UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI SCIENZE STATISTICHE

CORSO DI LAUREA IN STATISTICA E TECNOLOGIE INFORMATICHE

RELAZIONE DI LAUREA

Analisi ed ottimizzazione del database "Vision SQL"; stage presso l'azienda "Vision Software Gestionale".

Analysis and optimization of the "Vision SQL" database; stage at "Vision Software Gestionale" company.

Relatore:Prof. MASSIMO MELUCCI

Laureando: Pegoraro Marco

Matricola: 579205-STI

ANNO ACCADEMICO 2009/2010

INDICE

TEST E ANALISI DELLE PROCEDURE REALIZZATE

1. Introduzione

Questa tesi nasce da un'esperienza di stage effettuata presso un'azienda produttrice di software gestionale per piccole e medie imprese. La scelta di includere un' esperienza di stage nel percorso di studi è derivata da una voglia di avere un contatto diretto con il mondo del lavoro e vedere come ciò che ho studiato finora si integra nella realtà aziendale. Durante il percorso di studi, infatti, si apprendono moltissimi concetti nuovi ma talvolta può essere molto difficile riuscire ad inquadrarli in un contesto lavorativo e quindi un'esperienza di lavoro può risultare davvero utile a tale scopo. Ritengo quindi che la possibilità di compiere un'esperienza di stage durante la carriera universitaria di uno studente sia un modo per avere una conferma che quanto fatto finora sia servito per il proprio futuro e la futura carriera lavorativa.

Anticipo fin da subito che mi ritengo pienamente soddisfatto dell'esperienza che ho avuto, sia perché mi è piaciuto il progetto che ho portato a termine, sia per l'ambiente lavorativo che ho trovato; infatti credo che uno stage possa essere utile, oltre che per vedere applicato ciò che da prima era solo teorico, anche perché permette di conoscere la realtà lavorativa, molto diversa da quella scolastica.

La prima parte di questa relazione riguarderà la presentazione dell'azienda e dei principali prodotti che commercializza. In seguito analizzerò in dettaglio le nuove conoscenze che ho potuto acquisire grazie a questa esperienza, cercando di dare una breve introduzione teorica che permetta di inquadrare l'ambito nel quale si colloca il lavoro che ho svolto e che metta in luce i problemi che ho dovuto affrontare.

2. Presentazione dell'azienda e del progetto di stage

2.1. Presentazione dell'azienda

L'azienda presso la quale ho svolto lo stage si chiama "Vision Software Gestionale" ed ha sede a Pernumia, in provincia di Padova.

Si tratta di un'azienda con un'esperienza quasi trentennale che opera nel campo dell'informatica e più precisamente di quella parte dell'informatica dedicata alle applicazioni gestionali per piccole e medie imprese. Produce quindi software che vengono utilizzati nelle aziende quotidianamente per la contabilità, la gestione dei clienti e dei fornitori, la gestione dei dipendenti, il magazzino e tutti i possibili aspetti della gestione aziendale. Lo spirito con cui lavora Vision è quello di sviluppare software per aziende competitive che credono nell'importanza dell'organizzazione e della gestione aziendale per raggiungere i loro obiettivi. Per questo i loro software non si limitano alla semplice gestione della contabilità ma permettono di gestire ogni aspetto della realtà dell'azienda.

La linea di prodotti che commercializza è fatta su misura per le aziende, in quanto è una linea completa che comprende prodotti diversificati a seconda del settore di lavoro dell'azienda e anche della sua dimensione. Una piccola azienda artigianale, commerciale o di servizi avrà infatti delle necessità molto diverse rispetto ad un'azienda industriale medio grande; come del resto un software gestionale pensato per un'azienda commerciale non potrà essere ugualmente utile per un'azienda di autotrasporti o per una che opera nell'edilizia. Per questo "Vision Software Gestionale" ha ideato una linea di prodotti molto variegata che possa coprire la maggior parte delle realtà lavorative aziendali.

Il software principale su cui investe l'azienda si chiama "Vision" ed è stato concepito come software internazionale in quanto, grazie alla funzionalità "Make Your Language by Yourself", è possibile modificarne l'interfaccia nella lingua desiderata.

Partendo da “Vision”, poi, sono stati costruiti tutta una serie di prodotti per adattare il software il più possibile ai diversi settori nei quali operano le aziende. Tra questi ci sono:

- Vision Energy: è un software per la gestione contabile, finanziaria, commerciale e di magazzino per le aziende di distribuzione all'ingrosso di prodotti petroliferi e gas;
- Vision Fresh: è un software gestionale per il settore ortofrutticolo e ittico;
- Vision Autotrasporti: è un software gestionale per autotrasportatori che si occupano di trasporti di merci o persone;
- Vision Pref: è un software gestionale per le aziende che producono prefabbricati per l’edilizia;
- Vision Mobile: è un software gestionale per palmari, adatto a chi ha bisogno di operare in mobilità;
- Vision OnWeb: è un software che permette alle aziende di vendere i propri prodotti sul web;
- Vision SQL: è un software gestionale dedicato alle medie aziende industriali che gestiscono grandi quantità di dati e necessitano di analisi dei dati in tempo reale che permettano di prendere decisioni importanti in tempi rapidi. E’ un software di alto livello che permette di gestire tutti gli aspetti della gestione aziendale.

2.2. Presentazione del progetto

Il progetto di stage è stato definito in maniera molto chiara fin dall’inizio, infatti già durante il primo incontro avuto con il tutor aziendale abbiamo discusso e scritto gli obiettivi principali e secondari dello stage. In seguito abbiamo definito anche un calendario delle attività, in modo tale da organizzare nel migliore dei modi tutto il periodo di lavoro in azienda.

Riporto di seguito una presentazione dello stage, dapprima evidenziandone gli obiettivi e poi descrivendo tutti i temi che sono stati trattati. Rimando ai successivi capitoli per una spiegazione più dettagliata degli stessi.

L’obiettivo principale è stato duplice:

- realizzare i trigger riguardanti il controllo dei dati durante la cancellazione, con lo scopo di migliorare la qualità dei dati e la loro integrità;
- attività di test delle procedure realizzate, misurazione dei tempi di risposta del database a fronte dell’introduzione dei nuovi trigger e documentazione dei trigger implementati tramite la creazione di diagrammi del database.

L’obiettivo secondario dello stage è stato poi quello di realizzare trigger che garantissero l’integrità dei dati anche in fase di aggiornamento, oltre che in fase di cancellazione. Fin da subito è stato deciso che questo obiettivo si sarebbe dovuto completare solo nel caso in cui si fosse completato l’obiettivo principale in tempi minori del previsto, cosa che è avvenuta e quindi sono riuscito a portare a temine tutti e due gli obiettivi. Inoltre al termine dello stage mi è stato chiesto di presentare una relazione e un manuale tecnico al reparto di programmazione nel quale fosse descritto dettagliatamente il lavoro svolto.

Relativamente agli obiettivi descritti sopra è necessario specificare che il software che mi è stato fornito, ossia Microsoft SQL Server, include specifici strumenti atti a disegnare diagrammi in cui sono collegate tramite chiavi le tabelle del database. Inoltre il database sul quale ho lavorato disponeva già di alcuni trigger, quindi l’obiettivo è stato quello di implementare in modo completo l’uso di trigger per garantire una migliore coerenza dei dati.

La parte iniziale del progetto di stage ha riguardato dunque la creazione dei diagrammi del database di “VisionSQL”. L’obiettivo di questo lavoro è stato duplice. Da un lato l’azienda aveva la necessità che qualcuno provvedesse a creare i diagrammi, così da avere a disposizione uno strumento visivo che permettesse di capire tutte le relazioni tra le tabelle, sia perché anche il personale meno competente potesse capire quali sono le relazioni esistenti, sia come materiale di supporto per i programmatori che si occupano dello sviluppo del database. Da un altro lato lo sviluppo dei diagrammi doveva essere utile a

me per capire a fondo come fosse strutturato il database, in modo tale da poter portare a termine l'obiettivo dello stage, cioè la creazione di trigger che assicurassero un maggiore controllo durante la cancellazione dei dati e quindi una maggiore qualità complessiva della base di dati. Bisogna dire che alcuni diagrammi erano già stati creati in precedenza ma da allora, essendo "VisionSQL" un software in continua crescita, erano state introdotte nuove tabelle e nuove relazioni, per cui anche i diagrammi che erano già stati fatti avevano bisogno di essere aggiornati, e oltre a questi mi è stato chiesto di crearne degli altri. È necessario puntualizzare che normalmente la creazione dei diagrammi è il primo passo da compiere quando si vuole costruire una base di dati. In ogni corso di basi di dati, infatti, si insegna che solo quando lo schema è stato completato si può procedere fisicamente alla creazione del database. Quello che ho dovuto fare io invece è stato esattamente il procedimento inverso, cioè capire quale fosse lo schema della base di dati a posteriori. Non nascondo che non è stato un lavoro molto semplice, sia a causa delle dimensioni del database, sia perché l'unico metodo per risalire alle relazioni è stato quello di controllare tutte le tabelle e risalire a quali fossero le chiavi esterne. Per risolvere questi problemi mi sono state davvero utili le conoscenze teoriche acquisite durante i corsi di basi di dati, anche se, come spiegato in precedenza, non ho dovuto applicarle nella maniera "canonica". Un altro problema poi è stato quello di capire cosa rappresentassero le tabelle del database; infatti, essendo "VisionSQL" un software gestionale, sono entrato in contatto con la terminologia riguardante l'ambito della gestione aziendale e la contabilità. Fortunatamente, grazie all'aiuto del tutor aziendale ed anche a quello di altri colleghi, sono riuscito a capire tutto ciò di cui avevo bisogno.

Tutta la fase di studio del software "VisionSQL", della base di dati e della creazione di alcuni diagrammi è stata svolta assieme ad un'altra stagista, quindi inizialmente ho potuto sperimentare il lavoro "in team" e devo ammettere che è stato utile perché abbiamo potuto avere degli scambi di opinioni che ci hanno portato a capire meglio e più in fretta la struttura della base di dati.

Dopo questa prima fase di analisi del database e creazione dei diagrammi ho potuto procedere alla mappatura dei trigger presenti e mancanti, per poi creare dapprima i trigger di cancellazione, con la relativa documentazione, e poi creare i trigger di aggiornamento. Durante questa fase ho dovuto imparare il linguaggio utilizzato per la creazione dei trigger, chiamato T-SQL, acronimo di Transact-SQL. Si tratta di un'estensione proprietaria del linguaggio SQL, utilizzata da Microsoft SQL Server. Per la creazione dei trigger mi sono stati molto utili i diagrammi creati durante la prima fase dello stage, i quali mi hanno agevolato nel mappare i trigger già presenti e quelli da implementare. In seguito alla creazione dei trigger ho proceduto a riportare nei diagrammi le scritte "trigger di cancellazione" e "trigger di aggiornamento" dove questi fossero stati implementati. Così facendo ho creato una documentazione grafica del lavoro svolto, la quale potrà essere d'aiuto al team di sviluppo del database. Oltre a questa documentazione ho proceduto a crearne una più tecnica che contenesse la mappatura di tutte le relazioni del database e nella quale fosse specificato quali trigger fossero già presenti, quali fossero stati creati da me e quali non fossero necessari.

Terminata la prima parte del progetto ho proceduto con l'attività di test e di comparazione dei tempi di risposta del database in presenza e in assenza dei trigger. Questa seconda parte dello stage è stata particolarmente utile per vedere applicati all'informatica alcuni concetti di base della statistica, come la creazione di un campione di osservazioni che rispetti gli assunti statistici necessari per l'applicazione di test, il calcolo di intervalli di confidenza e la verifica di ipotesi sull'uguaglianza delle medie di due popolazioni.

Ora che ho descritto tutte le attività da me svolte durante il periodo di permanenza in azienda, procedo con una spiegazione più teorica degli argomenti, al termine della quale risulterà sicuramente più chiara l'utilità del lavoro che ho svolto.

IL DATABASE DI VISIONSQL

3. Analisi della struttura del database

3.1. Il DBMS

"Un sistema di gestione di basi di dati (in inglese Data Base Management System, abbreviato con DBMS) è un sistema software in grado di gestire collezioni di dati che siano grandi, condivise e persistenti, assicurando loro affidabilità e privatezza".

"Una base di dati è una collezione di dati gestita da un DBMS".

[Fonte: P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone (2002), Basi di dati, McGraw-Hill]

Analizziamo più in dettaglio le definizioni appena introdotte.

- Le basi di dati possono essere grandi, nel senso che possono avere delle dimensioni davvero considerevoli. Si può affermare che al momento l'unico limite di grandezza di una base di dati è dato dalle dimensioni del disco sulla quale risiede.
- Una base di dati deve essere condivisa, cioè devono poterci accedere diverse applicazioni e diversi utenti; ciò è molto importante perché se esistessero varie copie degli stessi dati, queste potrebbero essere differenti tra loro; viceversa se i dati sono memorizzati in un calcolatore in modo univoco, non è possibile incorrere in disallineamenti.
- Le basi di dati si definiscono persistenti perché non hanno un tempo di vita limitato a quello delle singole esecuzioni dei programmi che le utilizzano. Al contrario, i dati gestiti da un programma in memoria centrale hanno una vita che inizia e termina con l'esecuzione del programma.

- Il DBMS deve garantire affidabilità, ossia deve conservare il contenuto del database anche in caso di malfunzionamenti hardware e software. A questo scopo quindi deve permettere di effettuare backup e restore.
- Il DBMS deve anche garantire la privatezza dei dati, cioè deve poter garantire privilegi diversi a seconda dell'utente che accede alla base di dati.

3.1.1. Microsoft SQL Server e SQL Management Studio 2005

Microsoft SQL Server è un DBMS prodotto da Microsoft. Nelle prime versioni era utilizzato per basi di dati medio-piccole, ma a partire dalla versione 2000 è stato utilizzato anche per la gestione di basi di dati di grandi dimensioni. L'ultima versione rilasciata di questo software è "SQL Server 2008" ma durante la mia esperienza di stage ho utilizzato la versione "SQL Server 2005".

Più in particolare esistono diverse versioni di SQL Server 2005, tre delle quali a pagamento ed una gratuita. Naturalmente la versione gratuita, chiamata SQL Server 2005 Express, ha alcune limitazioni rispetto alle versioni a pagamento; in particolare supporta l'utilizzo di una sola CPU, fino ad 1 Gigabyte di RAM e database della dimensione massima di 4 Gigabyte. È proprio questa la versione che ho utilizzato durante la prima parte della mia esperienza; per la fase di test, invece, ho utilizzato "Microsoft SQL Server 2005 Developer Edition".

SQL Server dispone inoltre di una console per la gestione grafica chiamata "SQL Server Management Studio"; questo programma permette di gestire tutte le fasi riguardanti la creazione, la gestione e la messa in sicurezza dei database.

Uno strumento molto importante presente in Management Studio è il piano di esecuzione. Tramite questo strumento possiamo valutare le prestazioni di una query che abbiamo creato e capire se questa è performante o meno. Una volta creata la query, premendo il pulsante piano di esecuzione, quest'ultima viene eseguita mostrando le statistiche di esecuzione. Verrà quindi mostrato in quale

percentuale hanno inciso le singole operazioni che il server ha dovuto effettuare per eseguire la query stessa.

3.2. Il modello relazionale e i suoi costrutti

Il modello dei dati relazionale è stato proposto da Codd nel 1970 ed è tra i più implementati nei sistemi di gestione di basi di dati (DBMS). Esso si basa sul concetto matematico di relazione, definita come sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \ldots \times D_n$ di n insiemi di valori di tipo elementare, detti domini, ovvero come insieme di ennuple ordinate di valori $\{d_1, d_2,\ldots,d_n\}$, con $d_i \in D_i$, per ogni i.

Una relazione R definita su $D_1 \times D_2 \times \ldots \times D_n$ è un sottoinsieme di $D_1 \times D_2 \times \ldots \times D_n$. La tabella è la rappresentazione grafica normalmente accettata per rappresentare la relazione.

Il grado della relazione è il numero n delle componenti del prodotto cartesiano (numero di domini), mentre la cardinalità della relazione è il numero delle sue tuple.

L'attributo è il termine usato nella teoria per definire il nome di una colonna della tabella, quindi è il nome dato ad un dominio in una relazione.

Dando quindi delle definizioni informali, che nella pratica vengono utilizzate quando il database viene creato a livello fisico, possiamo affermare che una relazione è una tabella, un attributo è una colonna, una tupla è una riga, il dominio è il tipo di dato, la cardinalità è il numero di righe, il grado è il numero di colonne.

3.2.1. Vincoli di integrità e chiavi

La struttura del modello relazionale è sicuramente molto semplice ma allo stesso tempo impone un certo grado di rigidità per garantire che non esistano in un certo momento delle istanze che non rappresentano correttamente il mondo applicativo. Per questo esistono i vincoli di integrità, che in sostanza stabiliscono quali sono i valori assumibili dagli attributi di ogni ennupla. I vincoli quindi possono essere visti come proprietà che devono essere rispettate dalle istanze perché queste possano essere considerate valide.

I principali vincoli di integrità sono quelli che specificano:

- quali attributi possono assumere valore nullo;
- quali attributi sono chiavi;
- quali attributi sono chiavi esterne.

I valori nulli

L'esigenza di ammettere nel dominio di definizione di un attributo il valore nullo deriva dal fatto che, per varie ragioni, può capitare che non sia possibile specificare il valore dell'attributo stesso. Questo accade molto frequentemente, ad esempio quando non si conosce il valore dell'attributo.

Le chiavi

I vincoli di chiave sono i più importanti del modello relazionale e garantiscono l'univocità di ogni istanza di una relazione.

Si possono definire tre tipi di chiavi:

- una superchiave è un sottoinsieme di attributi della relazione tale che in nessuna istanza valida della relazione possano esistere due ennuple diverse che coincidono su tutti gli attributi della superchiave. Una superchiave quindi identifica univocamente ogni ennupla.

- una chiave di una relazione è una superchiave minimale, nel senso che se si elimina un attributo, i rimanenti non formano più una superchiave
- una chiave primaria è una delle chiavi, di solito si preferisce quella con meno attributi.

È evidente che una chiave primaria non può ammettere valori nulli nei suoi attributi, altrimenti non garantirebbe l'identificazione univoca di tutte le tuple della relazione.

Si può notare che in molti casi reali è possibile selezionare uno o più attributi che per la loro natura identificano univocamente ogni tupla della relazione. Quando ciò non è possibile, è necessario introdurre un attributo aggiuntivo, un codice, probabilmente non significativo dal punto di vista dell'applicazione, che venga generato al momento dell'inserimento della tupla.

Le chiavi esterne

Le chiavi esterne sono indispensabili per rappresentare le associazioni nel modello relazionale. Il loro scopo è quindi quello di associare ad un'ennupla di una relazione quell'ennupla della relazione riferita che ha il valore della chiave primaria uguale al valore della chiave esterna.

Nella pratica, quando due tabelle vengono associate, la chiave primaria di una viene inserita nell'altra come chiave esterna e il vincolo che ne scaturisce sta nel fatto che l'insieme dei valori che la chiave esterna può assumere è l'insieme dei valori assunti dalla chiave primaria cui si riferisce. Il vincolo di chiave esterna viene definito anche vincolo di integrità referenziale o di foreign key.

3.3. Diagrammi di database

Nel database di VisionSQL si possono contare quasi duecento tabelle, tra le quali alcune contano più di sessanta campi. Per database di queste dimensioni ricordare a memoria tutte le relazioni esistenti è quasi impossibile, e facendo affidamento solo alla memoria si rischia di commettere degli errori. Per questo possono essere molto utili i diagrammi, il cui scopo principale è quello di mostrare graficamente le relazioni presenti tra le tabelle del database. Sempre a causa delle dimensioni della base di dati e del numero di tabelle, i diagrammi che risultano più utili sono quelli che descrivono solo delle porzioni di database; infatti un diagramma che raffiguri le relazioni tra tutte le tabelle sarebbe di difficile interpretazione. I diagrammi che ho creato rappresentano dunque solo delle parti del database; in particolare mi è stato chiesto di creare quelli nei quali sono coinvolte le tabelle più importanti e maggiormente utilizzate. In ogni diagramma da me creato, quindi, è presente al centro la tabella di maggior interesse, con attorno tutte le tabelle direttamente collegate ad essa.

Per costruire i diagrammi ho dovuto cercare, per ogni tabella di interesse, tutte le tabelle ad essa collegate. Per fare ciò mi è stata molto utile una query, creata grazie anche all'aiuto del team di programmazione, che mostrasse tutte le tabelle del database nelle quali fosse utilizzato un certo campo. Infatti questa query mi ha permesso di ricostruire le relazioni nelle quali la chiave della tabella di interesse venisse utilizzata in altre tabelle come chiave esterna, altrimenti tale operazione sarebbe risultata davvero molto complicata da compiere.

Dato il largo utilizzo che ne ho fatto, riporto la query che ho utilizzato:

```
SELECT table_name, column_name
FROM information_schema.columns
WHERE data_type = 'varchar' and column_name like '%codclifor%'
   and left(table_name,2) <> 'P_'
ORDER BY table_name, column_name
```

3.3.1. Creazione dei diagrammi con Management Studio 2005

Per la creazione dei diagrammi del database ho utilizzato, come detto in precedenza, SQL Management Studio 2005. Nelle prossime righe, quindi, ne descriverò brevemente il funzionamento.
All'apertura di Management Studio appare una schermata con la quale è possibile connettersi a SQL Server, sia tramite l'autenticazione di Windows, sia autenticandosi con le credenziali di utente presente in SQL Server, come mostrato in figura 3.1.

Figura 3.1: Connessione a SQL Server

Una volta entrati, sulla sinistra vengono visualizzati tutti i database presenti, e per ognuno di essi è possibile visualizzarne i diagrammi, le tabelle e tutto ciò che riguarda la gestione del database. Espandendo il menù di una tabella compaiono una serie di altri menù tramite i quali è possibile visualizzare le colonne, le chiavi, i trigger e gli altri elementi associati alla tabella. Per creare nuove colonne, trigger o chiavi sarà sufficiente cliccare con il tasto destro del mouse sulla voce corrispondente e cliccare su Nuovo.

Se invece si espande il menù dei diagrammi è possibile visualizzare i diagrammi già creati, oppure crearne di nuovi.

Management Studio permette inoltre di connettersi a database che non si trovino nel computer sul quale si sta lavorando; infatti per connettersi ad un database è sufficiente che questo si trovi in un computer raggiungibile attraverso una rete e che si abbiano le credenziali per potervi accedere.

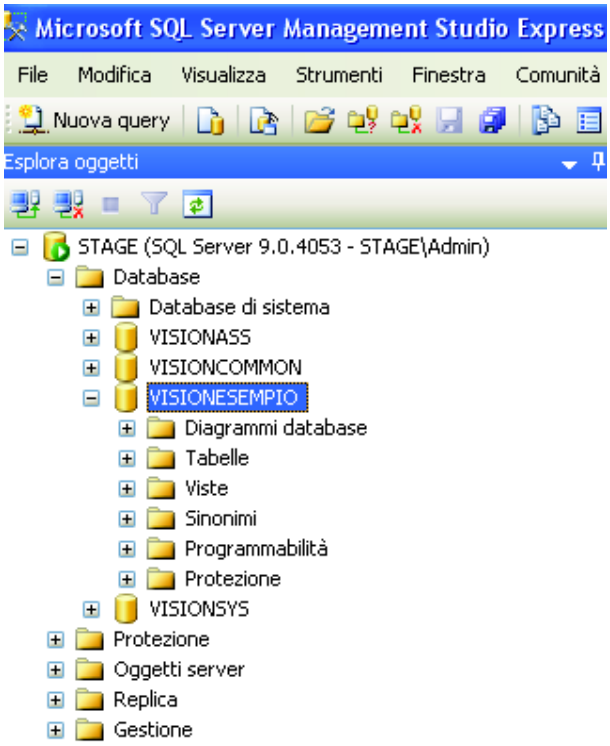

Figura 3.2: Esplora oggetti

Se si sceglie di creare un nuovo diagramma si aprirà sulla destra una nuova schermata, dalla quale si potranno aggiungere le tabelle sulle quali si vogliono creare le relazioni. Per creare una nuova relazione è sufficiente cliccare su un campo di una tabella e trascinarlo sul campo corrispondente dell'altra tabella. Apparirà così una schermata nella quale sarà possibile dare un nome alla relazione creata e impostare alcuni parametri della relazione stessa.

Le figure 3.3 e 3.4 mostrano le schermate di creazione di una relazione.

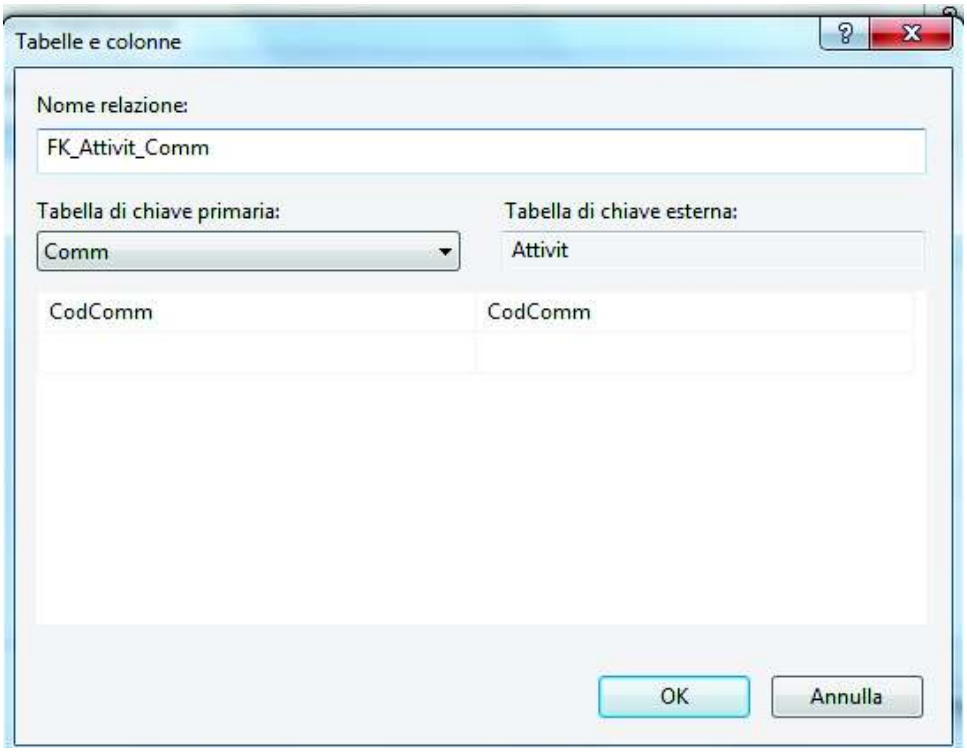

Figura 3.3: Schermata di scelta del nome della relazione e delle chiavi coinvolte

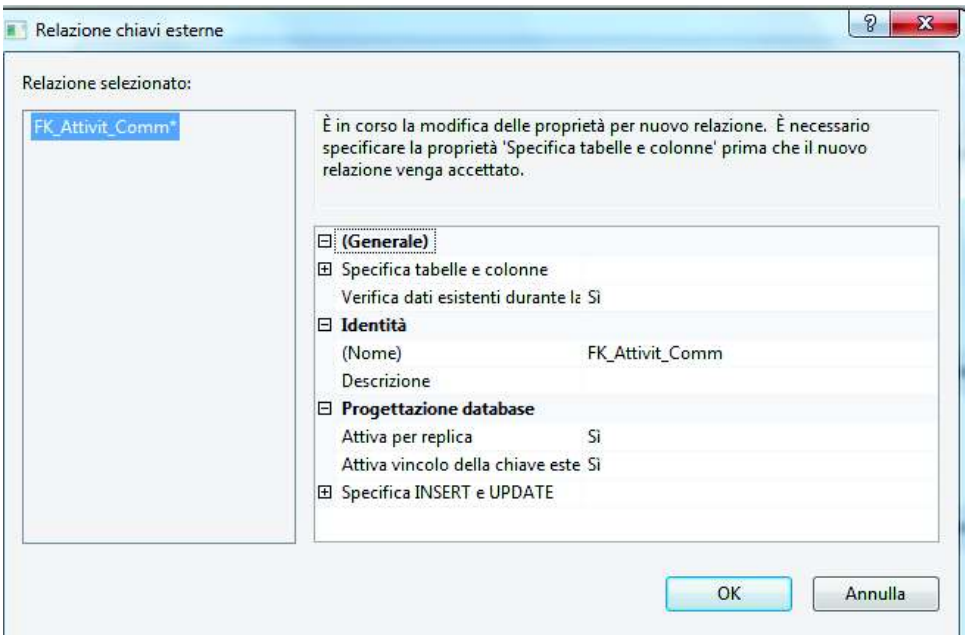

Figura 3.4: Scelta delle proprietà della relazione

Il collegamento tra due tabelle è rappresentato tramite una linea, la quale alle estremità ha dei simboli che indicano il tipo di relazione (uno a uno oppure uno a molti). Inoltre la linea può essere continua o tratteggiata a seconda che si crei una relazione nella quale il vincolo di chiave esterna sia applicato o meno. In Management Studio infatti è possibile scegliere al momento della creazione della relazione se applicare il vincolo di chiave esterna o meno, e nel caso in cui si scelga di non applicarlo la relazione avrà il solo scopo di evidenziare graficamente l'esistenza di una associazione tra le due tabelle, la quale però non deve essere gestita normalmente dal DBMS controllando se è rispettato il vincolo di integrità referenziale. Nei grafici da me creati ho utilizzato molto spesso questa tecnica perché permette di avere un riscontro grafico della relazione, la quale però può essere gestita tramite l'uso di trigger.

Naturalmente Management Studio offre un supporto alla creazione delle relazioni, ma lo stesso risultato si sarebbe potuto ottenere tramite l'esecuzione di una query con l'istruzione in linguaggio T-SQL per la creazione della relazione. Utilizzando questo applicativo, però, l'operazione risulta certamente più agevole e, cosa più importante, dà la possibilità di avere un riscontro grafico delle relazioni create. Management Studio inoltre, durante la creazione della relazione, si occupa del controllo dei campi coinvolti nella relazione, ossia controlla che questi siano dello stesso tipo e abbiano la stessa lunghezza.

La figura 3.3 mostra un diagramma da me creato con Management Studio. La tabella al centro del diagramma è quella di interesse e le tabelle attorno sono tutte quelle che si legano ad essa. Dai simboli alle estremità delle relazioni si capisce che le relazioni sono tutte del tipo uno a molti e che la parte "a molti" è su Attivit, quindi le chiavi primarie delle altre tabelle si trovano dentro alla tabella Attivit come chiavi esterne.

Sopra alle linee che esprimono le relazioni, ci sono le scritte "trigger di cancellazione" e "trigger di modifica", le quali stanno ad indicare che i trigger per il controllo dei dati sono stati creati e sono attivi sia in fase di cancellazione che in fase di aggiornamento.

RELAZIONI E VINCOLI DI INTEGRITA' REFERENZIALE CHE COINVOLGONO LA TABELLA Attivit.

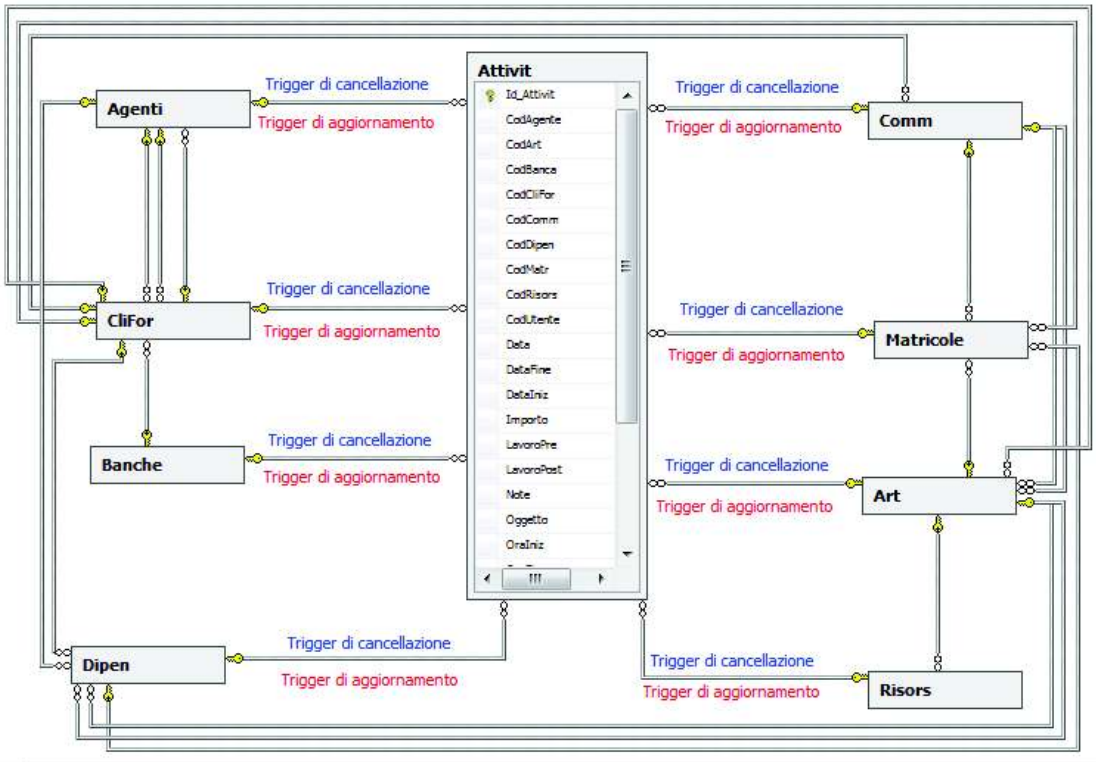

Figura 3.5: Diagramma della tabella Attivit

4. I Trigger

4.1. Definizione di trigger e loro utilità

I trigger sono delle procedure memorizzate nella base di dati ed associate alle tabelle, che vengono attivate dal DBMS quando si fanno determinate operazioni sulle tabelle stesse. Un trigger, quindi, specifica un'azione che deve essere attivata automaticamente nel caso in cui si verifichi un determinato evento su una tabella. Gli eventi che ne possono provocare l'attivazione sono le operazioni di inserimento, aggiornamento e cancellazione. Il corpo del trigger consiste in una serie di istruzioni in linguaggio T-SQL, le quali definiscono le azioni da svolgere nel caso in cui il trigger venga attivato.

A seconda del loro uso, possiamo dividere i trigger in passivi ed attivi. Un trigger è passivo se serve a provocare un fallimento della transazione corrente sotto certe condizioni, mentre un trigger è attivo quando, in corrispondenza di certi eventi, modifica lo stato della base di dati.

I trigger, attivi e passivi, sono utili per diversi scopi. Ad esempio possono servire per definire vincoli di integrità referenziale, oppure per fare dei controlli sulle operazioni ammissibili degli utenti, facendo in modo, ad esempio, che certi utenti possano inserire solo alcuni dati e non altri.

Poiché i trigger sono memorizzati nella base di dati e la loro attivazione è gestita dal DBMS, essi hanno i seguenti vantaggi:

- si semplifica la codifica delle applicazioni, che non devono preoccuparsi dei controlli effettuati dai trigger
- gli utenti che utilizzano la base di dati tramite un'applicazione non possono evitare i controlli garantiti dai trigger

Nella mia esperienza di stage ho utilizzato tutte e due le tipologie di trigger e lo scopo del loro utilizzo è stato quello di definire dei vincoli di integrità tra le tuple

di diverse tabelle. In particolare, nella fase di creazione dei trigger di cancellazione, ho inserito dei comandi che facessero in modo che quando venisse cancellata una riga di una tabella, il DBMS controllasse se la chiave primaria della riga cancellata fosse utilizzata come chiave esterna nelle tabelle collegate. In caso affermativo ho dovuto decidere se far apparire un messaggio che impedisse la cancellazione oppure cancellare a cascata le righe presenti sulle tabelle collegate.

Ci si potrebbe dunque chiedere perché utilizzare i trigger per definire i vincoli di integrità, anziché utilizzare i vincoli di chiave esterna. Nel mio caso la motivazione per cui si è deciso di utilizzare i trigger piuttosto che i vincoli di chiave esterna è stata la necessità di sapere in quali tabelle fosse realmente utilizzata la chiave che si cercava di cancellare. L'applicazione di un vincolo di foreign key quindi non soddisfava le esigenze funzionali dell'applicazione. Per capire meglio il problema utilizziamo un esempio. Ipotizziamo di voler cancellare un agente dalla tabella Agenti e ipotizziamo inoltre che questa sia collegata con le tabelle Clienti, che contiene l'anagrafica clienti dell'azienda, e Obiettivi, che contiene gli obiettivi che si è posto ogni agente. Se si decide di cancellare un agente, bisogna considerare che questo potrebbe essere utilizzato nella tabella Clienti, oppure in Obiettivi, o anche in entrambe. Se in fase di creazione delle tabelle si era applicato un vincolo di chiave esterna, allora provando a cancellare l'agente comparirebbe un messaggio nel quale il DBMS ci avvertirebbe che la cancellazione non è possibile perché altrimenti si violerebbe il vincolo di chiave esterna. Non sarebbe quindi possibile capire in quale delle tabelle collegate sia utilizzato l'agente che si vuole eliminare, né in quante righe. Utilizzando un trigger, invece, è possibile far apparire un messaggio più completo che spieghi dove è utilizzato l'agente e anche quante volte. Questo risulta fondamentale per un software gestionale come VisionSQL perché facilita di molto il compito di chi deve inserire e cancellare dati.

4.2. Tipi di trigger

Esistono principalmente due tipi di trigger:

- DDL: si attivano quando si cerca di fare una modifica strutturale ad un oggetto del database (DDL è infatti acronimo di Data Definition Language);
- DML: vengono scatenati quando si agisce sui dati (DML è acronimo di Data Manipulation Language).

Durante lo stage io ho lavorato solamente con i trigger DML, cioè quelli che si scatenano tramite operazioni di insert, update e delete. Ciò è comprensibile dato che chi utilizza un software gestionale che si interfaccia con una base di dati non ha comandi che gli consentano di modificare la struttura del database, quindi è inutile definire dei trigger a tale scopo.

I trigger DML possono essere di due tipi, a seconda che vengano eseguiti al posto dell'operazione che li ha scatenati (*instead of*) oppure dopo aver effettuato l'operazione (*after*). Nel caso di trigger di tipo *instead of* , quindi, l'operazione (insert, update o delete) che ha fatto scatenare il trigger non viene eseguita così com'è stata lanciata, ma al suo posto vengono eseguite le operazioni contenute nel trigger. In un trigger di tipo *after*, invece, i comandi definiti all'interno del trigger vengono eseguiti dopo aver compiuto l'operazione che scatena il trigger stesso.

Bisogna quindi fare attenzione alla scelta del tipo di trigger da creare perché, a seconda del tipo, può cambiare la logica delle operazioni da eseguire. Se, ad esempio, si vuole creare un trigger che impedisca la cancellazione di una riga di una tabella se la sua chiave è utilizzata in un'altra tabella, le operazioni da eseguire sono diverse a seconda che si tratti di un trigger "instead of" o "after". Infatti, con un trigger di tipo after, se durante il controllo nelle tabelle collegate si scopre che la chiave è utilizzata e quindi si decide che non deve essere cancellata, sarà necessario eseguire un'operazione di rollback, ossia l'annullamento della cancellazione, perché con un trigger di questo tipo il controllo verrà effettuato a operazione già avvenuta, e cioè quando la tupla è

già stata eliminata. Utilizzando un trigger instead of, al contrario, non sarà necessario eseguire il rollback, ma se invece si dovesse decidere di eliminare la chiave perché questa non è utilizzata in nessuna tabella collegata, bisognerà ricordare di inserire l'istruzione per farlo.

4.3. Definizione di cursori T-SQL e loro utilizzo nei trigger

Un cursore è un oggetto, definito su una generica interrogazione, che permette di accedere alle righe di una tabella una per volta.
I cursori Transact-SQL, quindi, possono essere utilizzati all'interno dei trigger per rendere disponibile il contenuto di un set di risultati ad altre istruzioni Transact-SQL. Per meglio intendere questa affermazione è necessario spiegare quali sono le operazioni che SQL Server compie durante la cancellazione o l'aggiornamento di uno o più record di una tabella.
In fase di cancellazione dei dati da una tabella, le righe cancellate vengono copiate in una tabella speciale, chiamata "deleted". Un trigger che abbia lo scopo di controllare se la chiave che si vuole cancellare sia utilizzata nelle tabelle collegate necessiterà di avere accesso alle righe cancellate, e quindi dovrà accedere alla tabella "deleted". Sarà necessario però accedere alle righe di questa tabella "una alla volta", in modo che il controllo sulle tabelle collegate avvenga per ogni codice che si sta cercando di cancellare. A questo scopo può quindi essere utile un cursore, il quale serve a rendere disponibili una alla volta le righe della tabella "deleted".

Il comando utilizzato per la creazione di un cursore è il seguente:

```
---------------------------------------------------------------------------------------------------

DECLARE nomeCursore CURSOR
FOR SELECT nomeCampo1, nomeCampo2, …, nomeCampoN
FROM nomeTabella

---------------------------------------------------------------------------------------------------
```

Nel nostro caso nomeTabella sarà proprio deleted, cioè la tabella che contiene le righe che si vogliono cancellare.
Dopo aver dichiarato un cursore è necessario aprirlo. Con l'apertura del cursore questo verrà riempito con le tuple prese dalla tabella alla quale è ancorato.
L'istruzione è molto semplice:

```
OPEN nomeCursore
```

A questo punto per scorrere ogni singola riga della tabella cui il cursore è associato e renderla disponibile ad altre istruzioni T-SQL è necessario utilizzare il comando FETCH nel modo seguente:

```
FETCH NEXT FROM nomeCursore INTO @nomeVariabile1 ,…, @nomeVariabileN
```

Si noti che le variabili nelle quali salvare il contenuto del cursore devono essere dichiarate all'inizio del trigger e devono essere dello stesso tipo e almeno della stessa lunghezza del dato che devono contenere.

Quando la lettura dei dati è completata è possibile procedere alla chiusura del cursore:

```
CLOSE nomeCursore
```

Questa istruzione permette di liberare alcune risorse che il cursore stava utilizzando, ma in questo modo il cursore potrà essere riaperto in seguito se lo si vorrà. Per liberare tutte le risorse che il cursore utilizza è necessario utilizzare il comando

---

```
DEALLOCATE nomeCursore
```

In questo modo il cursore viene completamente rimosso.

Per riepilogare, quindi, la procedura normalmente adottata per utilizzare un cursore Transact-SQL in un trigger è la seguente:

- Dichiarare all'inizio del trigger le variabili in cui includere i dati restituiti dal cursore.
- Associare un cursore a un'istruzione SELECT mediante l'istruzione DECLARE CURSOR.
- Utilizzare l'istruzione OPEN per eseguire l'istruzione SELECT e popolare il cursore.
- Utilizzare l'istruzione FETCH INTO per recuperare singole righe e trasferire i dati di ogni colonna nella variabile specificata.
- Eseguire l'istruzione CLOSE e, se lo si desidera, DEALLOCATE.

4.4. Mappatura dei trigger del database di VisionSQL

Come già anticipato in precedenza, il database di VisionSQL disponeva già di alcuni trigger. Dopo aver proceduto alla creazione dei diagrammi e aver quindi capito come fosse strutturato il database, ho potuto iniziare la fase di mappatura dei trigger. Questa fase è consistita nel controllare, per ogni relazione creata in precedenza, se fosse presente o meno un trigger per il controllo dei dati, inizialmente per la cancellazione e successivamente per l'aggiornamento. Per portare a termine questo lavoro ho costruito un documento di Excel, nel quale ho inserito, per ogni diagramma creato, tutte le tabelle che si collegavano alla tabella principale, scrivendo accanto se il trigger era già presente o meno.
Alla fine di questo processo ho proceduto alla creazione dei trigger mancanti, dove questi fossero necessari, aggiornando il file di Excel, il quale alla fine dell'opera è stato consegnato al team di sviluppo come documentazione ufficiale dei trigger presenti nel database.

Riporto di seguito una pagina del file Excel che riassume e mappa i trigger presenti nel diagramma Allegati. In ogni riga sono presenti rispettivamente: (I) il nome della tabella collegata, (II) lo stato del trigger, (III) su quale trigger deve essere inserito il controllo, (IV) il numero di relazioni. In basso c'è una legenda che spiega quali possono essere gli stati del trigger.

	Trigger		Trigger di cancellazione relativo	Trigger di aggiornamento relativo	Num rel
Nome Tabella	**Cancellazione**	**Aggiornamento**			
Doc	ok	aggiunto	delDoc	updDoc	1
CliFor	aggiunto	ok	delCliFor	updCliFor	1
Agenti	aggiunto	ok	delAgenti	updAgenti	1
Lotti	aggiunto	aggiunto	delLotti	updLotti	1
Matricole	aggiunto	creato	delMatricole	updMatricole	1
Art	aggiunto	aggiunto	delArt	updArt	1
Banche	aggiunto	ok	delBanche	updBanche	1
Cont	ok	aggiunto	delCont	updCont	1
PnTes	aggiunto	no (*)	delPnTes	updPnTes	1
CauCon	aggiunto	aggiunto	delCauCon	updCauCon	1
Comm	aggiunto	aggiunto	delComm	updComm	1
DocTes	ok	no (*)	delDocTes	updDocTes	1
Risors	aggiunto	aggiunto	delRisors	updRisors	1
Dipen	aggiunto	aggiunto	delDipen	updDipen	1

(*) non va il controllo perché la chiave primaria è un ID e l'utente non la può aggiornare

LEGENDA

ok *= il trigger era già presente*
ok FK *= il controllo sulla tabella non serve perché agisce il vincolo di chiave esterna*
aggiunto *= il trigger esisteva ma è stato aggiunto il controllo sulla tabella*
creato *= il trigger non esisteva ed è stato creato*

4.5. Creazione dei trigger con Management Studio 2005

La creazione di un trigger tramite Management Studio 2005 può essere fatta semplicemente cliccando su “nuovo trigger” dalla sezione esplora oggetti. Se invece lo si vuole creare da zero, lo si può fare tramite l’esecuzione di una serie di istruzioni in linguaggio T-SQL. In particolare, è necessario specificare le seguenti informazioni:

- Il nome del trigger
- La tabella alla quale ancorarlo
- Quando deve essere attivato
- Le istruzioni che deve eseguire una volta attivato

Il comando per la creazione di un trigger in Management Studio è il seguente:

```
-------------------------------------------------------------------------------------------------------
CREATE TRIGGER [nomeDatabase].[ nomeTrigger]
  ON [nomeDatabase].[nomeTabella]
  {INSTEAD OF|AFTER}  {INSERT|UPDATE|DELETE}
AS
  BEGIN
        [istruzioni in linguaggio T-SQL]
  END
-------------------------------------------------------------------------------------------------------
```

CREATE TRIGGER è l’istruzione che crea fisicamente il trigger mentre ON indica a quale tabella il trigger deve essere ancorato. Poi devono essere specificati il tipo (AFTER o INSTEAD OF) e quando il trigger deve essere attivato (INSERT, UPDATE, DELETE). Le istruzioni comprese tra i comandi BEGIN e END rappresentano invece il corpo del trigger.

Per capire meglio i comandi utilizzati nei trigger, inserisco uno dei trigger da me creati, con la spiegazione di ciò che significano i comandi inseriti.

```
-- =============================================
-- Author:                Pegoraro Marco
-- Create date:  27/05/2010
-- Description:   Trigger Cancellazione tabella GrpArt
-- =============================================
ALTER TRIGGER [dbo].[TrigDelGrpArt]
   ON  [dbo].[GrpArt]
   INSTEAD OF DELETE
AS
BEGIN
   DECLARE @DeleteCount int;
   DECLARE @NFound int;
   DECLARE @i int;
   DECLARE @Codice VARCHAR(10);
   DECLARE @Message VARCHAR(500);


   SELECT @DeleteCount = COUNT(*) FROM deleted;


   DECLARE crDeleted CURSOR
   FOR SELECT CodGrpArt
   FROM deleted;

   OPEN crDeleted;

SET @i = 1;
   FETCH NEXT FROM crDeleted INTO @Codice;
   WHILE @i <= @DeleteCount
   BEGIN
         SET @Message = ''

         -- Controllo della tabella ART
         SELECT @NFound = COUNT(*) FROM Art WHERE CodGrpArt = @Codice;
         IF @NFound > 0
         BEGIN
               SET @Message = @Message + N'Anagrafica Articoli (' + STR(@NFound) + ')'
               + CHAR(13);
         END;

         -- Controllo della tabella DOCRIG
         SELECT @NFound = COUNT(*) FROM DocRig WHERE CodGrpArt = @Codice;
         IF @NFound > 0
         BEGIN
               SET @Message = @Message + N'Righe Documenti (' +
               LTRIM(STR(@NFound)) + ')' + CHAR(13);
         END;
```

Creazione del trigger

Dichiarazione delle variabili

Conteggio del numero di record cancellati

Creazione cursore per scorrere le righe della tabella deleted e sua apertura

Trasferimento del primo codice cancellato dal cursore alla variabile @Codice e inizio del ciclo per il controllo o la cancellazione nelle tabelle collegate

Messaggio per impedire la cancellazione se il codice è utilizzato in ART

Messaggio per impedire la cancellazione se il codice è utilizzato in DOCRIG

```
        -- Elimino il record o stampo il messaggio di avviso
        IF LEN(@Message) = 0
        BEGIN                                   Cancellazione dei record nelle tabelle associate
                DELETE FROM Listini WHERE CodGrpArt = @Codice


                -- Elimino il record
                DELETE FROM GrpArt WHERE CodGrpArt = @Codice;
        END
        ELSE
        BEGIN
                -- Imposto la descrizione completa per il messaggio da visualizzare
                SET @Message = 'Il codice ' + @Codice + ' risulta usato negli archivi:' +
                CHAR(13) + @Message
                PRINT @Message                  Creazione e visualizzazione del messaggio
        END

        SET @i = @i + 1
        FETCH NEXT FROM crDeleted INTO @Codice

        -- Restituisco il messaggio costruito nei controlli
        SELECT @Message AS Messaggio
  END
  CLOSE crDeleted                               Chiusura del cursore
END
```

TEST E ANALISI DELLE PROCEDURE REALIZZATE

5. Misurazioni dei tempi di risposta del database

5.1. Introduzione

Il lavoro svolto durante la seconda parte dello stage è stato incentrato sull'attività di test dei trigger creati. In particolare mi è stato chiesto di condurre delle analisi per controllare se il tempo di risposta del database dopo l'introduzione dei trigger di cancellazione fosse aumentato. Non c'è stata quindi una fase di test per i trigger di aggiornamento, in quanto l'applicativo VisionSQL non permette ancora di aggiornare le chiavi primarie delle tabelle. Tale funzionalità sarà implementata a breve anche grazie al lavoro di creazione dei trigger di aggiornamento da me svolto.

Il lavoro che ho svolto in questa fase, quindi, è stato quello di provare a cancellare dei record direttamente dall'applicativo VisionSQL e misurare i tempi di cancellazione in presenza ed in assenza dei trigger.

Per meglio capire come siano state effettuate la misurazioni dei tempi di risposta è necessario spiegare che l'applicativo VisionSQL è stato sviluppato in linguaggio Microsoft Visual FoxPro. È bastato quindi, con l'aiuto del team di programmazione, creare delle istruzioni che facessero generare un file di log contenente i tempi desiderati. In realtà il codice creato non ritornava direttamente i tempi di cancellazione, ma il millisecondo in cui era iniziata l'operazione di cancellazione e il millisecondo in cui era terminata. Oltre alla creazione del codice, è stato necessario capire dove inserirlo, ossia capire da dove iniziare a misurare i tempi e fino a dove misurarli. Il problema principale che ho dovuto affrontare è derivato dal fatto che, quando si decide di cancellare un record, in VisionSQL appare un messaggio dove si richiede una conferma di cancellazione. Tale operazione richiede un input da parte dell'utilizzatore e quindi le misurazioni non dovevano tenere conto di questo

tempo. Il codice del programma però era molto complesso, con varie chiamate a funzioni e procedure esterne, quindi non è stato semplice capire dove inserire i nuovi comandi.

Dopo aver completato l'inserimento del codice ho potuto eseguire i test che mi erano stati richiesti e per farli mi è stato consentito di lavorare dapprima su un database di medie dimensioni, poi su un database di grosse dimensioni di un'azienda cliente.

5.2. Descrizione dei test di cancellazione

I test effettuati sono stati necessari per verificare se la differenza tra i tempi di cancellazione in presenza ed in assenza dei trigger fosse statisticamente significativa, e in tal caso per trovare una stima dei nuovi tempi di risposta del database per capire se risultassero ancora accettabili.

I test sono stati fatti sulle tabelle aventi il maggior numero di relazioni con altre tabelle, così che fossero maggiori sia i controlli da effettuare sulle tabelle collegate che le cancellazioni a cascata. Questo perché quando il trigger impone controlli su tabelle piccole, il maggior tempo di esecuzione è certamente trascurabile.

Lo scopo principale dei test è stato quindi quello di capire se quando si effettua la cancellazione di una riga che non sia collegata a nessun'altra riga di altre tabelle, il processo di cancellazione risultasse più lento di quando i trigger non erano implementati, e in caso affermativo avere una misura di quali fossero i nuovi tempi di cancellazione. Infatti controllare se la cancellazione di una riga impiega più tempo quando questa è collegata ad altre righe di altre tabelle non è di interesse perché è un'operazione che è importante eseguire per mantenere attivo il vincolo di integrità dei dati.

Le operazioni di test, quindi, sono state eseguite semplicemente inserendo un record in una tabella e poi cancellandolo. Infatti, così facendo, i trigger si attivano e fanno aumentare il tempo di esecuzione della query di cancellazione.

I test sono stati condotti analizzando i tempi di cancellazione nelle tabelle CliFor, Art e Doc. Per ognuna di queste tabelle sono state effettuate trenta misurazioni dei tempi di cancellazione in presenza dei trigger e trenta in assenza di trigger. Il test è stato ripetuto su due diversi database, SIVA e BIOSLINE, il primo di media grandezza e il secondo abbastanza corposo, per verificare che anche in grossi database i tempi rimanessero accettabili.

Il calcolatore utilizzato per eseguire i test ha le seguenti caratteristiche:

Intel Pentium 4 1.80GHz 1GB di RAM

È importante puntualizzare che test effettuati su calcolatori diversi porterebbero a stime dei tempi di cancellazione diverse. Ciò è naturale dato che i tempi di esecuzione delle query di cancellazione sono fortemente legati alla potenza di calcolo del computer nel quale vengono eseguite. Naturalmente però, al di la delle stime prodotte, i risultati delle analisi rimangono le stesse indipendentemente dal calcolatore che si utilizza.

5.3. Verifica degli assunti statistici

Ciò che mi è stato chiesto di fare durante la fase di test può essere visto, dal punto di vista statistico, come un test per la verifica dell'uguaglianza tra le medie dei tempi di risposta in due popolazioni, una nella quale sono presenti i trigger, e una dove non sono presenti. Per verificare questa ipotesi, inizialmente, avevo intenzione di utilizzare un test "t di Student a due campioni", ma durante l'analisi mi sono accorto della mancanza di un requisito fondamentale per l'applicazione di tale test, quindi ho utilizzato un test "t di Welch". Oltre al test, ho costruito degli intervalli di confidenza per la stima della media dei tempi nelle due popolazioni. Queste operazioni sono state fatte sia nel database SIVA, che nel database BIOSLINE.

Il test "t di Student" per la verifica di uguaglianza di due medie, essendo un test parametrico, richiede che siano rispettati alcuni assunti statistici.

Questi sono:

- Normalità delle popolazioni
- Indipendenza tra le osservazioni del campione (i.i.d.)
- Indipendenza tra le osservazioni dei due campioni
- Le varianze delle popolazioni devono essere note, o se non lo sono devono poter essere considerate uguali; in ogni caso sia media che varianza devono essere finite

Il primo assunto riguarda la normalità delle osservazioni. Nei casi in cui questa non possa essere assunta, vengono in aiuto i risultati del Teorema del Limite Centrale. Questo teorema ci rassicura sul fatto che qualunque sia la forma della distribuzione di n variabili casuali i.i.d., la loro somma tende asintoticamente ad una normale. Studi empirici hanno dimostrato che l'approssimazione risulta sufficientemente buona per n > 30 se la distribuzione di partenza è almeno simmetrica, con n > 50 invece l'approssimazione risulta buona qualunque sia la distribuzione di partenza.
Nel test da me effettuato ho deciso di utilizzare una dimensione campionaria di trenta unità per ipotizzare la distribuzione normale nelle popolazioni.

Per quanto riguarda l'indipendenza delle osservazioni, questa potrebbe non essere garantita a causa del meccanismo di "caching" che SQL Server utilizza. Questo meccanismo consiste nel memorizzare in una memoria temporanea delle informazioni riguardanti una query eseguita, per poi utilizzarle se viene rieseguita una seconda volta la stessa query. Per ovviare a questo problema è bastato eseguire, ad ogni cancellazione effettuata, due comandi che cancellano il contenuto della cache:

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE
```

In questo modo l'indipendenza delle osservazioni è garantita.

Infine, per quanto riguarda la variabilità, questa è sicuramente finita proprio per la natura dell'esperimento. Il problema che ho invece dovuto affrontare è stato quello della non omoschedasticità delle osservazioni, requisito fondamentale

per l'applicazione del test "t di Student". Proprio per questo motivo sono stato costretto ad utilizzare una "statistica alternativa" chiamata "t di Welch", la quale può essere utilizzata quando non sia rispettata l'uguaglianza delle varianze nelle popolazioni.

Gli assunti richiesti per poter eseguire la verifica d'ipotesi, quindi, sono stati tutti rispettati, almeno per poter trarre delle conclusioni asintotiche.

Per documentare il lavoro svolto ho utilizzato due documenti Excel, uno per ognuno dei due database utilizzati per i test, nei quali ho riportato tutte le rilevazioni e i calcoli effettuati. Questo materiale, al termine dei lavori, è stato consegnato al tutor aziendale, il quale ha provveduto ad inserirlo tra la documentazione ufficiale del software VisionSQL.

Per la visione dei calcoli effettuati rimando all'appendice, nella quale sono specificati tutti i passaggi effettuati.

5.4. Risultati delle analisi

I risultati ottenuti dalle analisi condotte erano prevedibili, ma sono comunque stati davvero molto utili per avere delle misure di riferimento dei tempi di cancellazione. Dalle analisi è infatti emerso che i trigger provocano in generale un aumento del tempo medio di cancellazione. In particolare si può verificare che il tempo aumenta all'aumentare del numero di righe presenti nelle tabelle in cui devono essere effettuati i controlli. In database molto grossi, quindi, il peso dei trigger può farsi sentire pesantemente, ma d'altra parte i trigger sono indispensabili per garantire l'integrità dei dati, quindi hanno una grande utilità.
Come anticipato in precedenza, l'analisi è stata condotta misurando, per ognuno dei due database, i tempi di cancellazione nelle tre tabelle CliFor, Art e Doc, sia in presenza che in assenza di trigger. La scelta di tali tabelle mi è stata suggerita dal tutor aziendale, il quale mi ha fatto notare che sono proprio le tabelle in cui i trigger fanno eseguire il maggior numero di controlli sulle tabelle

collegate. Avendo effettuato le rilevazioni su queste, quindi, sono stati rilevati i tempi di cancellazione più alti tra tutti i possibili tempi.
Come detto in precedenza, le stime dei tempi possono valere per calcolatori molto simili a quello nel quale sono state condotte le analisi, ma i risultati generali rimangono comunque validi per ogni calcolatore.
Il risultato dei test nel database SIVA ha mostrato che i tempi più lunghi si registrano in fase di cancellazione di un record dalla tabella CliFor, che corrisponde all'anagrafica dei clienti e dei fornitori. Per avere una stima del tempo di cancellazione ho costruito un intervallo di confidenza al livello del 95%, il quale è risultato essere:

IC per CliFor con trigger in secondi: [4,083412965 - 4,278320369]

Mediamente, quindi, l'utilizzatore del programma dovrà attendere tra i 4,1 e i 4,3 secondi perché venga terminata la cancellazione di un Cliente o di un Fornitore.
Dato che però CliFor è la tabella che registra i tempi più lunghi, il tempo di cancellazione nelle altre tabelle è mediamente più basso; inoltre bisogna tenere conto del fatto che la cancellazione di un Cliente o di un Fornitore non è un'operazione che venga eseguita molto frequentemente ma è invece importante controllare che non venga cancellato un cliente che sia utilizzato in righe di altre tabelle.

In assenza di trigger l'intervallo di confidenza al 95% per il tempo medio è invece risultato:

IC per CliFor senza trigger in secondi: [0,168733381 - 0,233799953]

Si nota che i tempi sono quindi molto differenti, e l'introduzione dei trigger li fa lievitare di molto.
Questi risultati valgono però in un database di medie dimensioni, come quello di SIVA. L'idea che è venuta in seguito è stata quella di provare a misurare i tempi sul database più grosso tra quelli delle aziende clienti (il database di

BIOSLINE), in modo da avere anche una misura di quali fossero i tempi per basi di dati di grandi dimensioni.
Le analisi condotte sul database BIOSLINE hanno dimostrato che anche in questo caso i tempi di risposta maggiori si registrano in fase di cancellazione di un Cliente/Fornitore. L'IC al livello 95% per il tempo medio di risposta in presenza dei trigger su CliFor è risultato:

IC per CliFor con trigger in secondi: [215,4952435 - 231,8178232]

Come si può vedere, i tempi in questo database sono risultati particolarmente alti; l'intervallo di confidenza espresso in minuti sarebbe infatti:

IC per CliFor con trigger in minuti: [3,59 – 3,86]

Questo tempo sembra essere inaccettabile perché molto alto. Al riguardo, è necessario ricordare che le misurazioni che ho effettuato sono state fatte su un PC Intel Pentium 4, CPU 1.80GHz con 1GB di RAM. Tali caratteristiche sono evidentemente insufficienti per gestire un database di grosse dimensioni come quello di BIOSLINE; infatti con un calcolatore più potente i tempi diminuirebbero di molto. La stessa azienda BIOSLINE utilizza infatti un server dedicato alla gestione del database, dotato di otto processori e 16GB di RAM; così facendo i tempi di cancellazione risultano molto minori. Malgrado ciò, durante l'ultimo periodo di stage ho provato a pensare ad una possibile soluzione per ridurre il tempo di cancellazione. È risultato subito evidente che la causa di un così alto tempo di risposta del database era dovuta a un controllo sulla tabella DocRig imposto dal trigger di cancellazione di CliFor. Tale tabella contiene infatti tutte le righe dei documenti di contabilità e conta, nel database BIOSLINE, quasi quattro milioni di record. Ciò che mi ha spinto a cercare un metodo per ridurre tale tempo è stato l'aver notato che anche in altri trigger di altre tabelle era presente il controllo su DocRig, ma i tempi risultavano nettamente minori. Ho cercato quindi di capire il motivo di questa differenza e sono arrivato alla conclusione che la causa fosse la presenza di alcuni indici creati appositamente

per ridurre i tempi di risposta del database. Dopo essermi documentato riguardo l'argomento "indici" ho quindi provato ad esporre al tutor aziendale la mia idea, consistente appunto nel creare un nuovo indice che velocizzasse la cancellazione dalla tabella CliFor, pur sapendo che gli indici migliorano le prestazioni in fase di ricerca ma le peggiorano in fase di scrittura. Purtroppo questa idea non risultava semplice da applicare perché il database disponeva già di moltissimi indici e non se ne volevano inserire di nuovi. L'idea da me lanciata ha però portato all'apertura di una discussione tra il team di programmazione e il tutor aziendale sulla possibilità di aggirare il controllo sulle righe inserendo un vincolo direttamente sul software gestionale VisionSQL. L'introduzione di questo vincolo, purtroppo, non poteva essere fatta da me perché richiedeva la conoscenza del linguaggio di programmazione del software gestionale (Microsoft Visual Fox Pro), quindi non ho potuto assistere alla modifica del software, la quale verrà fatta a breve dal team di programmazione. Malgrado ciò ho avuto comunque la possibilità di misurare i tempi che si sarebbero ottenuti dopo la modifica del software gestionale, in quanto è bastato eliminare dal trigger il controllo sulla tabella DocRig e rimisurare i tempi di cancellazione. Le misurazioni effettuate hanno portato alla seguente stima del tempo di cancellazione di un record dalla tabella CliFor:

NUOVO IC per CliFor con trigger in secondi: [18,65466688 - 19,41139978]

Così facendo si è quindi riusciti ad abbattere il tempo di cancellazione di un record dalla tabella CliFor del database BIOSLINE, portandolo da tre minuti e mezzo circa a diciannove secondi circa.

In definitiva, come ci si poteva attendere, il tempo di risposta medio nei database con i trigger risulta significativamente più alto, in particolar modo quando questi impongono controlli su tabelle con molti record. D'altra parte i trigger sono indispensabili per garantire l'integrità dei dati, quindi devono essere implementati. Si può quindi concludere l'analisi affermando che i trigger devono essere implementati ma è necessario prestare particolare attenzione a non inserire controlli laddove non siano necessari, perché il carico computazionale

che essi comportano aumenta all'aumentare del numero di righe che devono essere controllate.

6. Appendice

6.1. Formule statistiche utilizzate

Riporto di seguito le definizioni di media, varianza e intervallo di confidenza, per poi passare all'analisi dei test "t di Student" e "t di Welch".

Media aritmetica:

È un indicatore di posizione. La sua formula di calcolo è:

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

Varianza:

Si definisce varianza la media dei quadrati degli scarti delle osservazioni dalla loro media. È in indicatore della variabilità dei dati: La sua formula di calcolo è:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

In genere quando si calcola la varianza di un campione viene però utilizzata la varianza campionaria corretta. Si definisce corretta proprio perché si calcola moltiplicando la varianza per il fattore di correzione $\frac{n}{n-1}$. La sua formula di calcolo è quindi:

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

Tale correzione dipende dal fatto che, per piccoli campioni, la varianza campionaria è uno stimatore distorto della varianza della popolazione.

Intervallo di confidenza per la media di una popolazione:

L'intervallo di confidenza di livello (1 – α) per la media di una popolazioneè un intervallo di valori all'interno del quale ricade con un'alta probabilità il vero valore della media della popolazione. Il suo calcolo è diverso a seconda che la varianza della popolazione sia nota o ignota. Dato che nei test la varianza non è nota e deve essere stimata, ne riporto la formula per la varianza ignota:

$$IC = \left[\bar{y} - t_{n-1,\alpha/2} \sqrt{\frac{\hat{\sigma}^2}{n}} \;\; , \;\; \bar{y} + t_{n-1,\alpha/2} \sqrt{\frac{\hat{\sigma}^2}{n}} \right]$$

I test t di Student e t di Welch per il confronto delle medie di due popolazioni:

Il test t di Student è utilizzato per la verifica dell'uguaglianza tra i valori medi di due campioni. Si tratta di un test parametrico e quindi in quanto tale si può applicare se sono soddisfatti alcuni assunti. Questi sono:

- Indipendenza delle osservazioni nei campioni e tra i campioni
- Normalità delle due popolazioni
- Varianze ignote ma uguali

L'ipotesi che si testa è quindi la seguente:

$H_0 : \mu_1 - \mu_2 = 0$
$H_1 : \mu_1 - \mu_2 \neq 0$

Naturalmente l'ipotesi alternativa può essere anche unilaterale, quindi può essere

$H_0 : \mu_1 - \mu_2 = 0$ $\qquad$ $H_0 : \mu_1 - \mu_2 = 0$
$H_1 : \mu_1 - \mu_2 < 0$ $\quad$ oppure $\quad$ $H_1 : \mu_1 - \mu_2 > 0.$

Se gli assunti sono rispettati si può procedere a calcolare il valore osservato della statistica test

$$t = \frac{(\overline{y_1} - \overline{y_2}) - (\mu_1 - \mu_2)}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Che si distribuisce come una t con $n_1 + n_2 - 2$ gradi di libertà.

Per capire se l'ipotesi nulla va accettata o rifiutata è necessario osservare il tipo di ipotesi alternativa formulata. La seguente tabella spiega quali sono le regioni di rifiuto:

Ipotesi statistica	**Regione critica di ampiezza α**
$H_0 : \mu_1 - \mu_2 = 0$ $H_1 : \mu_1 - \mu_2 > 0$	Rifiuto per $t^{oss} \geq t_{1-\alpha,g}$
$H_0 : \mu_1 - \mu_2 = 0$ $H_1 : \mu_1 - \mu_2 < 0$	Rifiuto per $t^{oss} \leq -t_{1-\alpha,g}$
$H_0 : \mu_1 - \mu_2 = 0$ $H_1 : \mu_1 - \mu_2 \neq 0$	Rifiuto per $t^{oss} \geq t_{1-\frac{\alpha}{2},g}$ oppure $t^{oss} \leq -t_{1-\frac{\alpha}{2},g}$

Il valore g è il numero di gradi di libertà, cioè $n_1 + n_2 - 2$.

È necessario ricordare che in questo caso si ha che:

$$s^2 = \frac{s_1^2(n_1 - 1) + s_2^2(n_2 - 1)}{n_1 + n_2 - 2}$$

La varianza calcolata in questo modo, viene definita varianza pooled ed è data dal rapporto tra la somma delle due varianze, ponderate secondo i rispettivi gradi di libertà, e i gradi di libertà totali.

Come detto in precedenza, una delle assunzioni alla base del test t di Student è l'omoschedasticità nelle popolazioni. Nel caso in cui le varianze non possano essere considerate uguali, una statistica che può essere utilizzata per la verifica d'ipotesi sulla differenza di due medie è la seguente:

$$t = \frac{(\overline{y_1} - \overline{y_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Questo test viene definito test di Welch e la sua distribuzione approssimata è sempre una t, che però non ha $n_1 + n_2 - 2$ gradi di libertà. Il calcolo dei gradi di libertà, infatti, viene fatto utilizzando l'equazione di Welch-Satterthwaite nel modo seguente:

$$v = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{s_1^4}{n_1^2 \cdot (n_1 - 1)} + \frac{s_2^4}{n_2^2 \cdot (n_2 - 1)}}$$

Dove v è il numero di gradi di libertà.

In questo caso, quindi, non è necessario supporre l'uguaglianza delle varianze nelle due popolazioni.

6.2. Analisi statistiche sul database SIVA

Riporto di seguito i risultati delle analisi condotte sul database SIVA.

Database SIVA CON trigger

- **Tabella CliFor**

Media = 4,180866667

Varianza corretta = 0,068113568

Intervallo di Confidenza:

[4,083412965 - 4,278320369]

- **Tabella Art**

Media = 1,148366667

Varianza corretta = 0,100253757

Intervallo di Confidenza:

[1,030135532 - 1,266597801]

- **Tabella Doc**

Media = 0,7363

Varianza corretta = 0,003304976

Intervallo di Confidenza:

[0,714833281 - 0,757766719]

Database SIVA SENZA trigger

- **Tabella CliFor**

Media = 0,201266667

Varianza corretta = 0,007590892

Intervallo di Confidenza:

[0,168733381 - 0,233799953]

- **Tabella Art**

Media = 0,9676

Varianza corretta = 0,295668662

Intervallo di Confidenza:

[0,764558798 - 1,170641202]

- **Tabella Doc**

Media = 0,0805

Varianza corretta = 0,004615362

Intervallo di Confidenza:

[0,055132106 - 0,105867894]

A prima vista si nota che nel database senza trigger i tempi di cancellazione sono sempre minori rispetto a quando ci sono i trigger. Ciò era prevedibile dato che i trigger comportano un maggior carico computazionale perché aggiungono operazioni da compiere in fase di cancellazione.

Per quanto riguarda la varianza dei dati, sembra già a prima vista impossibile ipotizzare omogeneità tra i gruppi.

A questo punto è interessante applicare un test "t" per verificare se le medie nei due gruppi possono essere considerate uguali o se i tempi sono maggiori in presenza dei trigger.

Il test corrispondente è questo:

$H_0 : \mu_1 - \mu_2 = 0$
$H_1 : \mu_1 - \mu_2 > 0$

Dove $\mu_1 e \mu_2$ sono rispettivamente il vero tempo medio quando sono presenti i trigger e il vero tempo medio quando non sono presenti.
Per decidere che statistica test utilizzare per questa verifica d'ipotesi è necessario verificare se le varianze possono essere considerate omogenee all'interno dei gruppi. Grazie al software statistico R risulta semplice eseguire un test, ad esempio il test di Bartlett per la verifica di questa ipotesi.
Il test di Bartlett, applicato ad ogni tabella in presenza ed in assenza dei trigger, ha mostrato che nelle osservazioni da me rilevate non è possibile ipotizzare l'uguaglianza delle varianze nelle due popolazioni, quindi la statistica test che utilizzerò per la verifica di ipotesi è la statistica test "t di Welch":

$$t = \frac{(\overline{y_1} - \overline{y_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Che si distribuisce come una t con ν gradi di libertà, dove:

$$\nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{s_1^4}{n_1^2 \cdot (n_1 - 1)} + \frac{s_2^4}{n_2^2 \cdot (n_2 - 1)}}$$

Riporto i valori osservati della statistica test e il numero ν di gradi di libertà :

CliFor:

$(\overline{y_1} - \overline{y_2}) = 3.9796$

ν = 35.384

t^{oss} = 79.22082 $\quad p\text{-}value \cong 0$

Art:

$(\overline{y_1} - \overline{y_2}) = 0.1807667$

ν = 46.638

t^{oss} = 1.573526 $\quad p\text{-}value \cong 0.06118$

Doc:

$(\overline{y_1} - \overline{y_2}) = 0.6558$

ν = 56.455

t^{oss} = 40.36084 $\quad p\text{-}value \cong 0$

Interpretazione dei risultati:

Nel database SIVA, sia per la tabella CliFor che per Doc si nota che la differenza tra le due medie è significativamente diversa da zero, invece per Art si potrebbe anche accettare l'ipotesi di uguaglianza delle medie. Ciò può essere spiegato dal fatto che i trigger di CliFor e Doc impongono il controllo su tabelle molto corpose, che contengono migliaia di record, quindi il carico computazionale di tali operazioni è alto; il trigger di Art invece impone controlli su tabelle con meno record e quindi comporta un costo computazionale minore.

6.3. Analisi statistiche sul database BIOSLINE

Riporto di seguito i risultati delle analisi sul database BIOSLINE.

Database BIOSLINE CON trigger

- **Tabella CliFor**

Media = 223,6565333

Varianza corretta = 217,1903506

Intervallo di Confidenza:

[215,4952435 - 231,8178232]

- **Tabella Art**

Media = 1,930366667

Varianza corretta = 0,337154999

Intervallo di Confidenza:

[1,71354817 - 2,147185163]

- **Tabella Doc**

Media = 6,051366667

Varianza corretta = 0,075440102

Intervallo di Confidenza:

[5,948805569 - 6,153927765]

Database BIOSLINE SENZA trigger

- **Tabella CliFor**

Media = 0,5783

Varianza corretta = 0,113110838

Intervallo di Confidenza:

[0,452716195 - 0,703883805]

- **Tabella Art**

Media = 1,389866667

Varianza corretta = 0,274366602

Intervallo di Confidenza:

[1,19427644 - 1,585456893]

- **Tabella Doc**

Media = 0,0339

Varianza corretta = 0,000162714

Intervallo di Confidenza:

[0,029136856 - 0,038663144]

Anche in questo caso si nota subito che i tempi di cancellazione in presenza dei trigger sono maggiori di quando i trigger sono assenti. Da notare il tempo di cancellazione per CliFor che, espresso in minuti, risulta:

IC al livello del 95% per CliFor in presenza dei trigger espresso in minuti: [3,59 – 3,86]

Il tempo risulta molto alto ma è necessario specificare che, in generale, il tempo di risposta della base di dati varia anche in funzione delle caratteristiche del

calcolatore che si sta utilizzando. Utilizzando un calcolatore più potente, quindi, i tempi risultano notevolmente più bassi.

Calcoliamo i valori osservati della statistica t di Welch per verificare se la differenza tra i tempi medi è significativamente diversa da zero, anche se già a prima vista la risposta sembra essere affermativa. Ciò sembra naturale, dato che già nel database SIVA le differenze erano significative, quindi in un database più grande ci si può aspettare che risultino ancor più significative.

Ricordiamo le formule di calcolo per la statistica t e i rispettivi gradi di libertà:

$$t = \frac{(\overline{y_1} - \overline{y_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \qquad \nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{s_1^4}{n_1^2 \cdot (n_1 - 1)} + \frac{s_2^4}{n_2^2 \cdot (n_2 - 1)}}$$

CliFor:

$(\overline{y_1} - \overline{y_2}) =$ 223.0782

ν = 14.007

t^{oss} = 58.6173 $\qquad pvalue \cong 0$

Art:

$(\overline{y_1} - \overline{y_2}) =$ 0.5405

ν = 57.395

t^{oss} = 3.785737 $\qquad pvalue \cong$ 0.0001843

Doc:

$(\overline{y_1} - \overline{y_2}) =$ 6.017467

ν = 29.125

t^{oss} = 119.8686 $\quad pvalue \cong 0$

Interpretazione dei risultati:

Nel database BIOSLINE, come previsto, tutte le differenze sono significativamente diverse da zero. Ciò in qualche modo conferma la correttezza delle analisi fatte nel database SIVA; infatti all'aumentare della dimensione del database non può che aumentare anche il carico computazionale dovuto all'esecuzione dei trigger.

6.4. Nuove analisi per la tabella CliFor del database BIOSLINE

Riporto di seguito i risultati delle analisi condotte sul database BIOSLINE dopo aver rimosso il controllo sulla tabella DocRig in fase di cancellazione di un record da CliFor.

Database BIOSLINE CON trigger

- **Tabella CliFor**

Media = 19,03303333

Varianza corretta = 1,026744033

Intervallo di Confidenza:

[18,65466688 - 19,41139978]

Database BIOSLINE SENZA trigger

- **Tabella CliFor**

Media = 0,5783

Varianza corretta = 0,113110838

Intervallo di Confidenza:

[0,452716195 - 0,703883805]

CliFor:

$(\overline{y_1} - \overline{y_2}) =$ 18.45473

ν = 35.313

t^{oss} = 94.6768 $pvalue \cong 0$

Interpretazione dei risultati:

Nel database BIOSLINE, dopo aver rimosso dal trigger di cancellazione il controllo su DocRig, si nota che il tempo medio di cancellazione è diminuito di molto rispetto a quando il controllo era presente. Naturalmente però il test t di Welch conferma che la differenza tra le medie nelle due popolazioni è ancora significativamente diversa da zero, quindi il trigger fa aumentare il tempo medio di cancellazione.

L'aver diminuito drasticamente il tempo di cancellazione, però, è un risultato davvero molto importante perché garantisce tempi di risposta accettabili anche utilizzando calcolatori non molto potenti.

7. Conclusioni

Giunti alla fine di questa relazione, è il momento di tirare le somme dell'esperienza vissuta.

Questo stage secondo me è stato davvero molto utile perché mi ha permesso di vedere applicati gli studi fatti sia nel campo dell'informatica che nel campo della statistica, e in particolare sono riuscito a vedere un'applicazione della statistica come supporto all'informatica. Gli argomenti trattati sono stati tutti davvero interessanti in quanto hanno riguardato, in generale, l'ambito dell'integrità dei dati all'interno dei database, argomento, a mio parere, di fondamentale importanza. Inoltre mi è particolarmente interessato il lavoro di test svolto perché mi ha permesso di riflettere sulla difficoltà di conciliare tre ambiti contrastanti delle basi di dati quali l'integrità dei dati, la velocità di risposta del database e la dimensione del database. Infatti, come dimostrato dalle analisi che ho condotto, in ogni database è di fondamentale importanza la qualità dei dati, ma i controlli che questa comporta contrastano con l'esigenza di velocità di risposta, e inoltre sulla velocità incide moltissimo la dimensione del database stesso.

Altro aspetto che mi è piaciuto e che ho trovato utile è stato il tipo di organizzazione del lavoro. Mi è stato infatti chiesto di lavorare con una certa autonomia ma di domandare in caso avessi bisogno di aiuto. Il lavoro in autonomia, per quanto inizialmente possa sembrare difficile, è invece importantissimo perché permette di riflettere su ciò che bisogna fare, senza trovare tutto pronto. Inoltre, lavorando autonomamente, si commettono degli errori, dai quali si può imparare molto. Personalmente ho cercato sempre di riflettere da solo su tutti i passi da seguire per portare a termine il lavoro che mi è stato assegnato, e anche quando ho chiesto aiuto ho cercato sempre di avere già in mente qualche possibile soluzione al problema; così facendo credo di aver fatto un lavoro di approfondimento degli argomenti che invece non avrei potuto fare se fossi stato sempre aiutato e "sorvegliato" da qualcuno.

Ultimo aspetto positivo, ma non meno importante dei precedenti, che mi ha permesso di lavorare con tranquillità, è stato l'ambiente di lavoro che ho trovato

all’interno dell’azienda. Sia il tutor aziendale che i colleghi con cui ho avuto il piacere di lavorare sono stati sempre cordiali e di aiuto nel risolvere i vari problemi che ho dovuto affrontare.

In conclusione, reputo l’esperienza vissuta ampiamente all’altezza delle mie aspettative. Non nascondo di aver avuto un momento di forte preoccupazione nei primi giorni di lavoro, ma che poi si è risolto e mi ha permesso di portare a termine un progetto secondo me molto interessante.

8. Bibliografia e Sitografia

- Antonio Albano, Giorgio Ghelli, Renzo Orsini, Fondamenti di basi di dati, Zanichelli editore

- Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, Basi di dati – modelli e linguaggi di interrogazione, McGraw-Hill

- Inferenza Statistica I – dispensa per le lezioni, Adriana Brogini

- Domenico Piccolo, Statistica per le decisioni – La conoscenza umana sostenuta dall'evidenza empirica, Il Mulino

- http://en.wikipedia.org/wiki/Welch's_t_test - Welch's Test

- http://biol09.biol.umontreal.ca/BIO2041e/Correction_Welch.pdf - t-test with Welch correction

- http://beheco.oxfordjournals.org/cgi/content/full/17/4/688 - The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test

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