Schema mapping with Query Reverse Engineering

Ho scritto questo documento in italiano per questioni di chiarezza e velocità.

Di seguito riporto l'introduzione che ho trovato in un paper di Jagadish. Credo sia molto utile per inquadrare il contesto e anche per capire quali sono i limiti dei sistemi di schama mapping "classici".

"A schema mapping transforms a source database instance into an instance that obeys a target schema. It has long been one of the most important, yet difficult, problems in the areas of data exchange and data integration.

Traditional database applications in E-business, data warehousing and semantic query processing have required good schema mappings among heterogeneous schemas. Moreover, as the amount of structured Web-based information explodes (e.g., Wikipedia, Freebase, Google BigTable, etc.), users are directly exposed to the task of combining, structuring and repurposing information. Doing so inevitably requires schema mapping to be democratic: non-technical users cook their data with their own flavor, even if they "professional kitchenware" designed for database should be able to cannot master the experts.

End-users increasingly find the need to perform light-weight, customized schema mapping. State-of-the-art tools provide powerful functions to generate schema mappings, but they usually require an in-depth understanding of the semantics of multiple schemas and their correspondences, and are thus not suitable for users who are technically unsophisticated or when a large number of mappings must be performed.

Due to the importance of the schema mapping problem, a handful of mapping design systems have been developed. These systems include InfoSphere Data Architect (from Clio), BizTalk Mapper, Altova MapForce, and Stylus Studio. All of these systems are based on the same general methodology that was first proposed in Clio. The methodology consists of two phases. In the first matching phase, a set of correspondences between source and target schema elements is solicited from the user, with possible aid from automated techniques that find similar attribute pairs. During the second mapping phase, the set of matches yields an executable transformation from the source schema to the target schema.
Unfortunately, this traditional match-driven model is unsuitable for many modern schema mapping tasks. The user must either build attribute-level matches from scratch, or else painstakingly doublecheck an automatically-generated set of matches.

An implicit assumption made by these systems is that the user has detailed knowledge of both the source and target schemas. For traditional schema mapping tasks that involve a sophisticated administrator and a single high-value target database, this assumption makes sense. But modern mapping scenarios feature relatively unsophisticated users and a multiplicity of tasks: a DBA may only map two HR databases together when a corporate acquisition takes place, whereas a Web advertising analyst may need to combine schemas of different datasets multiple times a day."

Le motivazioni sopracitate hanno portato i ricercatori a creare dei sistemi che, interagendo con gli utenti, permettono di utilizzare metodi di schema mapping anche a utenti non esperti.
I due principali sistemi interattivi che ho analizzato sono:

- "Interactive Mapping Specification with Exemplar Tuples" di Bonifati [IMS]
- "Sample-Driven Schema Mapping" di Jagadish [MWeaver]

Ci sono altri sistemi di questo tipo che andranno messi nello stato dell'arte, come quello che Atzeni aveva presentato al SEBD.

Con il numero sempre crescente di sorgenti disponibili per un determinato dominio (film, libri, voli, …), non è pensabile che un utente debba fornire delle Exemplar Tuples e interagire con il sistema per ogni sorgente che si vuole integrare. Bisognerebbe quindi pensare a un sistema che possa lavorare in modo autonomo senza interagire con un utente.

L'idea alla base di questo (futuro) paper sarebbe quella di provare a presentare un sistema che sfruttando Query Reverse Engineering possa produrre i mapping in modo automatico.

Ora presento brevemente i due lavori "interattivi", poi analizzo il funzionamento ad alto livello di QRE ed infine introduco la metodologia (ancora assai acerba) a cui ho pensato.

IMS (Bonifati)

"The idea is to leverage a few exemplar tuples to infer the underlying mappings and iterate the inference process via simple user interactions under the form of boolean queries on the validity of the initial exemplar tuples.
(IMS) Given exemplar tuples as an input pair (E_S , E_T) provided by a non-expert user and a mapping M that the user has in mind, the Interactive Mapping Specification problem is to discover, by means of boolean interactions, a mapping M' such that ((E_S , E_T) satisfy M' and M' generalizes M."

Dalle exemplar tuples vengono estratte delle TGD che poi andranno a formare i mapping. Informalmente quello che viene fatto è sostituire le costanti con delle variabili. I mapping che si ottengono vengono poi detti Canonical Mapping.
Questi mapping però non sono di buona qualità, sono infatti troppo specifici. Il problema è dovuto all'ambiguità delle exemplar tuples. Basti pensare a due sorgenti che forniscono informazioni su voli. "Milano", potrebbe essere la citta di partenza di un volo, o quella di arrivo, oppure potrebbe essere la base di una compagnia aerea, oppure la citta di residenza di un passeggero; i mapping generati a partire da queste sorgenti produrrebbero molti mapping che in realtà sono sbagliati e non necessari.
Per questo motivo è necessario interagire con l'utente, per poter risolvere l'ambiguità delle exemplar tuples.

Un'altra limitazione di questo metodo è il poter derivare mapping solamente tra gli attributi presenti nelle exemplar tuples. Nel caso i Db delle due sorgenti contengano molte tabelle non è realistico chiedere ad un utente non esperto delle exemplar tuples che coprano tutte le relazioni.

MWeaver (Jagadish)

"In this paper, we propose a sample-driven approach that enables relatively unsophisticated end-users, not DBAs, to easily construct their own data. The key idea behind our approach is to allow the user to implicitly specify schema mappings by providing sample data of the target database. Behind the scenes, the system automatically elicits the mappings that transform the source database into this partially-described target. After the user has provided enough information, the system can determine a single best mapping. The process is iterative. As the user types more information from the target database, the system provides increasingly better estimates of the correct mapping."

Anche in questo caso il metodo richiede l'interazione con l'utente. Nel caso in cui i DB da mappare contenga attributi "ambigui", come quelli dell'esempio riguardante le città nel DB sui voli, il sistema prima di giungere a convergenza potrebbe necessitare molti esempi forniti dall'utente.
Il problema è ulteriormente peggiorato nel caso si vogliano mappare DB con molte relazioni ed attributi o molti DB diversi.
Ancora una volta la necessità di un metodo che riesca a lavorare in automatico risulta ben motivata.

Stato dell'arte

Vari metodi sono presenti nella letteratura per risolvere il problema dello schama mapping. Le principali tipologie di sistemi sono le seguenti:

- Metodi che confrontano i nomi degli attributi per determinare i mapping
- Metodi che analizzano i valori degli attributi per determinare i mapping
- Metodi ibridi

Paper da mettere nello stato dell'arte:

- Meta-Mappings for Schema Mapping Reuse [Atzeni]
- Characterizing Schema Mappings via Data Examples / Designing and Refining Schema Mappings via Data Examples [Alexe]
- Schema Mappings and Data Examples [Cate]
- Schema Mapping Discovery from Data Instances [Gottlob]

QRE

"Given a database D and a result table T , which is the output of some known or unknown query Q on D; the goal of QRE is to reverse-engineer a query Q' such that the output of query Q' on database D (denoted by Q'(D)) is equal to T (i.e., Q(D)). We say that two queries Q and Q' are instance-equivalent queries (IEQs) w.r.t. a database D if their results (w.r.t. D) are equal."

L'algoritmo QRE e anche i sistemi di schema mapping che ho analizzato danno per scontato la conoscenza dello schema-graph, ovvero come le relazioni all'intero del DBnsono connesse l'un l'altra (tutti i legami Privare-Key / Foreign-Key). Bisogna controllare che in letteratura ci siano dei metodi per imparare questi legami in modo automatico (non dovrebbe essere troppo complesso viste le caratteristiche che devono avere le chiavi).

L'algoritmo di QRE si compone essenzialmente di 3 passi:

1. Trovare delle Core Relations (le relazioni necessarie per la proiezione e per i join).
2. Trovare tutti i sub-schema-graph che contengono la le core relation (questo serve per generare delle query che sono diverse dalla query originale Q, sia per la parte di selezione sia per le relazioni coinvolte).
3. Determinare la clausola di selezione utilizzando dei decision tree.

L'output dell'algoritmo è la definizione delle 3 parti che compongono una query:

- Select: sono gli attributi della query Q originale (quella che ha generato T). Per la versione dell'algoritmo che non necessita la conoscenza di Q il problema di decidere gli attributi della proiezione e più complesso. Per l'implementazione del nostro metodo necessitiamo di entrambe le versioni di QRE, per questo motivo spiegherò come gestire il secondo casso successivamente (nella nostra metodologia).

- From: sono le relazioni presenti nelle core relation e nei sub-shema-graph.
- Where: vengono decise utilizzando i decision tree.

Il nostro metodo

Ora andiamo a presentare come andremo ad utilizzare QRE nel nostro metodo per schema mapping. Come detto in precedenza il metodo che stiamo ideando dovrà costruire i mapping in modo autonomo, senza bisogno di interagire con un utente.

Nei sistemi interattivi, l'interazione, oltre ad essere necessaria per fornire le exemplar tuples, serve principalmente a risolvere i problemi di ambiguità che scaturiscono dall'uso delle TGD o dal fatto che un esempio possa mappare su schemi diversi. Nel nostro sistema, l'utilizzo di QRE sarà proprio utilizzato per risolvere questi problemi di ambiguità in modo automatico.

Un classico esempio (Esempio 1) di ambiguità è il seguente.
Dato il DB di una sorgente che fornisce informazioni su voli potremmo generare questo esempio:

Mario Rossi	Milano

Quando però andiamo a cercare questo esempio nel DB del target potremmo trovare più di una corrispondenza:

ANAGRAFICA	
Nome	**Residenza**
Mario Rossi	Milano

BIGLIETTO	
Nome	**Partenza**
Mario Rossi	Milano

VOLO	
Pilota	**Destinazione**
Mario Rossi	Milano

Questo è un classico caso di ambiguità che con un sistema come quello delle TGD non è possibile risolvere in modo automatico.
L'idea alla base del nostro metodo è ottenere le IEQs dalla sorgente, ottenere le IEQs da ogni possibile schema del target ed andare a comparare ogni gruppo di IEQs del target con quello della sorgente per andare a capire quale degli schemi del target è davvero il corrispettivo di quello della sorgente.
Si noti, che anche nel caso si trovi una sola corrispondenza dell'esempio T nel target, non possiamo essere sicuri che questa sia corretta (nella sorgente l'esempio T potrebbe essere stato generato da una relazione che fornisce informazioni sull'anagrafica dei passeggeri, mentre nel target l'unica corrispondenza potrebbe essere stata trovata in una relazione che fornisce informazioni sui voli, ed in particolare i piloti di un volo). Quindi anche in questo caso sarà necessario andare a confrontare le IEQs generate dalla sorgente e dal target per capire se il mapping può essere prodotto.
Si potrebbe pensare che non sia necessario applicare QRE sia sulla sorgente sia sul target, ma bensì prendere l'esempio generato dalla sorgente (di cui sappiamo lo schema), ottenere la query del target che ha generato l'esempio ed andare a mappare gli attributi trovati nella clausola select con quelli noti della sorgente. Questo metodo però non risolve i problemi di ambiguità che abbiamo appena visto (siccome T nel target potrebbe mappare su schemi diversi come abbiamo visto nell'Esempio 1). [Sarebbe bello dimostrare in modo formale che QRE applicato solo al target equivale all'utilizzo delle TGD].

Il nostro metodo ci permette anche di derivare più mapping rispetto a quelli che comprendono gli attributi che sono presenti nello schema di T. Nell'esempio (Esempio 2) sottostante, andando ad analizzare le query che hanno generato la tabella d'esempio potremmo trovare:

- select Titolo, Regista

from FILM
where regista=”James Cameron” and anno>1997

- select Titolo, Regista
  from FILM
  where incasso > 2mld

Se andando ad analizzare le query che vengono generate dalla sorgente e dal target a partire dall’esempio sottostante, troviamo che in entrambi i casi sono presenti queste due query, non solo andremo a produrre i mapping che coinvolgono titolo e regista, ma potremo anche produrre dei mapping che riguardano anche anno e incasso.

FILM	
Titolo	**Regista**
Titanic	James Cameron
Avatar	James Cameron

Per poter risolvere i due problemi presentati negli esempi soprastanti (quello sul DB dei voli e quello sul DB dei film) abbiamo bisogno di un metodo per definire la similarità tra due query. Le due query sono state generate dallo stesso T quindi il loro risultato sarà identico. Abbiamo bisogno di un metodo per capire quando due IEQs sono simili concettualmente. Possiamo chiamarlo “Query Matching” questo problema o esiste già?
Una prima idea potrebbe essere quella di utilizzare metodi basati sulla similarità semantica dei nomi degli attributi; credo però, che sia più probabile che i valori degli attributi siano simili piuttosto che i nomi degli attributi siano simili quindi quando andiamo a cercare delle similarità tra le query andremo a comparare i valori degli attributi della clausola where. Se abbiamo un esempio (Esempio 3) che comprende i voli partiti da Los Angels, il nome dell’attributo il cui valore è Los Angel potrebbe essere (partenza, città di partenza, origine, …) ma in tutti i casi il valore probabilmente sarebbe Los Angels.

L’input del nostro metodo sono due DB, quello della sorgente e quello del target. Per iniziare viene richiesto che i due DB comprano informazioni appartenenti allo stesso domino. Siccome andremo a compare le query generate dalla sorgente e dal target per lo stesso esempio, viene anche richiesto che i due DB forniscano, almeno in parte, valori per oggetti comuni. Sarà poi interessante studiare in futuro quale è il minimo overlap tra i due DB affinché sia possibile generare mapping di buona qualità. E, nel caso l’overlap richiesto sia basso, sarebbe possibile applicare il metodo ad un maggior numero di casi.

L’output dell’algoritmo sarà invece una serie di mapping, con annessa la probabilità che siano corretti.

Uno dei possibili use case del nostro metodo è quello del truth finding, quando numerose sorgenti forniscono dati per un gruppo condiviso di oggetti, e da queste sorgenti si vuole generare un DB che copra il più alto numero di oggetti possibile, ed il valore di questi oggetti sia il più possibile accurato. La parte di entity resolution e la parte di data fusion rimangono necessarie, anche se in molti casi gli oggetti hanno di per se un valore univoco (libri -> ISBN, voli -> codice volo), che rende la fase di entity resolution non indispensabile.

Per riassumere QRE porta due benefici:

- permette di risolvere i problemi di ambiguità in modo automatico

- permette di generare più mapping di quelli che si potrebbero generare utilizzando solo le informazioni contenute nell'esempio (T)

L'algoritmo iterativo a cui ho pensato è il seguente (ogni passo è spiegato in seguito):

1. Scelta di T e della query Q che applicata al DB sorgente genera T
2. Esecuzione di QRE sulla sorgente
3. Rank delle IEQs ottenute dalla sorgente
4. Ricerca di tutti le possibili T che hanno una corrispondenza nel target
5. Esecuzione di QRE su ogni possibile schema per cui T ha trovato una corrispondenza nel target
6. Rank delle IEQs di ogni gruppo di IEQs trovate al punto precedente
7. Ricerca del giusto schema che mappa T nel target, andando ad analizzare la clausola where delle IEQs generate (Query Matching) e generazione del giusto mapping
8. Generazione di altri mapping utilizzando le query per le quali abbiamo ottenuto un match al passo precedente
9. Analisi dei mapping
10. Generazione di una nuova T, che aiuti ad ottenere dei mapping migliori o dei nuovi mapping. Si torna al punto 1

1 - Scelta di T e della query Q che applicata al DB sorgente genera T

Utilizzare esempi piccoli permette a QRE di trovare la clausola where in un tempo molto minore, poiché gli alberi binari che vengono creati saranno meno profondi. Utilizzare esempi piccoli permette anche di ottimizzare la fase di ricerca dell'esempio nel DB target. Infine utilizzare esempi piccoli aumenta la probabilità di trovare una corrispondenza dell'esempio T nel DB target, sia perché il DB target potrebbe fornire valori per un numero minore di oggetti, o per oggetti diversi, sia per il fatto che il DB target potrebbe fornire meno informazioni per ogni oggetto (meno attributi nelle relazioni), o fornire informazioni diverse per gli oggetti (altri attributi).
L'esempio non deve però essere allo stesso tempo troppo piccolo, poiché questo potrebbe causare un numero eccessivo di corrispondenze nel DB target, e quindi troppi possibili mapping tra cui scegliere quello giusto.

Le IEQs ricavate dall'algoritmo QRE possono essere di due tipi: strong e weak. Questo è dovuto al fatto che quando eseguiamo la proiezione, dopo che la clausola where ha filtrato i record, nel caso siano presenti record ripetuti questi vengono eliminati. Se sappiamo la query Q che ha generato T possiamo suddividere le IEQs in due gruppi, strong nel caso i record ritornati siano esattamente quelli prodotti dalla query originale Q che ha generato T, weak nel caso contrario.
Per essere sicuri di identificare mapping che sono significativi è opportuno lavorare con strong IEQs. Classificare una IEQs come weak/strong è però possibile solo sul DB sorgente, infatti, non è possibile sapere se la query Q originale applicata al DB target dia lo stesso risultato T.
Per questo motivo quando selezioniamo T, e quindi quando selezioniamo la query Q che genera T, dovremmo andare a selezionare almeno un attributo che permetta di identificare in modo univoco i record di T (se possibile), non possiamo però prendere la chiave, o per lo meno dobbiamo stare attenti nel farlo: spesso la chiave è un ID che non ha nessun significato semantico all'esterno del DB, e che quindi non troverà nessuna corrispondenza nel DB target, bisognerà invece selezionare un attributo che si comporta quasi come una chiave ma che in realtà possiede anche un significato all'esterno del DB. Alcuni esempi potrebbero essere: codice fiscale, ISBN, codice di un volo, titolo di un film, titolo di un libro, nome+cognome di persone.

Una possibile prima idea per selezionare T potrebbe quindi essere prendere un piccolo esempio formato da alcuni record e da due attributi, uno dei quali è quello "chiave" mentre l'altro è quello del quale si vuole trovare il mapping. Per quanto riguarda la scelta dei record da inserire in T l'idea è quella di scegliere un valore per l'attributo che non è chiave e poi scegliere un sottoinsieme dei record che hanno quel dato valore per l'attributo non chiave. Spieghiamo meglio questo concetto con un esempio: come colonna "chiave", nel caso stiamo utilizzando la relazione "Film" di un DB che fornisce informazioni su film, potremmo utilizzare il titolo del film, il secondo attributo che includiamo in T potrebbe essere "regista". In questo caso, per selezionare i record da inserire in T, scegliamo un regista (James Cameron), e poi di tutti i film diretti da James Cameron ne scegliamo una parte (Titanic, Avatar). Questa procedura permette all'algoritmo di QRE di andare a creare delle IEQs significative, che non siano semplicemente "select titolo, regista from File, where regista=James Cameron", ma IEQs che utilizzano anche gli altri attributi della tabella film (ad esempio "anno>1997") o di altre tabelle (ad esempio "ricavi>2mld", nel caso questa informazione sia in una relazione diversa da "Film").

2 - Esecuzione di QRE sulla sorgente

L'algoritmo QRE viene applicato alla sorgente per ricavare le IEQs di Q.

3 - Rank delle IEQs ottenute dalla sorgente

Molti componenti del nostro sistema hanno una complessità computazione alta. Bisognerà, ogni volta possibile, provare ad ottimizzare gli algoritmi che utilizziamo.
Specialmente quando gli esempi che utilizziamo (T) non sono molto grandi, sia per il numero di righe sia per il numero di colonne, le IEQs che generiamo potrebbero essere in numero considerevole.
Il rank delle query è utile per due motivi:

- Non tutte le IEQs generate andranno ad essere analizzate e confrontate con quelle della sorgente. Questo permette di risparmiare tempo.
- Sceglie le IEQs che sono più significative, e quindi possono essere utili per risolvere il problema dell'ambiguità.

Alcuni metodi sono stati presentati nella letteratura di query reverse engineering.
"As a query generally has many possible IEQs, it is useful to rank and prioritize the presentation of the derived IEQs to the user. Since our preference for simpler and more precise IEQs involves conflicting objectives, we present three reasonable criteria for ranking IEQs: a metric based on the minimum description length (MDL) principle and two metrics based on the F-measure."
Query corte sono senza dubbio più semplici da confrontare.

4 - Ricerca di tutti le possibili T che hanno una corrispondenza nel target (sample search)

Come abbiamo visto nell'Esempio 1, T può mappare su diversi schemi nel target. Per questo motivo dobbiamo cercare tutte le possibili corrispondenze di T nel target.

QRE include una versione che non prevede che la query Q che ha generato T venga fornita. In questo caso non è possibile conoscere in anticipo lo schema di T. Per questo motivo l'algoritmo deve scoprire tutti i possibili schemi di T nel DB, andando a cercare tutte le occorrenze di T nel DB. Il modo in cui questo viene fatto è andando a vedere una colonna alla volta di T con quali colonne del DB è compatibile, viene poi generata ogni combinazione di possibili colonne come schema di T. Questo metodo ha una complessità computazionale molto alta.

Jagadish, nel paper che ho presentato brevemente prima, ha studiato questo problema (che è NP-Hard) e ha creato un metodo molto più ottimizzato. Potremmo usare quello, dopo averlo studiato in dettaglio.

5 - Esecuzione di QRE su ogni possibile schema per cui T ha trovato una corrispondenza nel target

Dopo aver eseguito il punto 4 abbiamo ora uno o più schemi per l'esempio T.
Per ogni schema, utilizzando QRE, calcoliamo le IEQs che lo generano. Alla fine, avremo quindi dei gruppi di IEQs, uno per ogni schema che abbiamo trovato per T.

6 - Rank delle IEQs di ogni gruppo di IEQs trovate al punto precedente

Per ogni gruppo di IEQs trovate al punto precedente calcolarne il ranking.
Possiamo usare gli stessi algoritmi che useremo al punto 3, ma questa volta potremmo creare un algoritmo personalizzato che tenga presente gli stessi principi alla base dei metodi del punto 3 ma che allo stesso tempo utilizzi delle informazioni supplementari. Sempre allo scopo di facilitare la fase di "query matching" potremmo, estrarre delle informazioni dalle top-ranking query della sorgente (ad esempio i valori degli attributi della clausola where), ed utilizzare queste informazioni per incrementare la posizione delle IEQs del target che possiedono queste caratteristiche.

7 - Ricerca del giusto schema che mappa T nel target, andando ad analizzare la clausola where delle IEQs generate (Query Matching) e generazione del giusto mapping

La parte di "Query Matching" è il componente che viene utilizzato per risolvere i problemi di ambiguità.
Durante questa fase confrontiamo le top IEQs di ogni gruppo che abbiamo identificato nel target con le IEQs che abbiamo ottenuto dalla sorgente. Il gruppo di IEQs del target che ottiene più matching (e allo stesso tempo ne ottiene un numero sufficiente) con le IEQs della sorgente sarà scelto come mapping corretto. Notare, come già introdotto in precedenza, che anche nel caso sia stata trovata una sola corrispondenza di T nel target, è comunque necessario andare a confrontare le IEQs generate dalla sorgente e dal target al fine di validare il mapping (Si veda l'esempio 1).

Durante la fase di "Query Matching", come abbiamo già visto in precedenza, andiamo a confrontare i valori degli attributi della clausola where e non gli attributi stessi. Questa scelta assume che i valori siano confrontabili tra sorgente e target; questa assunzione è realistica come abbiamo potuto vedere nell'Esempio 3. Questa assunzione viene fatta anche da tutti i lavori precedenti che utilizzano sample per effettuare schema mapping (come il paper di Jagadish).

Supponiamo per esempio che T sia stato estratto da una relazione del DB sorgente che conteneva informazioni riguardanti l'anagrafica di registi e attori, e sia:

James Cameron	Londra
Steven Spielberg	Londra

E che nel DB target siano stare trovate due corrispondenze (i valori dell'esempio non sono reali):

FILM					
Titolo	**Regista**	**Anno**	**Location**	**Incasso**	**Produzione**
Titanic	James Cameron	1997	Londra	2mld	20th century fox
Schindler's List	Steven Spielberg	1993	Londra	1,5mld	20th century fox

e

REGISTI

Nome	Luogo di nascita	Anno di nascita
James Cameron	Londra	1954
Steven Spielberg	Londra	1946

e che le query che sono stato generate dalle due corrispondenze trovate nel target siano:

- Film:
  - … where produzione=”20th century fox” and anno>1992 and anno<1998
  - … where incasso>=1,5mld
- Registi:
  - … where location=”Londra” and anno>1945 and anno<1955

Andando a confrontare questi gruppi di IEQs (confrontando i valori nella clausola where) con il gruppo di IEQs ottenuto sulla sorgente possiamo ottenere il mapping corretto.
[Rimane un problema da risolvere, al quale non ho ancora trovato una soluzione “comoda”: nella tabella “Film” potrebbero esserci più film di questi 2 registi ambientati a Londra. Questo problema nel paper QRE è stato risolto con il concetto di at-least-one-semantic ma non ho ancora ben capito come integrarlo nella nostra metodologia].

8 - Generazione di altri mapping utilizzando le query per le quali abbiamo ottenuto un match al passo precedente

Una volta trovato il giusto schema nel target per l’esempio T possiamo utilizzare le IEQs tra le quali c’è stato un match per generare altri mapping. Si noti che per aumentare la probabilità che i mapping generati siano corretti, si andranno a generare nuovi mapping solamente a partire dal gruppo di IEQs del target per il quale al punto 7 si era ottenuto il mapping con la sorgente.
Si veda l’Esempio 2 per maggior chiarezza.

9 - Analisi dei mapping

In questa fase assegniamo ai mapping trovati al punto 7 ed 8 una probabilità di essere corretti. Li confrontiamo anche con i mapping che sono stati trovati in precedenza per poter meglio stimare la loro correttezza.
Dobbiamo stare attenti perché il mapping non è una funzione che mappa un attributo della sorgente su un attributo del target, potrebbero infatti esserci alcuni attributi che mappano su un attributo e viceversa. Si veda l’esempio sottostante:

DB1:

ATTORE	
ID	**Nome**

REGISTA	
ID	**Nome**

DB2:

PERSONA		
ID	**Nome**	**Ruolo**

In questi casi bisognerà pensare ad un modo per ottenere il giusto mapping, ad esempio andando ad analizzare il valore dell’attributo “ruolo” (operazione da fare in automatico, per esempio andando a contare i mapping che ci sono per “Attore->Persona” e “Regista->Persona” e vedendo se c’è una corrispondenza con la frequenza di “attore” e “regista” nell’attributo “Ruolo”).

10 - Generazione di una nuova T, che aiuti ad ottenere dei mapping migliori o dei nuovi mapping. Si torna al punto 1

Il metodo presentato è un metodo iterativo. Giunti a questo punto bisogna quindi tornare al punto 1. Ora però abbiamo più informazioni su come scegliere T. Sceglieremo un nuovo esempio che ci permetterà di generare un mapping che prima era sconosciuto o di affinare un mapping la cui probabilità di essere corretto non era sufficiente.

Altre considerazioni (sparse e disordinate)

- Sono tutti ragionamenti preliminari, quando inizieremo a fare i primi esperimenti sicuramente molte cose andranno modificare, e tante altre cose a cui non avevamo pensato andranno prese in considerazione.
- Al posto che comparare le query generate da QRE si potrebbe provare a comparare gli alberi decisionali che hanno generato le query.
- Circa una decina di metodi per query reverse engineering sono stati presentati, QRE è solo uno di questi. Dovremmo studiare anche gli altri per capire se hanno qualche caratteristica che può essere utilizzata nella nostra metodologia. Di questi metodi alcuni sono esatti (come QRE) mentre altri sono approssimati, sarebbe interessante scoprire, come l'utilizzo di metodi approssimati, possa migliorare le performance del nostro sistema.
- Non solo nel target DB, T potrebbe mappare su diversi schemi, lo stesso potrebbe succedere anche nel DB sorgente se ignoriamo la query Q che ha generato T. Potremmo provare a collegare più schemi allo stesso tempo (ammesso che in entrambe le sorgenti troviamo lo stesso numero di schemi per T).
- Per quanto riguarda gli esperimenti dobbiamo trovare dei DB forniti da alcune sorgenti che forniscono informazioni per un dominio condiviso. Domini sui quali sperimentare potrebbero essere:
    - Libri (come nel caso degli esperimenti del paper su source authority, potremmo cercare DB di vari negozi di libri)
    - Film (la ragazza di Hong Kong ci aveva dato un dataset contnentente informazioni su film, i dati erano forniti da 15 sorgenti diverse: imdb, allmovie, amazon, goodfilms, top250tv, agoodmovietowatch, …)
    - Stock market (innumerevoli sono i siti che aggregano informazioni riguardanti il mercato azionario, come ad esempio Bloomberg e investing.com, ci sono poi anche tutti i siti ufficiali come NYSE, Nasdaq, DowJones, Borsa Italiana, …) in questo caso i siti di aggregatori sono necessari per allineare gli schemi di sorgenti che non condividono stock come ad esempio NYSE e Borsa Italiana.
    - Voli aerei (tutti i siti di aggregatori come skyscanner e kayak, e poi tutti i siti delle compagnie aere, e degli aeroporti), anche in questo caso gli aggregatori o gli aeroporti sono necessari per mappare tra di loro i siti di diverse compagnie aere.

  QRE ha bisogno di un DB e non di un Dataset in cui tutte le informazioni sono in un'unica relazione, nel caso in cui non troveremo dei DB bisognerà quindi normalizzare.