UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Scienze Statistiche

Corso di laurea triennale in Statistica e Tecnologie Informatiche

Tesi di laurea

STUDIO DI UN SISTEMA EPIDEMIOLOGICO INTEGRATO CON IMPLEMENTAZIONE DI UN ALGORITMO PER LA SORVEGLIANZA DEL DIABETE

Relatore: Prof. Massimo Melucci

Laureando: Mattia Sardella

Matricola n. 618276

Anno accademico 2014-2015

Indice

Capitolo 1

Introduzione

Lo scopo principale del progetto di stage seguito era finalizzato alla progettazione e all'implementazione di una base di dati unica sulla quale poter mandare in esecuzione degli algoritmi in linguaggio SQL inerenti allo studio di alcune patologie, come ad esempio, il diabete o la cardiopatia ischemica.

Questi algoritmi, all'origine, erano stati scritti con un software chiamato SAS.

L'idea di base iniziale era la creazione di una base di dati dalla quale qualsiasi ASL della regione potesse estrapolare dati d'interesse ed avere un quadro generico dei pazienti residenti nel territorio di competenza.

Lo stage è stato svolto presso il Dipartimento Medicina Molecolare di Padova (sede di Igiene).

In questa relazione verranno descritti i procedimenti relativi la raccolta e la pulizia dei dati, la scrittura degli algoritmi e le scelte adottate durante il percorso.

I dati che avevamo a disposizione si riferivano solamente a due ASL (delle quali non diremo il nome per il rispetto della privacy).

Gli algoritmi implementati sono pressoché identici per entrambe, quindi mi limiterò a prenderne in considerazione solamente uno che verrà spiegato nelle prossime pagine.

Capitolo 2

Analisi del Progetto

2.1 Considerazioni iniziali

La disponibilità di archivi sanitari elettronici correnti è andata aumentando in maniera considerevole negli ultimi due decenni.

Allo scopo di confrontare le esperienze disponibili e verificare l'applicazione di procedure standardizzate, si ritiene indispensabile definire degli algoritmi per la stima delle frequenze di alcune delle maggiori patologie che affliggono la popolazione italiana.

Per il nostro progetto si è presa in considerazione la categoria diagnostica del diabete.

Le fonti sanitarie correnti prese in considerazione sono cinque:

- Certificati di morte
- Schede di dimissione ospedaliera
- Prescrizioni farmaceutiche
- Esenzioni da ticket
- Mobilità passiva

Un primo problema riscontrato è stata la disomogeneità nella disponibilità temporale dei dati, nelle dimensioni e nell'apparente qualità degli archivi.

In Italia l'esperienza più interessante riguardante l'uso di archivi elettronici a fini epidemiologici è stata realizzata nel campo della registrazione dei tumori. In Friuli-Venezia Giulia gli archivi sanitari, gestiti in un'unica warehouse centralizzata, sono utilizzati come base per costruire il sistema di registrazione dei tumori della Regione.

Da questa prima esperienza nasce l'esigenza di estendere l'utilizzo di questi strumenti informatici per la stima delle frequenze di altre patologie ai fini di migliorare le conoscenze sullo stato della salute della popolazione e per consolidare un programma di gestione e programmazione della sanità pubblica.

Grazie ai regolamenti regionali, gli archivi sono generalmente a sé stanti, organizzati e gestiti separatamente presso, almeno in parte, tutte le aziende sanitarie.

2.2 Il Sistema Epidemiologico Integrato (SEI)

Per Sistema Epidemiologico Integrato (abbreviato con SEI) s'intende un sistema di raccordo tra più fonti elettroniche sanitarie.

La banca dati del SEI è stata costituita nel 2002 per rispondere all'esigenza di supporto empirico alla programmazione sanitaria e per sfruttare la disponibilità informativa per finalità epidemiologiche.

Questa struttura permette di interrogare la banca dati integrata ricavando informazioni raccolte da fonti diverse che, riferendosi agli stessi pazienti, permettono di ottenere un quadro più articolato delle situazioni di ricorso al sistema sanitario, accrescendo la portata informativa posseduta singolarmente dagli archivi.

Gli obiettivi di questo sistema si possono riassumere come:

- Ottenere stime sufficientemente aggiornate della frequenza delle malattie nella popolazione;
- Incrociare i propri risultati con quelli ottenuti da parte di altri Enti preposti al controllo del territorio;
- Costituire una struttura di supporto essenziale per l'Azienda nella pianificazione degli interventi di sanità pubblica;
- Divulgare periodicamente informazioni sintetiche e strutturate sullo stato di salute della popolazione.

Questo sistema è chiarito nella pubblicazione "Epidemiologia e Prevenzione" dove, oltre ad altri algoritmi, è spiegato con precisione quello preso in considerazione ossia l'algoritmo di stima del diabete che si basa proprio sul sistema SEI utilizzando come input le seguenti fonti elettroniche sanitarie:

- Certificati di morte, contiene i dati relativi ai deceduti presso il territorio dell'Azienda;
- SDO (schede di dimissione ospedaliera), contiene i dati riguardanti le schede di dimissione ospedaliera;
- MP (mobilità passiva), contiene i dati concernenti la mobilità passiva, ossia quelle persone che hanno richiesto assistenza sanitaria al di fuori del proprio territorio di competenza;
- ET (esenzioni ticket), contiene i dati che si riferiscono alle esenzioni ticket
- PF (prescrizioni farmaceutiche), contiene i dati relativi alle prescrizioni farmaceutiche.

2.3 Metodi d'indagine

L'idea essenzialmente è quella di estrapolare i casi di una malattia incrociando le informazioni di più archivi sanitari elettronici correnti senza ricorrere alla raccolta attiva della casistica.

Ogni contatto tra assistito e servizio sanitario, di norma, è registrato in un archivio specifico e riporta informazioni che possono permettere di risalire alla motivazione del contatto; inoltre la maggior parte delle diagnosi di malattia è eseguita sulla base dei risultati ottenuti da esami specifici.

Ovviamente questa metodologia di "ricerca" non può essere applicata nei casi in cui gli archivi sanitari elettronici risultino incompleti e di bassa qualità.

2.4 Record Linkage

Per record linkage s'intende la procedura utilizzata per determinare se due record, appartenenti a due diversi set di dati, si riferiscono a uno stesso individuo.

In SQL questa procedura può essere vista come una sorta di JOIN tra tabelle diverse.

Le procedure utilizzate di record linkage sono tutte di tipo deterministico, ovvero basate sull'accordo esatto dell'insieme delle caratteristiche (campi) che costituiscono la chiave identificativa di un individuo (record di una tabella).

Generalmente il linkage viene fatto con l'anagrafe sanitaria o comunale.

2.5 Metodi per il calcolo delle stime

Gli archivi elettronici correnti utilizzati per le stime sono quelli contenenti le cause di morte (CM), le schede di dimissione ospedaliera (SDO), le prescrizioni farmaceutiche (PF) e le esenzioni ticket (ET) .

Ognuno di questi archivi non risaliva allo stesso anno. Nasce da qui l'esigenza di fissare una data di partenza comune, cosicché non si dovesse tener conto di eventuali trend temporali e del diverso apporto delle fonti nel tempo, condizione che rendeva impossibile un confronto dei risultati ottenuti, una volta implementato l'algoritmo.

2.6 Misure d'incidenza/prevalenza

In relazione alle caratteristiche delle fonti informative utilizzate e delle patologie studiate, per alcune malattie è stato possibile stimare misure di incidenza, per altre misure di prevalenza.

L'incidenza è la frequenza con cui si misura quanti nuovi casi di una data malattia, in questo caso il diabete, compaiono in un determinato lasso di tempo mentre la prevalenza è il rapporto fra il numero di eventi sanitari rilevati in una popolazione in un definito momento (arco temporale) e il numero di individui della popolazione osservati nello stesso periodo.

Per quanto riguarda la prevalenza, è stato scelto di calcolarla in due modi differenti a seconda delle caratteristiche della malattia e della conseguente necessità di ricorso a prestazioni sanitarie:

- definendola come numero di malati presenti nella popolazione nell'anno t, calcolati sulla base delle fonti riferite a quell'anno;

- utilizzando alcune fonti (SDO, ET) nell'anno di stima e in un intervallo temporale precedente; in questo modo si aumenta la sensibilità della stima per quelle patologie croniche per le quali gli accessi ad alcuni servizi sanitari avvengono solitamente con cadenza superiore ai 12 mesi.

Nel nostro caso, per il diabete, è stato ritenuto che le fonti informative delle prestazioni sanitarie usufruite nell'anno di stima fossero insufficienti ad assicurare una buona copertura nell'identificazione dei casi prevalenti.

Pertanto sono stati inclusi anche i pazienti trattati in anni precedenti e vivi all'inizio dell'anno di stima della prevalenza (per precisazione, sono stati considerati i quattro anni precedenti all'anno di stima).

L'incidenza, invece, è rappresentata dai nuovi casi di malattia in un determinato anno (ovvero diagnosticati per la prima volta in un paziente).

Nel caso del diabete è stata calcolata la prevalenza considerando la data di dimissione come data dell'evento.

Oltre alle fonti citate precedentemente, è stato utilizzato anche l'archivio della mobilità passiva per non escludere dalla stima i casi appartenenti alla popolazione in studio che hanno usufruito di servizi presso strutture afferenti ad altre aziende sanitarie.

Nella formula per il calcolo dei tassi annuali, al denominatore, si è utilizzata la popolazione residente al 30 giugno dell'anno, ricavata da fonte Istat.

Dopo aver implementato il nostro metodo, abbiamo calcolato il contributo di ciascuna fonte distinguendo un "contributo esclusivo", determinato da soggetti identificati come malati per mezzo di quell'unica fonte, e come "contributo assoluto" il numero di casi in cui la fonte considerata ha contribuito all'identificazione.

Per costruzione, la somma dei contributi esclusivi per le diverse fonti e dei soggetti identificati dalla combinazione di più di una fonte è pari al 100% dei casi individuati mentre, la somma dei contributi assoluti è maggiore o uguale al 100%.

Entrambi i contributi sono stati calcolati stratificando per fasce di età, sesso e anno di stima della popolazione.

All’ inizio è stato necessario raccogliere tutte le informazioni di cui potevamo disporre in quel momento (tipo dei DB iniziali, programmi che offre il mercato ecc.).

Fin da subito abbiamo optato per utilizzare dei software open source soprattutto per un fattore economico in quanto, essendo un progetto, il budget era abbastanza ridotto ed esisteva la possibilità che qualche ASL del territorio non avesse i fondi necessari per pagare le licenze software facendo così svanire l’idea di base di unire i dati e renderli disponibili a tutti gli enti competenti.

2.7 Anonimizzazione

Per anonimizzazione si intende il processo che rende le tabelle iniziali non riconducibili, per motivi di privacy, ai singoli pazienti.

Il metodo usato è molto semplice:

- Sono state create delle tabelle con la stessa struttura di quelle iniziali;
- Sono stati aggiunti dei campi identificativi, usati come chiave primaria delle nuove tabelle;
- Sono stati cancellati i campi che identificavano le singole persone, come ad esempio il nome e il cognome;
- Sono stati cancellati, inoltre, i campi “superflui” ai nostri fini;

Adottando queste modifiche, le nuove tabelle risultano identiche a quelle iniziali ma non è possibile ricondursi ad una specifica persona.

D’ora in poi si utilizzeranno queste tabelle come fonti iniziali per il nostro algoritmo del Diabete.

Capitolo 3

Il Linguaggio SQL

In informatica SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- Creare e modificare schemi di database (DDL – Data Definition Language);
- Inserire, modificare e gestire dati memorizzati (DML – Data Manipulation Language)
- Interrogare il DB (DQL – Data Query Language);
- Creare e gestire strumenti di controllo dei dati (DCL – Data Control Language)

La conoscenza di questo linguaggio è fondamentale per chi, come me, ha l'esigenza di interrogare e gestire DB più o meno grandi.

Il vero punto di forza sta nella semplicità della sintassi.

3.1 MySQL Workbench

MySQL Workbench è uno strumento visuale di progettazione per database con il quale è possibile creare, gestire e modificare un database MySQL, il tutto dentro lo stesso ambiente sinergico.

Le funzionalità principali che copre sono principalmente cinque:

- Sviluppo SQL: consente di creare e gestire le connessioni ai server database, ad esempio permette il settaggio dei parametri di connessione;
  Oltre a questo, offre la possibilità di eseguire query SQL utilizzando il proprio editor integrato;

- Data Modeling (Design): consente di creare i modelli dello schema database in modo grafico, oltre a permettere la modifica di colonne, indici, creare trigger e molto altro;
- Administration Server: permette di gestire le istanze dei server MySQL come, ad esempio, la creazione di utenti, l'esecuzione di backup e pure il ripristino;
- Data Migration: consente la migrazione da Microsoft SQL Server, Microsoft Access, SQLite e altre tabelle da diversi RDBMS in MySQL;
- Data Migration: consente la migrazione da Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, e altre tabelle RDBMS, oggetti e dati in MySQL.
- MySQL Enterprise Support: supporto per i prodotti aziendali come MySQL Enterprise Backup e MySQL Audit.

Nella Figura 1 è possibile vedere come la pagina iniziale sia suddivisa in tre settori:

- SQL Development, nella quale è possibile accedere ai DB esistenti e, al passo successivo, cominciare a scrivere Query d'interrogazione, script o eseguire modifiche ai DB Object;

- Data Modeling, creare e amministrare modelli;

- Server Administration, dove è possibile configurare i propri database server, amministrare gli account utente e molte altre funzioni server.

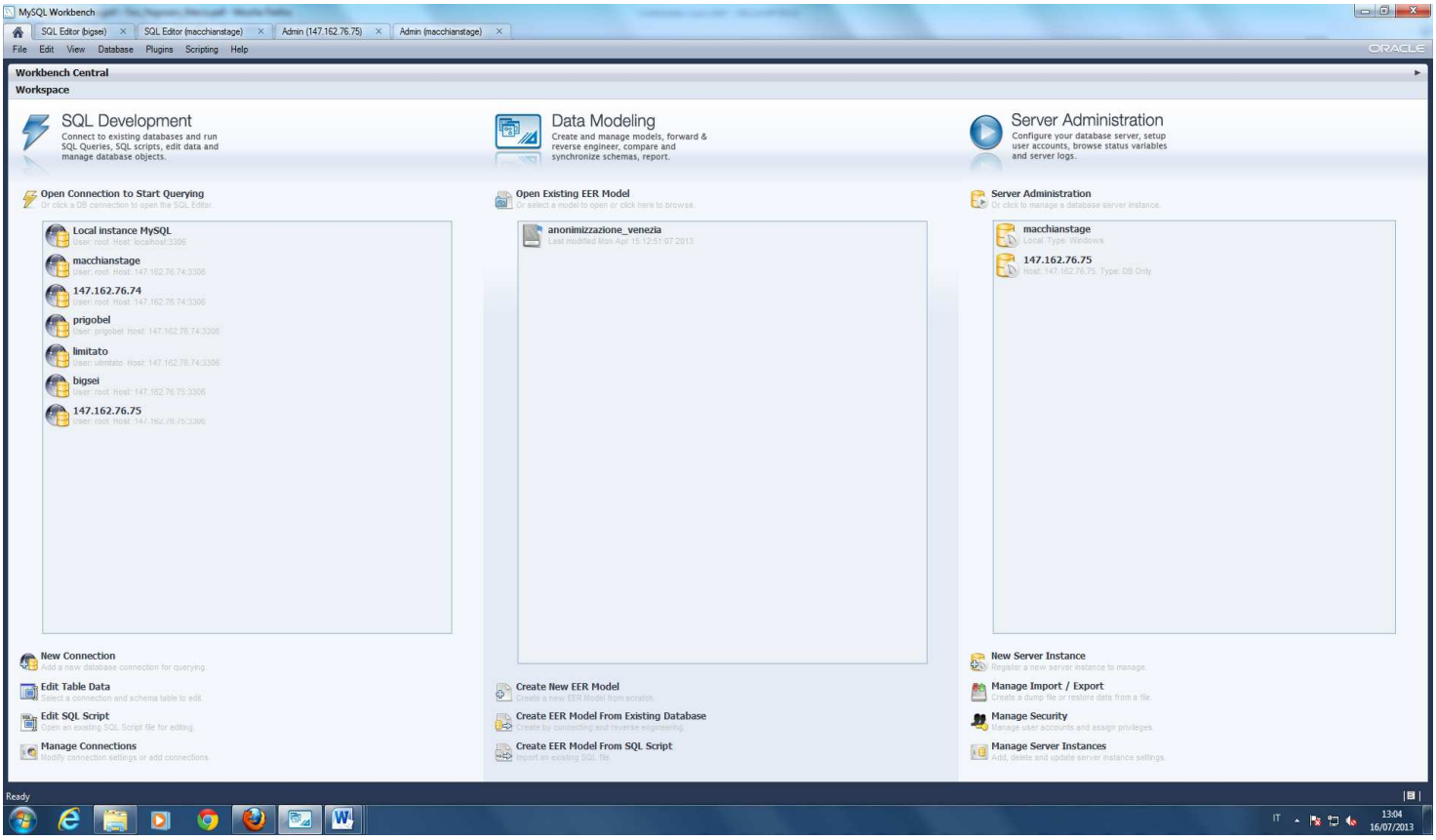

Figura 1: Pagina iniziale di MySQL Workbench

3.2 Esempio Query SQL

La struttura di base per un'interrogazione (query) in Mysql è questa:

- SELECT * FROM nome_tabella

Questa query seleziona TUTTI i campi (* significa proprio questo) dalla tabella "nome_tabella".

Se si vogliono filtrare i dati per uno o più criteri, si utilizza la clausola WHERE:

- SELECT * FROM nome_tabella WHERE nome_attributo =…

Nella clausola WHERE è possibile inserire più regole di selezione (unite dagli operatori logici AND oppure OR) e sono disponibili tutti gli operatori matematici (=, +, -, /, ecc.).

E’ possibile raggruppare i risultati e selezionarne alcuni attraverso la clausola GROUP BY e inserendo la condizione con la parola HAVING:

- SELECT campoDB, sesso FROM nomeTabella
  GROUP BY campoDB
  HAVING sesso= ’ M ’

Questa query ha senso se nella nostra tabella abbiamo più “campoDB” ripetuti.

La clausola INNER JOIN permette l’incrocio di due o più tabelle attraverso un qualsiasi attributo (contenuto in entrambe le tabelle):

- SELECT * FROM tab1 INNER JOIN tab2 ON tab1.attr1 = tab2.attr2
  WHERE tab1.attr3 > …

Con questa query vengono selezionati tutti i record che hanno attr1 e attr2 uguali (da notare che attr1 e attr2 sono in due tabelle diverse), e vengono filtrati per att3 > …

I record che non soddisfano la corrispondenza specificata nella clausola ON non vengono presi in considerazione.

Esistono anche altri tipi di JOIN :

_ LEFT JOIN: vengono presi tutti i record che soddisfano la condizione di join ma anche quelli della prima tabella selezionata (tabella a sinistra della JOIN) che non la soddisfano.

_ RIGHT JOIN: è come la left join ma vengono presi i record della tabella a destra della JOIN (non quelli a sinistra).

Esempio di query:

Con la seguente query uniamo le esenzioni ticket delle due ASL e filtriamo per l’anno 2006:

```
SELECT campoDB,anno_inizio_val,codice_esenzione FROM
(SELECT * FROM ASL1.ET
UNION ALL
(SELECT * FROM ASL2.ET)) as d
```

WHERE anno_inizio_val=2006

Con la clausola UNION, uniamo le due tabelle in una unica.
Le due tabelle (nel caso in esempio le ET) devono avere gli stessi campi (numero di colonne e nome) per essere unite.

3.3 Viste

Le viste (view) sono comunemente considerate un modo per mostrare i dati di un database con una struttura diversa da quella che hanno effettivamente sulla base dati.

Un uso possibile delle viste è quello di concedere ad un utente l'accesso ad una tabella mostrandogli solo alcune colonne: tali colonne saranno inserite nella vista, sulla quale l'utente avrà i permessi di accesso, mentre gli saranno negati quelli sulla tabella sottostante.

Altre possibili applicazioni riguardano la possibilità di leggere dati da più tabelle contemporaneamente attraverso JOIN o UNION, oppure di comprendere dati che non sono fisicamente presenti sulla tabella in quanto calcolati a partire da altri.

Le viste non possono avere lo stesso nome di una tabella facente parte dello stesso database. Ecco la sintassi (minima) da usare per la creazione di una vista:

CREATE VIEW *nome* [(*lista_colonne*)]
  AS *istruzione_select*

L'istruzione SELECT è quella che definisce quali dati (colonne) saranno visualizzati.
Questa istruzione può essere anche molto complicata, ma non può avere al suo interno delle subquery.

Esempio di View:

```
CREATE VIEW contr_assoluto_anno AS
   (select
      contributo.anno AS anno,
      sum(contributo.apf) AS apf,
      count(contributo.campoDB) AS totale,
      ((sum(contributo.apf) / count(contributo.campoDB)) * 100) AS %apf
   from
      soggetti_diabete as contributo
   group by contributo.anno)
```

Questa vista calcola il contributo assoluto raggruppato per anno. Nel caso in esempio prendiamo in considerazione solo il campo APF (farmaci).

I nomi scritti dopo la parola AS sono detti alias, ossia sono dei nomi che l'utente può mettere a sua discrezione e che verranno visualizzati al posto del nome "reale" della colonna.

Il risultato finale sarà una tabella del tipo:

ANNO	APF	TOTALE	%APF
…	…	…	…

Nella prima parte del progetto, sono state utilizzate solo delle view per implementare l'algoritmo del diabete.
In esecuzione è stato osservato che, avendo due viste identiche di partenza, se ne prendevamo una e la spezzavamo, il risultato era diverso da quello dato dall'altra view.

Alla fine si è deciso di procedere compattando il più possibile le viste, rendendo anche il tutto più facile da modificare in futuro.

3.4 Stored Procedure

Una stored procedure (procedura) è un insieme di istruzioni SQL che vengono memorizzate nel server con un nome che le identifica; tale nome consente in seguito di rieseguire l'insieme di istruzioni facendo semplicemente riferimento ad esso.

Creazione di una procedura:

CREATE PROCEDURE *nome* ([*parametro*[,...]])
corpo
//*parametri:*
[IN | OUT | INOUT] *nomeParametro tipo*

Come detto in precedenza, ogni procedura è identificata da un nome. Inoltre la store procedure è attribuita ad uno specifico database esattamente come una tabella. Di conseguenza viene assegnata ad un DB al momento della creazione, ed i nomi referenziati al suo interno si riferiranno allo stesso.

Ogni procedura può avere uno o più parametri, ciascuno dei quali è formato da un nome, un tipo di dato e l'indicazione se si tratta di parametro d'input, di output o entrambi. Se manca l'indicazione, il parametro è considerato d'input.

Chiamata della procedura:

CALL *nome* (*parametro1, parametro2, …*);

Nella chiamata, i parametri son passati tra virgolette se sono di tipo stringa o data, oppure scritti normalmente in caso siano numeri.

Esempio:

```
DELIMITER $$

CREATE PROCEDURE `et`(in codice varchar(15), in annosup int(4), in tabet varchar(20))

BEGIN

set @a=concat('SELECT * from ',tabet,' WHERE anno= ',annosup,' and codice_esenzione= ',codice,' ' );

PREPARE stmt from @a;
```

EXECUTE stmt;

END

Questa procedura seleziona dalla tabella che passiamo come parametro (tabet) i record che hanno il campo anno = ‘annosup’ e il campo codice_esenzione = ‘codice’.

Il comando DELIMITER, serve al client MySQL per modificare il normale delimitatore delle istruzioni, che sarebbe il punto e virgola. Infatti, siccome la stored procedure contiene più istruzioni, al suo interno il punto e virgola viene utilizzato più volte. Di conseguenza, per memorizzare la procedura, si deve comunicare al server che il delimitatore è un altro; in caso contrario, al primo punto e virgola penserebbe che la nostra CREATE sia terminata. Decidiamo quindi che il nuovo delimitatore sia un doppio $.

I parametri che vogliamo passare alla procedura sono dichiarati all’inizio.

Tra il blocco BEGIN – END si trova la query SQL che vogliamo eseguire; questa è all’interno della funzione concat poiché è una concatenazione tra una stringa fissa (SELECT …) e i parametri che vogliamo passare.
Il risultato della query è salvato nella variabile @a .

Con i comandi PREPARE ed EXECUTE si prepara lo statement a partire dalla variabile settata col risultato della query (@a) ed eseguiamo il tutto.

Per chiamare questa funzione bisognerà scrivere:

CALL et (‘013250’, ‘2008’ , ‘ET’)

E’ possibile dichiarare delle variabili all’interno di una procedura. Esse avranno valenza solo all’interno della funzione.

Per dichiarare una variabile bisogna scrivere:

DECLARE *nome tipo*;

Questa dichiarazione va inserita subito dopo il BEGIN.
Ogni dichiarazione deve essere separata da un ”;”

Esempio:

BEGIN

```
DECLARE  campoDB VARCHAR(9);
DECLARE  anno  INT(4);
.
.

END
```

Nel progetto le procedure sono state ampiamente utilizzate poiché rendono gli algoritmi parametrici e quindi permettono una sorta d'iterazione con l'utente.

Per la seconda ASL, ad esempio, non sono state implementate delle viste se non per calcolare delle statistiche sulle tabelle finali, create da delle procedure.

Le stored procedure sono più simili ad un programma se inglobate una dentro all'altra.
Le view, invece, devono essere eseguite una alla volta, rendendo il tutto meno "fluido" agli occhi di un utente.

Capitolo 4

SAS

Il SAS, acronimo per Statistical Analysis System, è uno dei software più utilizzati per condurre analisi che vanno da semplici statistiche descrittive e inferenziali alle più complesse tecniche statistiche multivariate. La sua popolarità è imputabile alla capacità di gestire grossi volumi di dati utilizzando procedure già implementate, disponibili in numerosi moduli che rispondono a esigenze diverse, oppure programmando personalmente le analisi che si desidera effettuare mediante un linguaggio di programmazione di alto livello.

Esiste un modulo di base, denominato appunto BASE, che consente di compiere analisi statistiche elementari, a cui si affiancano vari moduli per applicazioni più specifiche quali ad esempio:

- STAT per applicare metodologie statistiche complesse
- QC per il controllo qualità
- OR per la ricerca operativa

- TSA per l’analisi di serie temporali
- ML per la manipolazione di matrici
- GRAPH per applicare tecniche grafiche complesse

Come già detto, è anche possibile utilizzare il SAS come un qualsiasi linguaggio di programmazione. In teoria è infatti possibile programmare una qualsiasi analisi statistica mediante questo software, anche se in generale questa programmazione ad hoc non è necessaria perché le analisi statistiche comunemente utilizzate sono effettuabili in modo molto semplice utilizzando le procedure (denominate PROC) che sono contenute nei diversi moduli del SAS.

Capitolo 5

Standardizzazione dati iniziali

I due algoritmi presi in considerazione per questo progetto erano inizialmente scritti utilizzando il programma SAS mentre le fonti potevano provenire sia da SAS che da Access.

Per questo motivo c'era il bisogno di trasformare i dati in tabelle da elaborare mediante SQL.

Prima di far ciò è stato necessario "studiare" i record delle tabelle, per capire la loro struttura e ricercare eventuali errori d'inserimento da parte degli operatori.

I'operazione di Record Linkage è stata eseguita precedentemente al mio stage, quindi tutte le tabelle cosiddette "iniziali" teoricamente dovevano risultare pronte all'uso.

Analizzandole si è notato come, in alcuni casi, alcuni campi risultassero incompleti o con errori evidenti d'inserimento ma, fortunatamente, tutti correggibili perché intuitivi.

Ogni campo è stato codificato con il relativo tipo in MySQL così da adattare il tutto alle nostre esigenze (ad esempio una variabile di tipo CARATTERE in SAS diventerà VARCHAR in MySQL).

In seguito si è studiata una struttura "standard" per le tabelle così, in caso di aggiornamento dei DB, i nuovi dati si sarebbero adattati perfettamente così da rendere più agevole e veloce il tutto (gli algoritmi implementati hanno bisogno di determinati campi nelle tabelle SQL).

Le tabelle che importiamo hanno uno schema di attributi ben definito e, l'unica modifica che può essere apportata, dopo l'importazione, è l'aggiunta di indici sui campi che prendiamo in maggior considerazione nei nostri algoritmi, il tutto per aumentare le prestazioni ed avere dei risultati in un minore tempo.

Per questo motivo sono stati aggiunti, nelle tabelle dove non era già possibile identificare univocamente i record, uno o più campi che fungessero da chiave primaria.

Questa scelta ha portato molti vantaggi per quanto riguarda la velocità di esecuzione delle query e anche nei passi di JOIN all’interno degli algoritmi.

Capitolo 6

Importazione dei dati

Vediamo ora come importare i dati da Access e da SAS:

- DA ACCESS

Per l'importazione di tabelle in formato Access, ci viene in aiuto il programma open-source Bullzip.

Questo ci permette di creare un file dump (scritto con la sintassi di MySql) dalla quale è possibile copiare il codice sorgente (aprendo il file con un editor di testo) e incollarlo sulla finestra di Mysql Workbench.

Altra possibilità è importare il tutto attraverso la finestra di import nella parte chiamata "Server Administrator", sempre in Mysql Workbench.

Essendo Access un RDBMS, l'import in MySQL comprende, oltre ai dati ovviamente, la struttura completa della tabella (indici, chiavi primarie ecc.).

Le modifiche che abbiamo apportato quindi si sono limitate all'aggiunta di un campo identificativo.

- DA SAS

Per prima cosa bisogna collegarsi al database attraverso il seguente comando SAS:

libname mydb mysql server='indirizzo IP' user=root password='password' database='nomeDB' ;

Dove:
mydb → Nome che assegniamo alla libreria
Server→ Indirizzo dove si trova il nostro DB

User → Solitamente è root
Password → Password per accedere al DB
Database → Nome nel DB alla quale vogliamo collegarci

Dopo esserci collegati, mandiamo in esecuzione la copia dei dati da SAS a Mysql:

```
PROC COPY IN=mort OUT=mydb MEMTYPE=ALL;
  SELECT nomeTabellaSAS;
RUN;
```

Dove:
In → Nome della libreria dove si trova la tabella SAS da copiare
Out →Nome della libreria dove si trova il collegamento a Mysql (creato con il precedente comando)
Dopo il SELECT →Nome della tabella SAS da copiare

Alla fine di questa procedura, si avrà la stessa tabella contenente tutti i dati.

In SAS, a differenza di Access, viene effettuata una vera è propria copia dei dati, ossia non ci sono altre informazioni (indici, ecc.) sulla struttura della tabella quindi bisogna procedere alla creazione "manuale" attraverso delle query di modifica.

Il procedimento di import può essere così riassunto:

Schema 1: Procedimento di import dei dati

Capitolo 7

Tabelle Iniziali

Dopo aver importato i dati come descritto in precedenza, la situazione in MySQL era questa:

- SDO
- MP
- ET
- FARMACEUTICA
- MORTALITA'
- ANAGRAFICA
- ASL

Tabella SDO:

Contiene le informazioni riguardanti le SDO (schede di dimissione ospedaliera) dell'ASL presa in considerazione.

Tabella MP:

Contiene le informazioni principali sugli assistiti che richiedono delle prestazioni sanitarie al di fuori del proprio territorio di competenza (Mobilità passiva).

Tabella ET:

Informazioni sugli assistiti che godono di un esenzione ticket.

Tabella FARMACEUTICA:

Archivio dei farmaci prescritti in un certo arco temporale.

Tabella MORTALITA':

Informazioni sui deceduti che hanno come ASL assegnata quella presa in considerazione.

Tabella ANAGRAFICA:

Dati anagrafici degli assistiti dell'ASL considerata.

TABELLA ASL:

Popolazione dell'ASL suddivisa per anno, sesso e classe d'età.

In seguito sono stati creati gli indici sui campi maggiormente presi in considerazione dagli algoritmi (ricerche e selezioni):

- FARMACEUTICA
    - campoDB VARCHAR(10)
    - atc_farm VARCHAR(8)
    - anno_prescr INT(4)
    - id (creato da noi come identificativo univoco, chiave primaria) BIGINT(20) AUTOINCREMENT

- ET
    - campoDB VARCHAR(9)
    - anno INT(4) (campo inserito successivamente)

- ANAGRAFE
    - data_nascita DATE
    - campoDB (chiave primaria) VARCHAR(9)

- ASL
    - g_eta2 VARCHAR(2)
    - sesso VARCHAR(1)
    - anno INT(4)

  L’unione di questi tre campi è chiave primaria

- MORTALITA
    - data_morte DATE
    - data_nascita DATE
    - campoDB (chiave primaria) VARCHAR(9)

- MP
    - campoDB VARCHAR(9)
    - diap VARCHAR(5)
    - dia1 VARCHAR(5)
    - dia2 VARCHAR(5)
    - dia3 VARCHAR(5)
    - dia4 VARCHAR(5)
    - dia5 VARCHAR(5)

- SDO
    - campoDB VARCHAR(9)
    - anno INT(4)
    - diap VARCHAR(5)
    - dia1 VARCHAR(5)
    - dia2 VARCHAR(5)
    - dia3 VARCHAR(5)
    - dia4 VARCHAR(5)
    - dia5 VARCHAR(5)

I campi denominati “campoDB” verranno utilizzati nelle join per unire le tabelle.

Come si può notare sono stati creati molti indici, anche vista la mole di dati movimentata per ogni query (che sia una semplice SELECT oppure una JOIN più o meno complessa).

Capitolo 8

Implementazione algoritmo del Diabete

Possiamo dividere il processo d'implementazione in due fasi:

1) Implementazione dell'algoritmo attraverso l'uso di Viste SQL
2) Implementazione dell'algoritmo attraverso l'uso di Procedure SQL

Non esiste un particolare motivo per il quale sono stati utilizzati questi due metodi. Essendo un progetto, si è potuto verificare e sperimentare questi due modi diversi per ottenere gli stessi risultati così da poter mettere a confronto pregi e difetti di entrambi.

Capitolo 9

Sviluppo del progetto

9.1 Elaborazione dei dati attraverso le Viste

SDO e MP

- **Sdo_mp_linkati:**

  In questa vista vengono collegate le SDO e le MP all'anagrafica dell'ASL e successivamente vengono filtrati i record per il codice diagnosi che identifica il diabete ossia '250'

- **Temp:**

  Collego la mortalità all'anagrafica filtrando per anno di morte < dell'anno preso in considerazione.
  In seguito seleziono la finestra temporale (anni 4+1) e raggruppo per campoDB.

FARMACI

- **Farm:**

  Collego i farmaci filtrati per codice ATC 'A10' e anno di prescrizione = anno di stima all'anagrafica.
  Raggruppo per campoDB e tengo solamente i record con più di due prescrizioni.

ET

- **Et_diab_linkati:**

  Collego le esenzioni ticket con codice = '013.250' all'anagrafica.

- **Et_diabl_2003:**

  Linko le esenzioni ticket alla mortalità filtrata per anno di morte < anno di stima e raggruppo per campoDB.

- **Unione_tot_n :**

  Questa vista si limita ad unire i risultati delle tre viste “Temp”, “Farm”, “Et_diabl_2003” e ad inserire in ognuna di queste un campo denominato “anno” alla quale do il valore dell’anno di stima.

  Per la vista “Et_diabl_2003” la finestra temporale è di anni 3+1

- **Contributo:**

  In questa view vengono calcolati i contributi assoluti ed esclusivi.

  Dopo aver creato queste viste, è stato inserito nella tabella ‘**soggetti_diabete’** i record per anno di stima:

- Insert into soggetti_diabete (select * from contributo where anno=ANNO DI STIMA)

All’interno di questa tabella ora c’è tutta la popolazione considerata diabetica dell’ASL presa in considerazione.

I risultati vengono salvati quindi in questa tabella per avere un risultato fisico sul quale poter fare tutte le verifiche che si vogliono e calcolare qualsiasi statistica.

9.2 Statistiche calcolate sulla tabella “Soggetti_Diabete”

- **Pop_anno_eta**:

  numero di casi e calcolo del tasso grezzo diviso per anno di stima e gruppo di età (quindicinale)

- **Pop_anno_eta_2**:

  numero di casi e calcolo del tasso grezzo per anno e gruppo di età (quindicinale) con aggiunta del vincolo

  (((fonte_apf > 2) and (apf = 1) and (sdo = 0) and (et = 0))

  or (sdo = 1)

  or (et = 1))

- **Pop_anno_sesso**:

  numero di casi e calcolo tasso grezzo per anno e sesso

- **Contr_assoluto_anno**:

  contributi assoluti per anno di stima

- **Contr_escl_anno**:

  contributi esclusivi per anno di stima

- **Contr_escl_sesso**:

  contributi esclusivi per sesso

Tutte queste statistiche sono il risultato di viste appositamente scritte.

9.3 Funzioni create

- classe(data_nascita , data2 , n)

  Calcolo dell'età del soggetto e raggruppamento in classi quinquennali(n=1) o quindicinali (n=2)

Parametri:

- data_nascita: data di nascita del soggetto
- data2: altra data (es. data di morte)
- n: tipologia di raggruppamento

Risultati:

la classe d’età di appartenenza.

- contr_escl(f1,f2,f3)

  Calcolo del contributo esclusivo

Parametri:

- f1: apf
- f2: sdo
- f3: et

Risultati:

- 1 se il soggetto è identificato solo dalle sdo-mp
- 2 se il soggetto è identificato solo dai farmaci
- 3 se il soggetto è identificato solo dalle et
- 4 se è identificato da più fonti

9.4 Elaborazione dei dati attraverso le Stored Procedure

Per implementare l’algoritmo del diabete attraverso l’uso delle stored procedure sono state create delle tabelle “di appoggio” dalle quali poter prendere i dati necessari già filtrati adeguatamente.

TABELLE DI APPOGGIO CREATE

1) Farmaci
2) Sdo_mp_ultima
3) Et_new
4) Unione
5) Soggetti_diabete_new

TABELLA FARMACI

Tabella contenente i farmaci con codice ‘A10’ e con anno di prescrizione 2009 (nel caso in esempio).

Ulteriore filtro dopo il raggruppamento per campoDB (fonte_apf >= 2 sempre nel caso di esempio)

TABELLA SDO_MP_ULTIMA

Tabella contenente l’unione delle SDO e della MP filtrate per codice diabete (‘250’) e incrociate con l’anagrafe.

Incrocio con la mortalità, seleziono solo i non deceduti prima dell’anno di stima e con anno di dimissione tra (anno di stima) e (anno di stima-4)

TABELLA ET_NEW

Tabella contenente le ET con codice esenzione = ‘013.250’ e linkate con l’anagrafe.

Incrocio con mortalità, seleziono i non deceduti prima dell’anno di stima.

TABELLA UNIONE

Unione delle tabelle farmaci,sdo_mp_ultima e et_new.

TABELLA SOGGETTI_DIABETE_NEW

Tabella finale contenente tutti i record e il calcolo del contributo esclusivo.

PROCEDURE UTILIZZATE

- getfarm(anno,'farmaceutica','A10',2,'farmaci','anagrafe','n','050','*');

Parametri:

- anno: anno della prescrizione
- farmaceutica: nome della tabella iniziale dei farmaci
- A10: codice relativo alla malattia (A10 = diabete)
- 2: numero minimo di fonti per campoDB
- farmaci: nome tabella risultante
- anagrafe: nome tabella anagrafica
- n: n° della asl presa in considerazione
- 050: regione
- *: linko con tutta l'anagrafica (se è ' ', prendo in considerazione l'ultimo anno di anagrafica)

Funzionamento:

Inserisce nella tabella 'farmaci', i record (farmaceutica linkata all'anagrafe) con codice 'A10' e anno di prescrizione dato.

I record vengono raggruppati per campoDB (fonte_apf >= 2).

- sdo_mp_unione_diabete('250',anno,'sdo','mp','anagrafe','mortalita','sdo_mp_unione','n','050','*');

Parametri:

- 250: codice malattia
- anno: anno di stima
- sdo: nome tabella sdo
- mp: nome tabella mp
- anagrafe: nome tabella anagrafe
- mortalita: nome tabella mortalità
- sdo_mp_unione: nome tabella risultante
- n: n° asl presa in considerazione
- 050: regione
- *: linko con tutta l'anagrafica (se è ' ', prendo in considerazione l'ultimo anno di anagrafica)

Funzionamento:

Inserisce nella tabella sdo_mp_unione, l'unione delle SDO e della MP filtrate per codice diabete ('250') e incrociate con l'anagrafe.

Incrocio con la mortalità, seleziono solo i non deceduti prima dell'anno di stima e con anno di dimissione tra (anno di stima) e (anno di stima-4)

- et ('013250',anno,'et','anagrafe','mortalita','et_new','n','050','*');

Parametri:

- 013250: codice esenzione
- anno: anno di stima
- et: nome tabella et
- anagrafe: nome tabella anagrafe
- mortalita: nome tabella mortalità
- et_new: nome tabella risultante
- n: n° asl presa in considerazione
- 050: regione
- *: linko con tutta l'anagrafica (se è ' ', prendo in considerazione l'ultimo anno di anagrafica)

Funzionamento:

Inserisce nella tabella et_new, le ET filtrate per codice esenzione = ‘013250’ e incrociate con l’anagrafe.

Incrocio con mortalità, seleziono i non deceduti prima dell’anno di stima.

- get_unione_tot_n(anno,'farmaci','sdo_mp_unione','et_new','unione');

Parametri:

- anno: anno di stima,
- farmaci: nome tabella di appoggio farmaci
- sdo_mp_unione: nome tabella finale sdo-mp
- et_new: nome tabella finale et
- unione: nome tabella risultante

Funzionamento:

Questa procedura si limita ad unire i risultati delle tre precedenti in un'unica tabella chiamata ‘unione’.

- contributo_diabete('soggetti_diabete');

Parametri:

- soggetti_diabete: nome tabella risultante

Funzionamento:

Inserimento nella tabella denominata ‘soggetti_diabete’ di tutti i record e calcolo del contributo esclusivo.

- proc_diabete(‘diabete’)

Parametri:

- diabete: nome tabella finale

Funzionamento:

Inserimento nella tabella ‘diabete’ di tutti i record, variando l’anno dal 2002 al 2009.

Capitolo 10

Conclusioni

Attraverso questa esperienza sono stati applicati i concetti studiati, come l'implementazione di una base di dati, il linguaggio SQL e le interrogazioni attraverso l'uso di query, spaziando anche su nuovi argomenti come ad esempio la struttura di una vista SQL e di una procedura.
Da notare quanto il settore epidemiologico risulti vasto e complesso e allo stesso tempo quanto l'informatica sia versatile e applicabile in svariati ambiti.
Oltre alla categoria diagnostica del diabete, è stato sviluppato anche l'algoritmo per la cardiopatia ischemica.
Questo, risulta molto simile a quello spiegato in questa relazione e, attraverso l'uso delle procedure parametriche, risulta molto semplice riprodurlo variando i parametri.
Una volta terminato il periodo di stage, sono state constatate alcune differenze sostanziali sull'uso di viste o procedure.
Per quanto riguarda questo progetto, l'uso di procedure è risultato migliore rispetto l'uso delle viste in quanto, variando i parametri di input, è possibile ottenere risultati riguardanti diversi settori epidemiologici, cosa che non è possibile con l'uso delle seconde.
Posso ritenermi pienamente soddisfatto dell'esperienza vissuta.

RINGRAZIAMENTI

Ringrazio quanti mi hanno seguito ed aiutato nello svolgimento dello stage.
In particolare, colgo l'occasione per ringraziare il Dottor Paolo Rigobello per avermi dedicato tempo ed attenzione, consigliandomi ed aiutandomi nei momenti di maggiore difficoltà; un ringraziamento ai Prof. Vincenzo Baldo e Lorenzo Simonato, responsabili del Laboratorio di Sanità Pubblica e Studi di Popolazione (LASP) del Dipartimento di Medicina Molecolare e le componenti del gruppo Dott.sse Laura Cestari, Cristina Canova, Valentina Zabeo e Patrizia Furlan.

Un pensiero di gratitudine alla mia famiglia, che mi ha sempre sostenuto con affetto.