

$$s = \sqrt{s^2} = 0{,}1929$$

Ricerca del percentile di una distribuzione "t di Student"

$$t_{0{,}975}(29) = 2{,}05$$

Intervallo di confidenza per la media campionaria di livello 0.95:

$$\left[\bar{y} - \frac{s * t_{1-\frac{\alpha}{2}}(m-1)}{\sqrt{m}}; \bar{y} + \frac{s * t_{1-\frac{\alpha}{2}}(m-1)}{\sqrt{m}}\right] = [1.7525; 1.8669]$$

- Applicazione del test "t di Student a due campioni".

Indicando con μ e η le medie campionarie dei due gruppi di osservazioni, interessa verificare il sistema di ipotesi:

$$\begin{cases} H_0 : \mu = \eta \\ H_1 : \mu \neq \eta \end{cases}$$

Assumo le varianze campionarie uguali nei due gruppi. Calcolo quindi la varianza totale:

$$s^2 = \frac{1}{n+m-2}\left[\sum_{i=1}^{n}(y_i - \bar{y})^2 + \sum_{i=1}^{m}(x_i - \bar{x})^2\right] = 0{,}0368$$

Statistica test:

$$t_{oss} = \frac{\bar{y} - \bar{x}}{s\sqrt{\frac{1}{n} + \frac{1}{m}}} = 0{,}8484$$

Regola di tipo accetto/rifiuto:

Se $\left|t_{oss}\right| \leq t_{1-\frac{\alpha}{2}}(58) \Rightarrow 0{,}8484 \leq 1{,}96 \Rightarrow$ Accetto

Livello di significatività osservato:

$$2[1 - P(t(n+m-2) <= \mid t_{oss} \mid)] = 2[1 - P(t(58) <= 0{,}8484)] \cong 0{,}4$$

9. Elenco delle fonti

- Gunderloy, L.Jorden, W. Tschanz, "*SQL Server 2005*", SYBEX.

- Andy Oppel, "*SQL Tecniche e soluzioni*", McGraw-Hill.

- *Lucidi delle lezioni di inferenza statistica* I, Guido Masarotto.

- Atzeni, Ceri, Paraboschi, Torlone, "*Basi di dati - seconda edizione*", McGraw-Hill.

- http://msdn.microsoft.com/it-it/library: Documentazione in linea sui prodotti microsoft.

- http://groups.google.it/group/microsoft.public.it.sql: Procedura per l'import-export dei diagrammi e per lo svuotamento della memoria cache nel server.

- http://www.vsh.it: Sito web dell'azienda "Vision Software House".