Un approccio cognitivo per il reverse engineering di basi di dati relazionali

Oreste Signore - Mario Loffredo - Mauro Gregori - Marco Cima

SEAL (Software Engineering and Applications Laboratory)
CNUCE - Institute of CNR - via S. Maria, 36 - 56126 Pisa (Italy)
Phone: +39 (50) 593201 - FAX: +39 (50) 904052
E.mail: oreste@vm.cnuce.cnr.it

Abstract - La ricostruzione dello schema concettuale delle basi di dati a partire dallo schema fisico costituisce un problema rilevante nella reingegnerizzazione delle applicazioni e nella identificazione delle scelte di progettazione. Nel presente lavoro vengono descritti gli effetti delle scelte operate più di frequente nella fase di Forward Engineering, e le conseguenti difficoltà che possono essere incontrate in un processo di Reverse Engineering. Viene successivamente illustrato un approccio al reverse engineering di basi di dati relazionali basato sulla identificazione di indicatori di schema, di chiave primaria, di codice SQL e procedurali, che portano alla definizione di fatti Prolog e, mediante regole euristiche, alla ricostruzione dello schema concettuale.

1. Introduzione

La quantità di risorse dedicate alla manutenzione delle applicazioni, e gli investimenti effettuati dalle imprese nella realizzazione di sistemi applicativi ormai obsoleti, o non in grado di utilizzare al meglio le nuove e sempre più ricche potenzialità offerte dai prodotti disponibili sul mercato, rendono particolarmente importante il recupero del patrimonio software e il riutilizzo non solo del software, ma soprattutto degli aspetti significativi della progettazione. Il *Reverse Engineering* (RE) è un processo di analisi di un sistema allo scopo di identificarne i componenti e le loro interrelazioni e creare delle rappresentazioni in un' altra forma o ad un più alto livello di astrazione. Un processo di RE in generale comporta l' estrazione di elementi di disegno o la sintesi di astrazioni che sono meno dipendenti dall' implementazione, ma può partire da qualsiasi livello di astrazione ([Chikofsky90]).
In questo quadro grande interesse riveste l' allestimento di un processo di RE di basi di dati dal livello fisico al livello concettuale, specialmente nel caso in cui tale processo è applicato ad organizzazioni di dati superate dal punto di visto tecnologico e mancanti di una documentazione aggiornata.

2. Il processo di FE

Lo sviluppo di un processo generale di RE richiede una profonda comprensione del processo in avanti o *Forward Engineering* (FE), soprattutto di tutte quelle tecniche che non rientrano in metodologie ben formalizzate e standard ma scaturiscono dall' intuizione umana.
La progettazione complessiva di una base di dati consiste di una serie di processi classificati nelle tre tipiche fasi consecutive: concettuale, logica e fisica.
La fase *concettuale* consiste nel raccogliere ed analizzare i requisiti espressi dall' utente, e formalizzarli nello schema concettuale. Non ha nessun impatto sul processo di RE. Nella fase

logica tutte le informazioni contenute nello schema che non sono direttamente traducibili nel DMS[1] utilizzato vengono trasformate opportunamente (es. le relazioni n-rie, le relazioni N:M o con attributi). La fase *fisica* consiste di una prima fase di traduzione nel DDL (*Data Definition Language*) durante la quale alcuni vincoli impliciti del modello concettuale non DMS-conformi (es. la cardinalità e la totalità o parzialità delle associazioni, l' integrità referenziale) vengono espressi mediante vincoli procedurali, implementati nel linguaggio ospite o mediante costrutti opportuni forniti dal DMS (trigger). Successivamente la base di dati viene messa a punto in accordo ai requisiti fisici imposti dall' utente e alle caratteristiche dell' ambiente ospite (performance, occupazione di memoria, etc.). In questa fase si procede alla definizione delle viste utente e anche, a volte, a una denormalizzazione delle relazioni.
In conclusione, la produzione di uno schema di base di dati eseguibile può essere vista come l' applicazione sequenziale, a partire dallo schema concettuale, di una serie di trasformazioni che introducono ad ogni passo determinati tipi di *degradazione dello schema*, rendendolo via via meno completo, semplice, leggibile ed intuitivo.
Ogni metodologia di sviluppo di basi di dati relazionali deve affrontare, quindi, il problema del divario di espressività semantica fra il modello di partenza (in genere un ER o ER esteso) ed il modello di arrivo, che, come è noto, non permette la rappresentazione diretta di alcuni concetti fondamentali del modello ER, come generalizzazione, aggregazione e attributi multivalore ([Teorey86]).
Di conseguenza, lo schema fisico risulta spesso molto distante da quello concettuale, per cui non è immediato catturarne la semantica.

3. Il processo di RE

3.1 Generalità

Le esperienze di DBRE riportate in letteratura o concretizzate in tool sono spesso limitate ad un unico modello di dati ed in genere sono viziate da numerose ipotesi restrittive di scarsa applicabilità nella pratica:

- tutte le specifiche concettuali sono state tradotte in strutture dati e vincoli;
- la traduzione dallo schema concettuale allo schema logico è stata attuata in modo semplice e diretto, senza utilizzare alcuna astuzia di rappresentazione;
- lo schema non è stato sottoposto ad alcuna fase di ristrutturazione per l' adeguamento ai requisiti aggiuntivi imposti dall' utente o dall' ambiente ospite;
- la scelta dei nomi degli elementi dello schema è stata effettuata con criteri razionali.

[1] Data Management System, ovvero un sistema per la gestione dei dati tra i molti utilizzabili: File Processing System, DBMS, etc.

Invece, è abbastanza realistico pensare che un processo di RDBRE (Relational DBRE) debba essere in grado di analizzare tipologie di schemi appartenenti ad un' ampia casistica di soluzioni implementative, dalle più prevedibili alle più insolite.
Inoltre, è necessario prevedere l' analisi di schemi realizzati in DBMS relazionali non recenti, quindi con limitazioni del DDL come, ad esempio, l' assenza di costrutti e supporti per la definizione esplicita delle chiavi e dei vincoli di integrità referenziale. In definitiva, sarebbe auspicabile lo sviluppo di una teoria generale che supporti il DBRE di applicazioni reali.
L' applicazione delle metodologie di FE, in generale, produce due insiemi di specifiche:

- un insieme di relazioni contenenti l' informazione espressa concettualmente nella specifica del modello ER, normalizzate al grado desiderato;
- un insieme di procedure per l' implementazione dei vincoli impliciti sul modello concettuale.

Dunque, da queste due fonti di informazione deve partire un generico processo di DBRE che si prefigga di estrarre ed astrarre tutta la conoscenza necessaria a ricostruire uno schema concettuale corrispondente all' implementazione fisica.

3.2 Alcune proposte

Batini, Ceri e Navathe ([Batini92]) propongono un processo di RE su basi di dati relazionali estremamente semplice e limitato, che rappresenta comunque un buon modello iniziale. In esso sono esemplificati in modo chiaro e lineare i passi di analisi delle relazioni e di riconoscimento dei concetti, da applicare nei casi in cui si possieda una buona conoscenza semantica dello schema relazionale di partenza. Inoltre gli autori trattano in maniera diversificata approcci ed esempi di RE con riferimento ad altri modelli di dati.
Il secondo approccio di RDBRE, proposto in [Premerlani93], è caratterizzato da un' impostazione essenzialmente sperimentale; in esso, tramite un articolato insieme di metodi, tecniche ed esempi pratici, viene affrontata un' ampia casistica di situazioni reali di partenza.
Hainaut et al. ([Hainaut93]) espongono una breve panoramica introduttiva sullo stato attuale del DBRE e di applicazioni data-oriented. In tale lavoro sono sintetizzate alcune problematiche relative alle difficoltà inerenti al processo di DBRE, indipendenti dai modelli dei dati utilizzati, con l' intento di puntualizzare gli aspetti su cui sarebbe opportuno concentrare gli sforzi di ricerca. Il problema generale del DBRE può essere espresso, secondo Hainaut, come la ricerca di un possibile schema concettuale in grado di condurre ad una organizzazione fisica data dall' insieme delle strutture dei dati nel DDL e nel linguaggio ospite e dai requisiti operazionali.
Il processo risolutivo può essere suddiviso in due fasi generalmente sequenziali: *estrazione della struttura dei dati* (DSE), inversa della fase fisica, e *concettualizzazione della struttura dei dati* (DSC), inversa della fase logica.

[Chiang94] propone una metodologia di ricostruzione dello schema ER che utilizza le informazioni desumibili dal catalogo e dai dati memorizzati nelle relazioni.

4. L' approccio proposto

4.1 Descrizione generale ed aspetti innovativi

Il processo di RDBRE proposto, il cui principale elemento di novità è costituito essenzialmente dal particolare approccio "cognitivo" adottato, si prefigge di ricostruire le specifiche di una base di dati al livello di astrazione tipico della fase di analisi, individuando gli elementi concettuali che ne costituiscono lo schema ER.

Il riconoscimento manuale, umano, del ruolo giocato da ogni elemento di uno schema relazionale è fortemente indirizzato dai molteplici appigli semantici offerti dai nomi stessi degli elementi, come ad esempio dalle eventuali assonanze esistenti tra i nomi delle relazioni e quelli delle loro chiavi. Senza contare, poi, la capacità di sfruttare informazioni provenienti dalle fonti più varie: dalla documentazione stessa delle applicazioni (se esistente), fino ad altre forme di conoscenza spesso difficilmente formalizzabili (ad esempio la conoscenza delle "tradizioni" di *naming* di una certa realtà aziendale).

Un approccio prevalentemente automatico, a meno di non ricorrere a tecniche tipiche dell' AI (sistemi esperti), deve necessariamente lavorare su indizi strutturali ad un livello di astrazione molto più basso (domini, struttura delle tuple e delle chiavi di una relazione, etc.).

Alcune metodologie utilizzano tecniche di verifica delle ipotesi concettuali basate sull' analisi dei dati del *database* stesso. Il risultato di tali tecniche, che per altro è semidecidibile[2], è fortemente limitato dalla quantità e dalla qualità dei dati contenuti nelle istanze di database considerate a certi istanti: ad esempio una relazione in fase di inizializzazione, in generale, fornirà scarse indicazioni.

Il metodo proposto, partendo dallo schema delle relazioni della base di dati, tenta di interpretare l' uso che le applicazioni fanno dei dati in essa contenuti, individuando ed analizzando alcune procedure di interrogazione e di manipolazione (codice SQL/linguaggio ospite) considerate significative per il rilevamento delle proprietà cercate.

Il campo di applicazione favorevole a questo tipo di indagine è quello delle cosiddette applicazioni orientate ai dati, cioè applicazioni "in cui l' aspetto prevalente è costituito dai dati che vengono memorizzati, ricercati, modificati e non tanto dalle funzioni di manipolazione degli stessi, che sono abbastanza semplici e relativamente standard" ([Ghezzi91]).

---

[2] Ad esempio, si desideri verificare la validità di chiave candidata dell'attributo A per la relazione T tramite l'analisi dei dati. Se viene rilevata la presenza di due tuple nella relazione aventi lo stesso valore per A allora ciò indica inequivocabilmente la non univocità dell'attributo, altrimenti nulla può essere asserito.

4.2 Ipotesi iniziali e problematiche

La metodologia di RE descritta in questo paragrafo delinea un processo di *ricostruzione dello schema concettuale*, espresso nel *modello ER,* di *basi di dati relazionali*.
Le fonti informative delle attività automatizzate di esplorazione e di deduzione sono costituite essenzialmente dalle specifiche degli *schemi delle relazioni* e dal *codice dei programmi* che utilizzano la base di dati.
Ogni DBMS mette a disposizione un insieme di meccanismi per la definizione degli schemi delle relazioni, non sempre corrispondenti alle caratteristiche dello standard attuale e questo vale soprattutto per le basi di dati meno recenti, che sono anche le più soggette ad essere re-ingegnerizzate. Inoltre l' espressività di una definizione di schema dipende anche dalle scelte operate dagli implementatori nello sfruttare tutti i meccanismi messi a disposizione dal particolare ambiente di sviluppo. Perciò non si è voluto porre particolari limitazioni al tipo di DDL-SQL utilizzato per la definizione dei dati. I concetti e le convenzioni adottati in riferimento alla definizione degli schemi delle relazioni si riferiscono genericamente agli standard ANSI e ISO, ed alle estensioni più comuni ([Albano91], [Date87], [Korth91], [ISO9075])**.** Le informazioni relative agli schemi delle relazioni sono contenute nel catalogo del sistema, la cui struttura non differisce in modo sostanziale tra i vari prodotti.
Dal catalogo si suppone con certezza di poter estrarre i nomi delle relazioni, i nomi degli attributi di ogni relazione e, per ogni attributo, il tipo e l' eventuale opzione `NOT NULL` (attributo obbligatorio). Inoltre per ogni relazione è possibile rilevare la presenza di insiemi di attributi definiti *indici* della relazione, nonché l' eventuale *opzione di unicità* degli indici (`UNIQUE`). Infine non si esclude l' eventualità di operare su schemi definiti con DDL che adottano clausole per la specifica esplicita di chiavi primarie, chiavi candidate e chiavi esterne e relazioni referenziate. Per quanto riguarda i programmi, essi potranno essere codificati in un qualsiasi linguaggio procedurale ospite che integrerà alcuni comandi SQL (*embedded SQL*) per l' accesso e la manipolazione dei dati del database.
Chiunque si ponga il problema di definire una metodologia di DBRE dovrebbe tener conto di almeno tre aspetti fondamentali: le teorie dei modelli dei dati, la metodologia del FE, la presenza di tecniche di implementazione non standard.
I primi due aspetti non pongono particolari problemi in quanto i modelli trattati (il relazionale e l' ER) sono ben assestati, basati su definizioni univoche e relativamente semplici e le metodologie di FE, pur presentando alcune differenze, hanno alla base un insieme di metodi, tecniche ed indicazioni sufficientemente standardizzate e con una complessità conteuta.
L' ultimo aspetto risulta veramente difficile da gestire. La conoscenza di una vasta casistica di queste tecniche, l' abilità nel riconoscere e discriminare astute ottimizzazioni, scelte anomale o imposte da requisiti dell' applicazione o da limiti dell' ambiente di progettazione, richiedono una consolidata esperienza nello sviluppo di basi di dati.

Un breve esempio può chiarire come questi aspetti possano manifestarsi ed introdurre degradazione dello schema compromettendo il riconoscimento automatico degli elementi concettuali. Il concetto di identificatore degli elementi di una classe (insieme di entità) a livello concettuale è ben definito e non ambiguo; la traduzione nel modello logico relazionale in questo caso è unica, e consiste nella definizione di una chiave primaria della relazione corrispondente alla classe. Ma nel momento in cui ci si prefigge lo scopo di definire un metodo per il riconoscimento della chiave primaria di una generica relazione, occorre tener conto della molteplicità delle forme implementative.

Ad esempio, possono verificarsi una o più delle seguenti situazioni:

- il DDL non supporta meccanismi di definizione esplicita delle chiavi;
- non è definito un *indice univoco* sugli attributi della chiave;
- confidando sulla correttezza delle procedure di manipolazione delle tuple della relazione, l' implementatore non ha specificato a livello di definizione di schema l' obbligatorietà delle componenti della chiave (vincolo `NOT NULL` o `NULL NOT ALLOWED`);
- per ottimizzare lo spazio, la chiave primaria della relazione è definita con una funzione di codifica dei valori dell' identificatore stabilito a livello concettuale;
- viene assunto come meccanismo di individuazione delle tuple un valore di *timestamp*.

4.3 Linee guida

Dalla consapevolezza di queste problematiche sono maturate alcune *linee guida*, elencate nel seguito, per la definizione della metodologia di RDBRE presentata in questo lavoro.

1. Formulazione di un sistema di euristiche (o regole) basato sul bagaglio teorico, le conoscenze pratiche ed alcune supposizioni realistiche sulla natura e sul possibile utilizzo di ogni elemento dello schema relazionale.

   L' applicazione delle regole determina, a seconda del caso:

   - l' individuazione di proprietà particolari degli elementi in analisi, dette indicatori. Definiamo indicatore un insieme di informazioni rilevabili da una o più fonti disponibili (catalogo, codice SQL/ospite, risultati di una fase di analisi precedente), costituente un indizio ritenuto significativo per la caratterizzazione, nel modello concettuale, di uno o più elementi dello schema relazionale;
   - l' identificazione certa, se gli indizi lo consentono, della funzione di elementi dello schema relazionale (chiavi esterne, attributi multivalore, tabelle associative, etc.) o di elementi concettuali (entità, gerarchie, associazioni, etc.);
   - la formulazione di ipotesi eventualmente da confermare o da respingere sulla base di investigazioni successive o su richiesta esplicita dell' utente.
2. Il processo non può essere completamente automatico, ma deve prevedere fasi di interazione con l' utente il quale, sulla base dell' informazione fornita dagli indicatori rilevati, dell' eventuale documentazione e della conoscenza semantica dell' applicazione,

può convalidare le ipotesi proposte o introdurre asserzioni alternative. Al fine di facilitarne il reperimento nel codice dell' applicazione e la successiva analisi "manuale", dovrebbe essere fornita all' utente, come informazione aggiuntiva, l' indicazione "topologica" (nome del modulo, nome della procedura, linea di codice) dei costrutti di programmazione conformi strutturalmente a *patterns* generici predefiniti. È evidente che l' importanza del contributo portato dall' utente dipende fortemente dalla sua esperienza.

3. Sia il risultato dell' applicazione delle regole di riconoscimento, sia le decisioni prese manualmente dall' utente, debbono essere memorizzate in forma di *asserzioni* (*predicati*) andando a popolare la base di conoscenza disponibile per la fase successiva.

4.3.1. Gli indicatori

Gli indicatori, utilizzati in tutto il processo di RE, possono essere suddivisi in quattro categorie, in funzione della fonte informativa da cui possono essere dedotti:

- *indicatori di schema*: estratti dal catalogo e dalle informazioni dedotte nella fase di identificazione delle chiavi, forniscono informazioni su particolari strutture dello schema di una relazione;
- *indicatori di chiave:* ricavati dall' analisi delle chiavi primarie, servono a definire proprietà della chiave primaria di una data relazione;
- *indicatori SQL*: estratti dall' analisi dei comandi SQL, forniscono indicazioni sulle modalità con cui le relazioni vengono interrogate e manipolate;
- *indicatori procedurali*: estratti dall' analisi del codice del linguaggio ospite, integrano l' informazione contenuta negli indicatori SQL, individuando variabili e strutture di controllo tipiche (*patterns*), proprie del linguaggio ospite, con cui in genere si effettuano manipolazioni condizionate della base di dati (cicli di prelievo delle tuple selezionate, controllo dei vincoli di integrità referenziale, operazioni su tabelle che implementano gerarchie, etc.).

Nella tabella in appendice è riportato l' elenco degli indicatori e la loro funzione, che verrà spiegata in seguito.

4.4. Strumenti

Le caratteristiche generali del processo delineato nel paragrafo precedente hanno suggerito l' adozione di un *linguaggio di programmazione logica* come il Prolog per l' implementazione di un prototipo. Infatti in tale ambiente risulta naturale e relativamente semplice:

- modellare la conoscenza acquisita ad ogni fase del processo con un insieme di *fatti* da inserire in un database Prolog;
- trasformare le euristiche stabilite per il riconoscimento di proprietà significative degli elementi dello schema in clausole di Horn e quindi in *goals* Prolog;
- sfruttare il meccanismo del *backtracking* per tentare di risoddisfare un insieme di obiettivi in seguito alle modifiche apportate manualmente dall' utente sulla base di conoscenza.

Inoltre, è importante sottolineare che la proposta di RDBRE qui illustrata deve essere intesa come un nucleo coerente e facilmente adattabile sia per assoggettarsi alle caratteristiche del DDL e dei costrutti di programmazione SQL/procedurali di volta in volta analizzati, sia per accettare possibili ulteriori raffinamenti delle potenzialità di riconoscimento. Questi requisiti possono essere agevolmente soddisfatti considerando le caratteristiche del Prolog che ne fanno un buon strumento di *rapid prototyping* ([Ceri86], [Canfora93]) nel quale nuove assunzioni e regole possono essere progressivamente aggiunte ad un programma funzionante e rese eseguibili immediatamente.
Va inoltre tenuto presente che, essendo essenziale per l' approccio di DBRE proposto la possibilità di operare il riconoscimento di specifici costrutti SQL e procedurali, si rende necessaria una fase di analisi statica del codice dell' applicazione. Si presuppone quindi la possibilità di attingere a tutte le informazioni necessarie relative al codice, importate e disponibili nel formato più adatto ai requisiti stabiliti nella fase di implementazione del prototipo ([Signore94]).

4.5. Il processo di RDBRE

Il processo di reverse si articola essenzialmente in tre fasi (Fig.1):

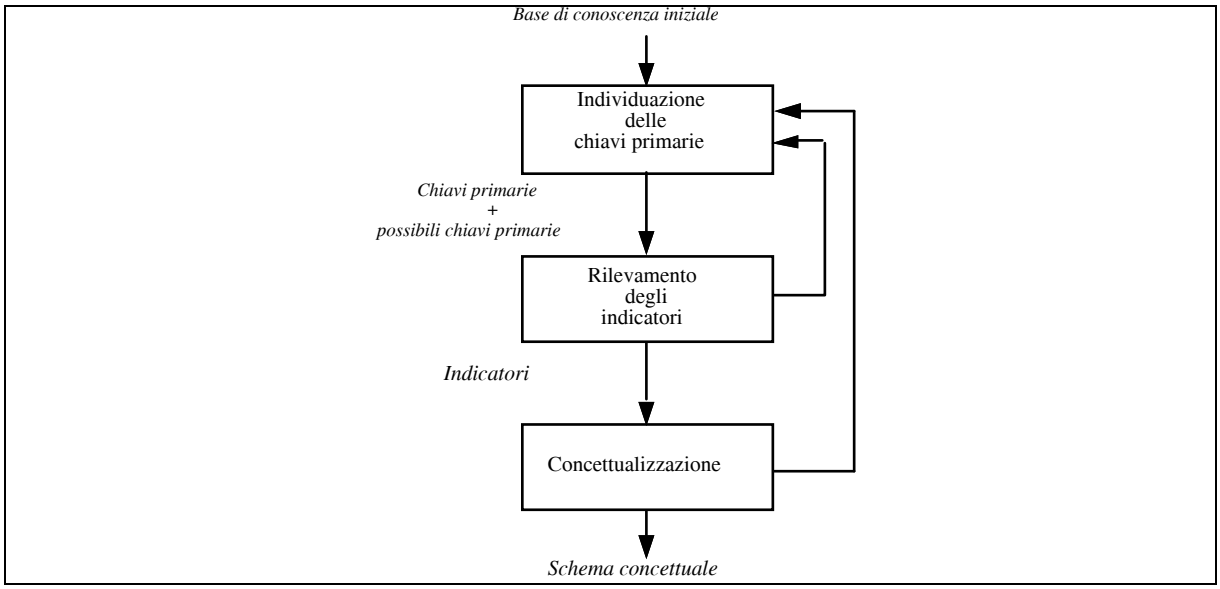

Fig.1 - Le fasi del processo di reverse.

1. *Individuazione delle chiavi primarie*

   Vengono utilizzate tutte le indicazioni estraibili dagli schemi delle relazioni. Se da esse risulta impossibile identificare le chiavi, si analizzano le richieste SQL e la presenza di cicli di prelievo dei dati definiti nel linguaggio ospite, per escludere dall' insieme delle potenziali chiavi di una relazione gli insiemi di attributi che non identificano univocamente singole tuple della relazione. Infine, se necessario, si invoca l' intervento

umano per una identificazione delle chiavi primarie sulla base degli indizi individuati automaticamente.

2. *Rilevamento degli indicatori*

   Vengono ricercati alcuni *indicatori* predefiniti costituenti indizi significativi per la caratterizzazione, nel modello concettuale, di uno o più elementi dello schema relazionale. In particolare in questa fase si tenta di individuare le chiavi esterne che definiscono le associazioni tra le relazioni, le proprietà delle associazioni, la composizione delle chiavi primarie, indizi sulla natura gerachica di insiemi di relazioni, etc.. Anche in questa fase può essere decisivo l' intervento umano per effettuare scelte in situazioni critiche.

3. *Concettualizzazione*

   L' analisi ed il confronto degli indicatori individuati per ogni relazione si traduce nell' identificazione delle cosiddette tabelle "spurie" (di associazione, di attributo multivalore, di decodifica) e nella formulazione di ipotesi sul significato concettuale degli elementi dello schema relazionale. Tali ipotesi, insieme alla segnalazione di relazioni non identificate concettualmente, vengono infine sottoposte al vaglio dell' utente, che potrà eventualmente decidere di rieseguire fasi del processo di RE, operando scelte alternative a quelle effettuate in precedenza.

4.5.1. Individuazione delle chiavi primarie e candidate

Si distinguono tre casi mutuamente esclusivi relativamente alla generica relazione T (Fig.2):

```
table_key_detects(T) :-
   nl,nl,nl,write('      TABELLA  : '),
   write(T),nl,
   write('     ------------------------------------'),
   nl,nl,
   index_detects(T),
   (primary_key(T,_) -> true;
      (candidate_key(T,_) ->
        primary_key_def(T,candidate);
        (setof(A,not_null_column(A,T),LA),
         subsets(LA,LAS),
         rules_verify(T,LAS),
         acquire_information(T)))).
```

Fig.2 - Predicato table_key_detects

1. Nel catalogo viene rilevata la definizione esplicita di chiave primaria oppure la definizione di un solo indice unico per l' insieme K di attributi di T. Allora si asserisce :

$$primary\_key(T,K).$$

2. Nel catalogo vengono rilevate due o più definizioni esplicite di chiave candidata oppure due o più indici unici. In questa eventualità in corrispondenza di ognuno di tali insiemi di attributi K_i si asserisce:

$$candidate\_key(T,K_i)$$

Quindi si procede nella individuazione della chiave candidata minimale K considerando eventualmente il rapporto $U_T(K_i)$ che misura la frequenza di utilizzo della chiave candidata in esame.

$$U_T(K_i) = \frac{S_T(K_i)}{Q_T} = \frac{\text{n° di utilizzi di } K_i \text{ nelle } < \text{search\_condition} >}{\text{n° di istruzioni SQL che riferiscono T}}$$

A questo punto l' utente, dipendentemente dal grado di affidabilità riconosciuta alle informazioni raccolte, decide quale dei due fatti asserire:

primary_key(T,K) o *possible_primary_key(T,K)*

3. Qualora non venga rilevata la definizione esplicita di chiave primaria e non risulti definito nessun indice unico per T, si definisce l' insieme P di attributi `NOT NULL` di T e viene proposto come possibile chiave per T ogni sottoinsieme S di P per il quale tutti i suoi attributi sono referenziati in almeno una clausola `WHERE` di una istruzione SQL e la struttura del codice esclude che la selezione basata su tali attributi comporti il ritrovamento di più tuple (cicli di prelievo, operatori di aggregazione, clausole `ORDER_BY` o `GROUP_BY`, etc.).

In questo caso, per ognuno di tali sottoinsiemi, viene generata l' asserzione:

proposed_possible_key(T,S)

L' utente viene quindi posto di fronte alla scelta di definire una chiave primaria certa o possibile per T oppure esaminare le frequenze di utilizzo di alcune possibili chiavi.

Il caso 3 si deve concludere obbligatoriamente con l' individuazione della chiave primaria per T attraverso l' asserzione di uno dei due fatti:

primary_key(T,K) o *possible_primary_key(T,K)*

4.5.2. Indicatori dei riferimenti fra tabelle

Questa fase consiste nella rilevazione di particolari situazioni che testimoniano l' esistenza di associazioni tra due relazioni. Il primo passo è costituito dall' identificazione dei sinonimi, i successivi dalla individuazione delle chiavi esterne e dell' esistenza di controlli di integrità.

4.5.2.1. Ricerca dei sinonimi

Mentre i casi di omonimia sono facilmente risolti con la scelta iniziale di utilizzare in tutto il progetto il *nome esteso* degli attributi (`nome_relazione.nome_attributo`), l' individuazione dei sinonimi richiede un po' di elaborazione aggiuntiva.

Infatti, essendo SQL un linguaggio con una struttura dei tipi molto semplice ed un controllo statico dei tipi molto debole, nulla o poco varrebbe confrontare i tipi di domini attraverso le informazioni contenute nel catalogo.

Quindi il metodo seguito (Fig.3) per l' individuazione dei sinonimi prevede la rilevazione automatica di alcuni indicatori SQL che suggeriscono identità di dominio fra attributi diversi, a seguito della presenza di predicati di confronto nelle clausole `WHERE` (equijoin).

La sinonimia potrebbe essere dedotta anche da particolari strutture del codice, analizzando il ruolo svolto dalle variabili ospite. Un caso tipico (Fig.4) è costituito dallo smembramento di un *join* in due interrogazioni separate su tabelle diverse. Questo particolare aspetto è ancora

oggetto di ulteriore analisi, perché una sua corretta implementazione richiede l' identificazione delle dipendenze tra i dati.

In entrambi i casi si procede alla generazione del fatto Prolog:

synonim_attributes (Table1, Attribute1, Table2, Attribute2).

```
explicit_join_detects([ID|LID]) :-
   from(ID,LT),
   where(ID,LCong),
   search_condition_analyzer(LT,LCong),
   explicit_join_detects(LID).

explicit_join_detects([]).

search_condition_analyzer(LT,LCong) :-
   (LCong = [[Cong]|List] ->
      (condition_conjunction_analyzer(LT,Cong),
       search_condition_analyzer(LT,List));
      condition_conjunction_analyzer(LT,LCong)).

search_condition_analyzer(_,[]).
```

Fig.3 - Predicati explicit_join_detects e search_condition_analyzer

```
implicit_join_detects([sub(T,C,ID)|Lsub]) :-
   select(ID,_,LC),
   from(ID,LT),
   (LC=[Arg]->
      (is_attribute(LT,Arg,T1,C1)->
        assert(synonim_attributes(T,C,T1,C1));
        true),
   implicit_join_detect(Lsub)).

implicit_join_detect([]).
```

Fig.4 - Predicato implicit_join_detects

4.5.2.2. Individuazione delle chiavi esterne

Dopo il rilevamento delle relazioni di sinonimia tra attributi, può essere intrapreso il procedimento di individuazione delle chiavi esterne, *modellanti le associazioni tra tabelle*. Il procedimento (Fig.5) si sviluppa nell' esecuzione di tre passi consecutivi:

1. *Annotazione delle chiavi esterne dichiarate esplicitamente*

   Se il DDL-SQL utilizzato possiede meccanismi di dichiarazione appropriati, l' unica azione da eseguire è asserire il fatto:

   foreign_key (Referencing_table, FK_attributes_list, Referenced_table)

   per ogni tabella (Referencing_table) in cui è definito un riferimento (FK_attributes_list) ad un altra tabella (Referenced_table).

2. *Individuazione di chiavi esterne non dichiarate esplicitamente*

   Per ogni relazione T per cui sia certa la composizione della chiave primaria PK, si individuano i sinonimi degli attributi componenti PK appartenenti ad una certa relazione T'. Essi costituiscono le componenti di una chiave esterna definita in T' che riferisce T. Conseguentemente viene asserito il fatto:

   foreign_key (T', FK_attributes_list, T).

3. *Individuazione di chiavi esterne relative a chiavi primarie non certe*

Nel caso di relazioni per cui è disponibile solo un' indicazione di *possible_primary_key* (PPK), viene eseguito lo stesso procedimento del passo precedente, ma la caratteristica di incertezza dell' informazione sulla chiave primaria deve essere trasferita anche al risultato. Quindi, rilevando la presenza di un' insieme di attributi, tutti appartenenti alla stessa relazione T', sinonimi delle componenti della PPK di T, viene asserito il fatto

possible_foreign_key (T', PFK_attributes_list, T).

```
foreign_key_detects(T,[K|LK],NT,NK) :-
   foreign_key_detects(T,LK,NT,NK1),
   (synonim_attributes(T,K,NT,K1) ->
      append([K1],NK1,NK);
      (synonim_attributes(NT,K1,T,K) ->
         append([K1],NK1,NK);
         fail)).

foreign_key_detects(_,[],_,[]).
```

Fig. 5 - Predicato foreign_key_detects

4.5.2.3. Vincoli di integrità referenziale.

Data la natura non certa delle indicazioni derivate al passo precedente, è opportuno tentare una loro conferma sulla base di informazioni derivabili dalla eventuale presenza di controlli dei vincoli di integrità referenziale eseguiti a programma (Fig.6).

L' esistenza di un vincolo di integrità referenziale è normalmente rivelata dalla presenza di procedure ad hoc che scattano a seguito di inserzioni, cancellazioni o modifiche di tuple.

È realistico ipotizzare la possibilità di riconoscere i semplici *patterns* procedurali di uso più frequente automaticamente o, in alternativa, in modo manuale.

Corrispondentemente all' individuazione del riferimento FK ∅ PK tra le tabelle T_referencing e T_referenced viene asserito il fatto:

referential_integrity_constraints(T_referencing, FK, T_referenced)

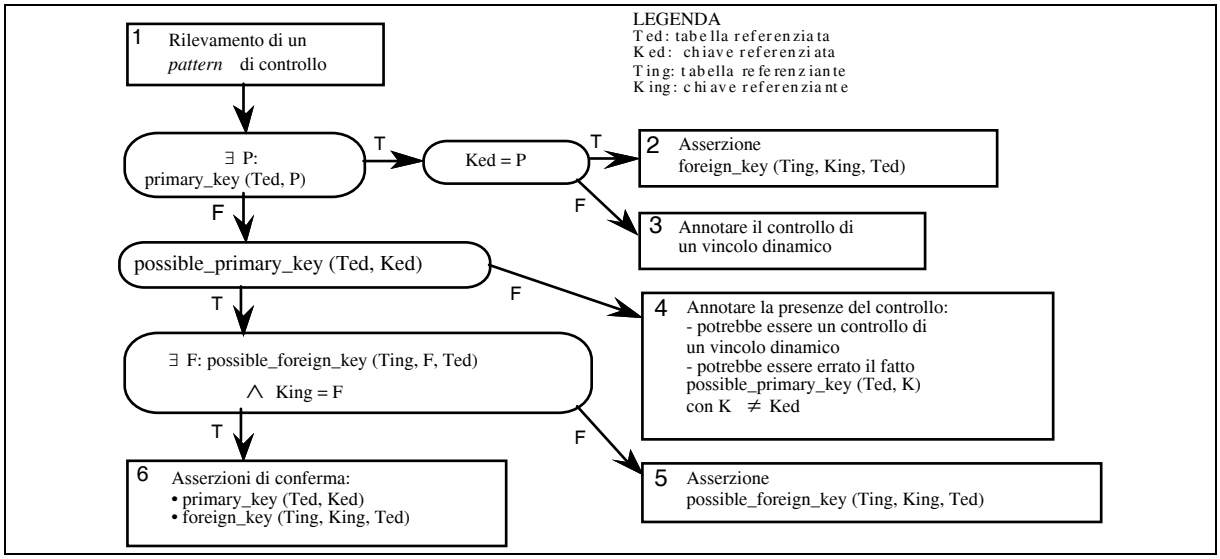

Fig. 6 - Algoritmo di analisi dei controlli dei vincoli di integrita referenziale

4.5.3 Individuazione degli indicatori

Le fasi precedenti forniscono a questo punto le informazioni necessarie per implementare la ricerca degli indicatori, passo fondamentale per la corretta rilevazione degli elementi dello schema concettuale. Un predicato implementa questa fase attraverso un ciclo di ricerca su ciascuna relazione, in cui, ad ogni iterazione, vengono individuati ed elencati gli indicatori per categoria di appartenenza. Vediamo, per ciascuna di tali categorie, gli indicatori individuati, considerando una generica relazione T:

Indicatori di schema	
all_key(T)	T è una relazione tutta PK
predominance_key(T)	T è una relazione con predominanza di PK;
only_one_attribute_not_key(T)	T ha un solo attributo non PK
obbligatory_attribute_not_key(T)	T ha un solo attributo non PK ed obbligatorio
possibile_predominance_key(T)	T ha possibile predominanza di PK
Indicatori di chiave primaria	
corresponding_primary_key(T,List)	la PK di T è in corrispondenza con le PK delle relazioni in List
including_foreign_key(T,List)	la PK di T contiene le PK di altre relazioni,specificate in List
composed_of_foreign_key(T,List)	la PK di T è composta dalle PK di altre relazioni,indicate in List
Indicatori di codice SQL	
in_join_with_other_table(T,List),	T ha join con altre relazioni
refered_from_only_one_table(T,Table)	T è riferita da una sola relazione
refered_from_more_table(T,List)	T è riferita da più relazioni.
Indicatori procedurali	
primary_key_reference *possible_primary_key_reference*	controlli dei vincoli di integrità referenziale sulla relazione T
conditioned_insert_statement *conditioned_update_statement* *conditioned_delete_statement*	comandi d'inserzione, modifica e cancellazione sulla relazione T condizionati da un'istruzione di controllo.

```
conceptual_meaning_table_detects(T) :-
   (is_relationship_table(T) ->
      (assert_t(proposed_relationship_table(T,L)),
       write('    relationship_table between '),
       write(L),nl,nl);
      true),
   (is_sub_entity_table(T,T_up) ->
      (assert_w(proposed_sub_entity_table(T,T_up)),
       write('    sub_entity_table of '),
       write(T_up),nl,nl);
      true),
   (is_generic_entity_table(T,LT) ->
      (assert_w(proposed_generic_entity_table(T,LT)),
       write('    generic_entity_table on '),
       write(LT),nl,nl);
      true),
   (is_entiy_table(T) ->
      (assert_w(proposed_entity_table(T)),
       write('    entity_table '),nl,nl);
      true).
```

Fig.7 - Predicato conceptual_meaning_table_detects

4.5.4. Concettualizzazione

In questa fase, gli indicatori rilevati nelle fasi precedenti vengono utilizzati per identificare le relazioni "spurie" che sono state introdotte nella fase di progettazione logico-fisica per

rappresentare gli elementi dello schema ER che non avevano un equivalente diretto (associazioni con attributi, associazioni n-arie, attributi multivalore, etc.).
La corrispondenza tra i concetti e gli indicatori è presentata nella matrice degli indicatori riportata in appendice. Al predicato *conceptual_meaning_table_detects* è affidato il compito di individuare il significato concettuale di ogni relazione dello schema (Fig. 7).

5. Conclusioni

La ricostruzione dello schema ER di una base di dati permette di identificare anche i vincoli che vengono garantiti a livello procedurale, e quindi consente di reingegnerizzare le applicazioni traendo vantaggio dalle caratteristiche dei DBMS più recenti, che permettono di definire a livello di schema molti vincoli (intervallo di valori, integrità referenziale, etc.).
Nel presente lavoro sono state illustrate le problematiche e le linee guida di una metodologia di RDBRE che partendo dalle descrizioni degli schemi relazionali contenute nel catalogo e da informazioni estratte dal codice (SQL/linguaggio ospite) di programmi che utilizzano la base di dati, deduca gli elementi costitutivi e la struttura dello schema concettuale corrispondente.
La metodologia proposta appare dotata di un' impostazione innovativa rispetto ad approci similari descritti in letteratura. Partendo, infatti, dalla conoscenza strutturale statica della base di dati, il particolare approccio "cognitivo" adottato nelle fasi di riconoscimento di specifiche proprietà delle relazioni tenta di interpretare l' *uso* che le applicazioni fanno dei dati, attraverso l' individuazione e l' analisi di alcune procedure di interrogazione e di manipolazione (*patterns* SQL/procedurali) considerate significative per il rilevamento delle proprietà cercate.
È stato sviluppato in linguaggio Prolog un prototipo dimostrativo di un "sistema esperto", che implementa le indicazioni progettuali derivate dai concetti metodologici. La base di conoscenza costituita dalle regole di riconoscimento dei patterns può essere modificata con relativa facilità, sia per aumentarne le potenzialità con la definizione di nuove regole, sia per specializzare le regole alle caratteristiche del DDL, DML e linguaggio ospite analizzati.
Poiché l' approccio proposto richiede una certa interazione con l' utente è ovvio che le potenzialità di riconoscimento e la qualità della base di conoscenza ottenuta dipendono dal suo grado di esperienza di sviluppo e di implementazione di applicazioni orientate ai dati e dalla conoscenza della realtà applicativa.
Come sviluppo futuro, si prevede l' ingegnerizzazione dello strumento realizzato e la sua integrazione in un tool di reverse engineering in corso di realizzazione ([Signore93]).

Bibliografia

[Albano91] Albano A., Orsini R.: *Modelli dei dati e linguaggi per basi di dati*, Servizio Editoriale Universitario di Pisa, 1991.

[Batini92] Batini C., Ceri S., Navathe S.B.: *Conceptual Database Design: An Entity-Relationship Approach*, The Benjamin/Cummings Publishing Company,Inc., 1992.

[Canfora93] Canfora G., Cimitile A., Munro M.: *A Reverse Engineering Method for Identifying Reusable Abstract Data Type*, Proceedings IEEE. Working Conference on Reverse Engineering, Baltimora 1993.

[Ceri86] Ceri S., Gottlob G.: *Normalization of Relations and Prolog*, Communications of the ACM, June 1986, Vol.29, num.6.

[Chiang94] Chiang R.H.L., Barron T.M., Storey V.C.: *Reverse engineering of relational databases: Extraction of an EER model from a relational database*, Data & Knowledge Engineeering, Vol. 12, N. 2 (March 1994), pp. 107-142

[Chikofsky90] Chikofsky E.J., Cross II J.H.: *Reverse Engineering and Design Recovery: A Taxonomy*, IEEE Software, 7(1): 13-17, January 1990.

[Date87] Date C.J., White C.J.: *A guide to DB2 - Second edition*, Addison-Wesley(1987)

[Ghezzi91] Ghezzi C., Fugetta A., Morasca S., Morzenti A., Pezzè M.: *Ingegneria del Software - Progettazione, Sviluppo e Verifica*, Mondadori Informatica, 1991.

[Hainaut93] Hainaut J-L., Chandelon M., Tonneau C., Joris M.: *Contribution to a Theory of Database Reverse Engineering*,, Proceedings IEEE Working Conference on Reverse Engineering, Baltimora 1993

[ISO9075] *Information technology - Database languages - SQL2*; ISO standard N. 9075

[Korth91] Korth H.F., Silberschatz A.: *Database System Concepts*, McGraw-Hill International Editions, Second Edition, 1991.

[Premerlani93] Premerlani W.J., Blaha M.R.: *An Approach for Reverse Engineering of Relational Databases*, Proceedings IEEE Working Conference on Reverse Engineering, Baltimora 1993.

[Signore93] Signore O., Loffredo M.: *Re-Engineering towards Object-Oriented Environments: the TROOP Project*, Proceedings of The Eighth International Symposium on Computer and Information Sciences (ISCIS VIII), November 3-5, 1993, Istanbul, Turkey (Sponsored by IEEE)

[Signore94] Signore O., Menicucci S., Bartoli R.: *Un repository per la reingegnerizzazione di programmi e l' integrazione in un tool CASE"*, (comunicazione presentata al Second Italian SYBASE Developers Meeting, Firenze 17-18 giugno 1993). Rapporto tecnico PFSICP

[Teorey86] Teorey T.J., Yang D., Fry J.P.: *A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model*, ACM Computing Surveys, Vol.18, No.2, Giugno 1986.

Appendice: Matrice degli indicatori

Indicatori di schema	Indicatori di chiave primaria	Indicatori codice SQL	Indicatori procedurali	CONCETTO
	• non corrisponde a PK di altre relazioni • non contiene FK			**Relazione di entità** - senza attributi multivalore - non appartiene a gerarchie
• possibile predominanza di chiave (PK + 1÷2 attributi) • vincoli di integrità referenziale sulle FK (se il DDL ne permette la definizione)	• composta da più FK	• *join* con relazioni associate	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Relazione di associazione**
• predominanza di chiave (PK + un attributo) • attributo non chiave obbligatorio ed unico	• non contiene FK	• Viene riferita in più relazioni (la sua PK è FK in più relazioni) • poche o nessuna *insert*, *delete* o *update* (opp.con privilegi *DB administrator*)	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Relazione di tipo enumerativo (di decodifica)**
• predominanza di chiave (PK + un attributo) • attributo non chiave obbligatorio ed unico	• corrisponde a PK di altre relazioni	• Le procedure di input/output sulle relazioni che hanno PK corrispondente sono effettuate utilizzandola sempre per operazioni di codifica/decodifica di chiave. • Le operazioni di selezione e manipolazione sulle relazioni che hanno PK corrispondente possono utilizzarla per operazioni di codifica/decodifica di chiave		**Relazione di codifica di chiave**
• tutta PK • vincoli di integrità referenziale sulla FK (se il DDL ne permette la definizione)	• PK formata da FK + un solo attributo • generalmente FK e PK della relazione riferita con stesso nome	• viene riferita da una sola relazione	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Attributo multivalore semplice**
• predominanza di chiave (PK + un attributo) • attributo non chiave obbligatorio • vincoli di integrità referenziale sulla FK (se il DDL ne permette la definizione)	• PK formata da FK + un solo attributo discriminatore • generalmente FK e PK della relazione riferita con stesso nome	• viene riferita da una sola relazione	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Attributo multivalore semplice** (i singoli valori per ogni entità vengono distinti dal valore del discriminatore)
• tutta PK • vincoli di integrità referenziale sulla FK (se il DDL ne permette la definizione)	• PK contenente una FK + altri attributi • generalmente FK e PK della relazione riferita con stesso nome	• viene riferita da una sola relazione	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Attributo multivalore composto**
• vincoli di integrità referenziale sulla FK (se il DDL ne permette la definizione)	• PK formata da FK+ un solo attributo discriminatore • generalmente FK e PK della relazione riferita con stesso nome	• viene riferita da una sola relazione	• controlli dei vincoli di integrità referenziale se non definiti esplicitamente	**Attributo multivalore composto** (i singoli valori per ogni entità vengono distinti dal valore del discriminatore)
	• insieme di relazioni (#≥2) con PK corrispondenti	• ogni relazione dell'insieme ha *join* autonomi con altre relazioni	• presenza di *switch* oppure cascata di *if…then…else* per condizionare le operazioni sulle tuple dei sottinsiemi tramite inserzione, selezione, cancellazione, aggiornamento	**Gerarchia disaggregata di sottinsiemi o di partizioni** (coinvolge più relazioni)
		• in generale è caratterizzata da associazioni con numerose relazioni	• presenza di *switch* oppure cascata di *if…then…else* per condizionare le operazioni sulle tuple dei sottinsiemi tramite inserzione, selezione, cancellazione, aggiornamento	**Gerarchia aggregata di sottinsiemi o di partizioni** (coinvolge una sola relazione)
	• corrisponde a PK di altre relazioni	• ha *join* autonomi con altre relazioni	• presenza di *switch* oppure cascata di *if…then…else* per condizionare le operazioni sulle tuple dei sottinsiemi tramite inserzione, selezione, cancellazione, aggiornamento	**Entità Generica di gerarchia**
	• corrisponde a PK di altre relazioni	• ha *join* autonomi con altre relazioni	• presenza di *switch* oppure cascata di *if…then…else* per condizionare le operazioni sulle tuple dei sottinsiemi tramite inserzione, selezione, cancellazione, aggiornamento	**Entità di sottinsieme**

• in generale non ha predominanza di chiave	• contiene FK	• in generale ha *join* autonomi		**Entità debole**