Progettazione Concettuale di Data Warehouse da Schemi Logici Relazionali

Matteo Golfarelli, Stefano Rizzi

DEIS - Università di Bologna
{mgolfarelli|srizzi}@deis.unibo.it

Sommario

I sistemi di data warehousing consentono ai dirigenti di impresa di acquisire e integrare informazioni provenienti da sorgenti eterogenee e di interrogare efficientemente basi di dati di grandi dimensioni. La costruzione di un data warehouse richiede l'adozione di tecniche di progetto completamente diverse da quelle utilizzate per i sistemi informativi operazionali; d'altronde, nessun passo significativo è stato ancora fatto per mettere a punto una metodologia completa di progetto. In questo lavoro viene proposto un approccio alla progettazione concettuale di data warehouse a partire dallo schema logico della base di dati relazionale sottostante; il modello concettuale utilizzato, detto Dimensional Fact model, consiste di un insieme di schemi di fatto i cui elementi costituenti sono misure, dimensioni e gerarchie. Per ogni fatto individuato viene costruito in maniera semi-automatica un albero che evidenzia le dipendenze funzionali presenti tra gli attributi dello schema logico; ciascun albero può poi essere facilmente trasformato in uno schema di fatto.

1. Introduzione

Il mondo accademico sta dedicando crescente attenzione ai temi riguardanti i data warehouse; infatti, lo sviluppo di sistemi di supporto alle decisioni sarà probabilmente uno degli argomenti principali di ricerca per i prossimi anni. Le imprese, dopo aver investito tempo e risorse per costruire sistemi informativi complessi, chiedono strumenti per ottenere in tempi ridotti informazioni riassuntive che possano essere di aiuto ai dirigenti nella pianificazione e nel processo decisionale. I sistemi di data warehousing soddisfano questa esigenza permettendo ai dirigenti di acquisire e integrare informazioni provenienti da diverse fonti e di interrogare efficientemente database di grandi dimensioni. Queste informazioni vengono raccolte in un singolo "contenitore", chiamato *data*

warehouse (DW), che può essere direttamente consultato e usato come sorgente per costruire *data mart* orientati verso specifiche aree dell'impresa [15].

Mentre è universalmente riconosciuto che un DW si appoggia su un modello multidimensionale dei dati, poca ricerca è stata compiuta sulle metodologie per il progetto concettuale a partire dalle specifiche degli utenti. D'altro canto, è evidente che un accurato progetto concettuale è requisito fondamentale per la costruzione di un sistema informativo ben documentato e pienamente rispondente alle specifiche. Il modello Entity/Relationship è molto diffuso nelle imprese come strumento concettuale per la documentazione standard dei sistemi informativi relazionali; essendo però orientato a interrogazioni di navigazione più che di sintesi dei dati, mal si presta a essere adottato nel caso dei DW [12].

In questo lavoro viene descritto un modello concettuale grafico per DW, chiamato *Dimensional Fact Model* (DFM). La rappresentazione costruita tramite il DFM è detta *schema dimensionale* e consiste di un insieme di schemi di fatto i cui elementi costituenti sono misure, dimensioni e gerarchie.

In [6] abbiamo delineato una metodologia per derivare uno schema dimensionale dagli schemi Entity/Relationship che descrivono il sistema informativo operazionale sottostante. D'altronde, spesso gli schemi Entity/Relationship in possesso delle imprese sono incompleti e scorretti; a volte, l'unica documentazione presente è costituita da schemi logici. Proponiamo pertanto un approccio alla progettazione concettuale di DW a partire dallo schema logico della base di dati relazionale sottostante. Per ogni fatto individuato viene costruito in maniera semi-automatica un albero che evidenzia le dipendenze funzionali presenti tra gli attributi dello schema logico; ciascun albero può poi essere facilmente trasformato in uno schema di fatto.

2. Il progetto di data warehouse nella letteratura

Dal punto di vista funzionale, il processo di data warehousing consiste in tre fasi: estrazione dei dati da sorgenti operazionali distribuite e loro integrazione; organizzazione dei dati nel DW; accesso efficiente e flessibile ai dati integrati. La prima fase include problematiche tipiche dei servizi informativi distribuiti, come la gestione di dati inconsistenti e di strutture dati incompatibili (si veda, ad esempio, [16]). La terza fase richiede capacità di navigazione degli aggregati [7], ottimizzazione di

interrogazioni complesse [3], tecniche di indicizzazione avanzate [13] e interfacce visuali amichevoli per l'OLAP [4] e il data mining [5].

Per quanto riguarda la seconda fase, la progettazione di un DW richiede tecniche completamente differenti da quelle adottate nei sistemi informativi operazionali. D'altronde, la maggior parte della letteratura scientifica sul progetto di DW riguarda i modelli logici e fisici e tratta argomenti specifici come la materializzazione delle viste [2] [10] e la selezione degli indici [8] [11]; nessun passo significativo è stato ancora fatto per mettere a punto una metodologia completa di progettazione. In [14], l'autore propone un approccio al progetto di DW basato su un modello dell'impresa che coincide di fatto con uno schema relazionale. Purtroppo i progetti concettuale e logico appaiono mischiati; poiché il progetto logico è necessariamente orientato verso un modello logico (relazionale in questo caso), non viene previsto nessun modello concettuale unificante. [1] e [9] propongono due modelli di dati per basi di dati multidimensionali e le relative algebre; entrambi i modelli sono di tipo logico, quindi non trattano aspetti di modellazione concettuale quali la struttura delle gerarchie di attributi e i vincoli di non additività.

3. DFM: un modello concettuale per il data warehouse

La rappresentazione generata dal DFM è detta *schema dimensionale* e consiste di un insieme di *schemi di fatto*. Gli elementi di base degli schemi di fatto sono i fatti, le misure, le dimensioni e le gerarchie. Un *fatto* è un evento di interesse per l'impresa; le *misure* descrivono quantitativamente un fatto; le *dimensioni* determinano la granularità minima adottata per rappresentare un fatto; le *gerarchie* descrivono come le istanze di un fatto possono essere aggregate e selezionate significativamente per il processo decisionale.

Uno schema di fatto è strutturato come un albero la cui radice è un fatto. Quest'ultimo è rappresentato da un rettangolo che riporta il nome del fatto e, tipicamente, uno o più attributi numerici a valori continui (*misure*) che "quantificano" il fatto da diversi punti di vista. La Figura 1 contiene un esempio di schema di fatto che modella le spedizioni effettuate da una ditta, basato sullo schema a stella riportato in [12]; `quantità spedita` e `importo` sono misure.

I circoli bianchi rappresentano *attributi dimensionali* definiti su domini discreti (ad esempio `destinatario` e `città`). Quelli connessi direttamente al fatto sono detti *dimensioni* (nell'esempio:

`data`, `prodotto`, `destinatario`, `magazzino`, `contratto` e `promozione`); gli altri sono organizzati in *gerarchie* con radice nelle dimensioni. La dimensione in cui una gerarchia ha radice rappresenta la sua granularità più fine; gli altri attributi dimensionali della gerarchia definiscono granularità via via più spesse.

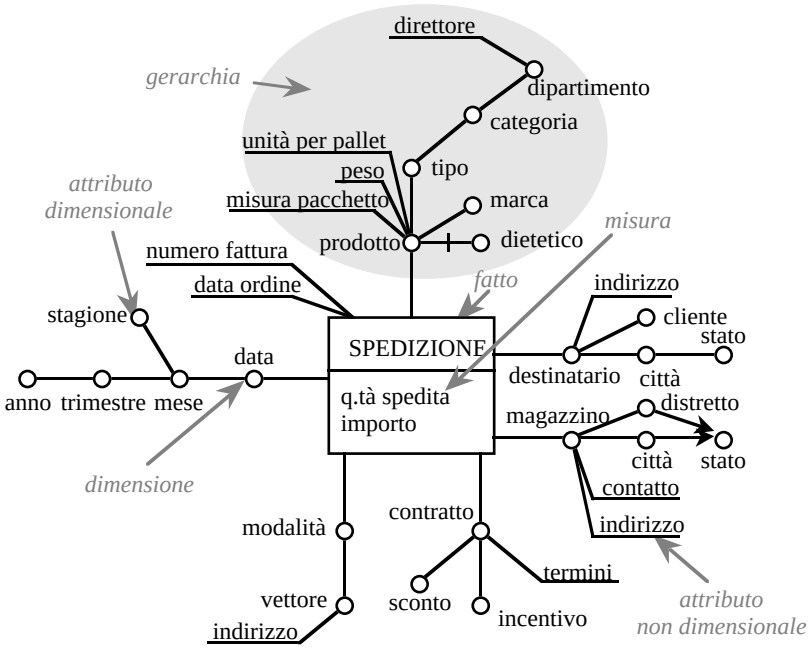

Figura 1. Schema di fatto per le spedizioni effettuate da una ditta.

Alcuni attributi nelle gerarchie sono rappresentati da linee invece che da circoli (ad esempio `peso` in Figura 1), e sono detti *attributi non dimensionali*. Un attributo non dimensionale contiene informazioni aggiuntive su un attributo dimensionale; diversamente da quest'ultimo, però, non può essere utilizzato per l'aggregazione e quindi non definisce alcuna granularità.

Ogni arco presente nello schema rappresenta una relazione -a-uno, ovvero una dipendenza funzionale, tra due attributi. Ad esempio, un negozio determina esattamente una città di appartenenza e un indirizzo. Di conseguenza, ogni percorso orientato all'interno di una gerarchia rappresenta una relazione -a-uno tra l'attributo di partenza e quello di arrivo. Da un attributo non dimensionale non possono uscire archi.

Lo schema di fatto può non essere un "vero" albero: infatti, può accadere che due o più percorsi distinti connettano due dati attributi dimensionali all'interno di una gerarchia, a patto che ogni percorso orientato continui a rappresentare una relazione -a-uno. Si consideri ad esempio la gerarchia su `magazzino`: ogni stato contiene città e distretti, tra i quali non esiste alcuna relazione di inclusione;

ciò nonostante, un magazzino appartiene a un unico stato qualunque dei due percorsi venga seguito (ossia, `magazzino` determina `stato`).

Alcuni archi delle gerarchie sono contrassegnati da un trattino; essi esprimono relazioni opzionali tra coppie di attributi. Ad esempio, l'attributo `dietetico` assume un valore solo per i prodotti alimentari; per gli altri prodotti assumerà valore nullo.

3.1. Istanze di fatto

Un fatto esprime un'associazione molti-a-molti tra le dimensioni. Ciascuna combinazione di valori assunti dalle dimensioni definisce un'*istanza di fatto primaria*, caratterizzata da uno e un solo valore per ogni misura; le istanze di fatto costituiscono le informazioni elementari rappresentate nel DW. Nell'esempio, una istanza di fatto primaria descrive la quantità di un dato prodotto spedita in una certa data da un certo magazzino a un certo destinatario sulla base di un certo contratto e adottando una certa modalità di spedizione.

L'interrogazione di un DW è in genere mirata a estrarre dati riepilogativi per riempire un report strutturato che possa poi essere convenientemente analizzato per scopi statistici o decisionali. Poiché analizzare i dati al massimo livello di dettaglio è spesso fonte di confusione, può risultare indispensabile aggregare le istanze di fatto primarie in *istanze di fatto secondarie* a differenti livelli di astrazione. L'aggregazione richiede di definire un operatore per comporre i singoli valori assunti dagli attributi delle istanze primarie in valori che caratterizzino gli attributi delle istanze secondarie. La Figura 2 riporta un semplice esempio.

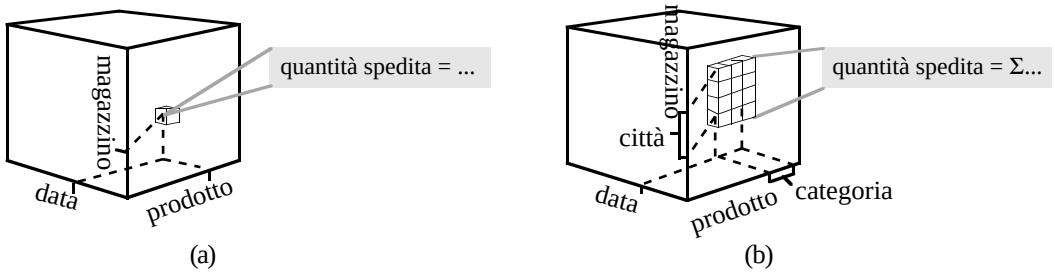

Figura 2. Istanze di fatto primarie (a) e secondarie (b) per l'esempio delle spedizioni (considerando, per comodità di disegno, le sole dimensioni `prodotto`, `data` e `magazzino`).

In linea di massima, la maggior parte delle misure dovrebbe essere *additiva* [12]; ciò significa che l'operatore di somma viene utilizzato per l'aggregazione lungo tutte le dimensioni. Una misura additiva rispetto a tutte le dimensioni è `quantità spedita`: infatti, ad esempio, il numero di

spedizioni per una categoria di prodotti è la somma del numero di spedizioni per tutti i prodotti di quella categoria.

Una misura è detta *non additiva* lungo una dimensione se i suoi valori non possono essere sommati quando si aggregano le istanze primarie lungo quella dimensione. Esempi di misure non additive sono tutte quelle che esprimono un livello (inventari, temperature, ecc.). Un livello di inventario è non additivo lungo la dimensione tempo, ma lo è lungo le dimensioni prodotto e magazzino. Una misura di temperatura è non additiva lungo tutte le dimensioni, poiché sommare due temperature ha comunque poco significato. D'altronde, per questi tipi di misure, l'aggregazione può essere effettuata tramite altri operatori, come quelli di media, massimo, minimo. Per altre misure, qualunque tipo di aggregazione è intrinsecamente impossibile per motivi concettuali.

Nel DFM, le misure sono additive lungo tutte le dimensioni per default. Ciascuna misura viene poi eventualmente collegata con le dimensioni lungo cui non è additiva tramite una linea tratteggiata; se è possibile utilizzare un operatore di aggregazione diverso dalla somma, esso viene riportato esplicitamente sullo schema. La Figura 3 mostra un esempio.

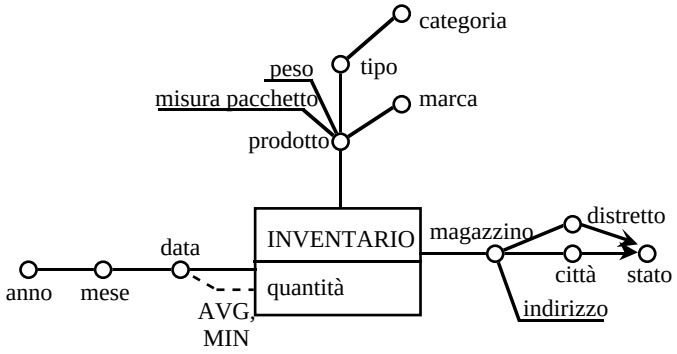

Figure 3. Rappresentazione della non additività sullo schema di fatto.

4. Progettazione concettuale da schemi relazionali

La maggior parte dei sistemi informativi realizzati durante l'ultimo decennio è basata sul modello relazionale; è quindi di estremo interesse studiare come lo schema concettuale di un DW possa essere derivato dagli schemi logici che descrivono una base di dati relazionale. La metodologia che proponiamo per la costruzione di uno schema dimensionale consiste dei seguenti passi:

1. Definizione dei fatti.

2. Per ciascun fatto:
   a. Costruzione di un albero degli attributi.
   b. Potatura e innesto dell'albero degli attributi.
   c. Definizione delle dimensioni.
   d. Definizione delle misure.
   e. Definizione delle gerarchie.

Nei paragrafi seguenti descriveremo ciascun passo con riferimento all'esempio delle spedizioni, per il quale riportiamo di seguito uno schema logico semplificato (le chiavi primarie sono sottolineate; per ogni chiave importata viene indicato lo schema di provenienza).

```
SPEDIZIONI(data,codProd:PRODOTTI,codDest:DESTINATARI,quantità,codOrd:ORDINI,
           annoOrd:ORDINI,codMag:MAGAZZINI,mod:MODALITÀ)
ORDINI(codOrd,annoOrd,dataOrdine,numFattura,importo)
DESTINATARI(codDest,denominazione,indirizzo,codCittà:CITTÀ)
MAGAZZINI(codMag,indirizzo,codCittà:CITTÀ,numDistr:DISTRETTI,
          codStato:DISTRETTI)
CITTÀ(codCittà,nomeCittà,stato:STATI)
STATI(stato)
DISTRETTI(numDistr,stato:STATI)
PRODOTTI(codProd,descrizione,peso,codMarca:MARCHE,tipo:TIPI)
PROD_DIETETICI(codProd:PRODOTTI,dieta:DIETE)
DIETE(dieta)
PROD_IN_MAGAZZ(codProd:PRODOTTI,codMag:MAGAZZINI,quantità)
MARCHE(codMarca)
TIPI(tipo,categoria)
MODALITÀ(mod,descrizione,vettore:VETTORI)
VETTORI(vettore,indirizzo)
```

La chiave di `SPEDIZIONI` è stata scelta ipotizzando che non possano essere effettuate, nella stessa data, più spedizioni dello stesso prodotto allo stesso destinatario. I distretti sono numerati univocamente all'interno di ciascuno stato.

4.1. Definizione dei fatti

I fatti sono concetti di interesse primario per il processo decisionale; tipicamente corrispondono a eventi che accadono dinamicamente nell'impresa.

Sullo schema di database un fatto corrisponde a uno schema relazionale `F`. Gli schemi che descrivono relazioni frequentemente aggiornate (come `SPEDIZIONI`) sono buoni candidati per la definizione di fatti; quelli che rappresentano proprietà strutturali del dominio, corrispondenti a relazioni quasi-statiche (come `MAGAZZINI` e `PRODOTTI`), non lo sono.

Ciascun fatto identificato sullo schema di database diviene la radice di un differente schema di fatto. Nel seguito focalizzeremo l'attenzione su un singolo fatto, corrispondente allo schema relazionale `R`. Nell'esempio, il fatto di interesse primario per il processo decisionale è la spedizione di una partita di un prodotto, rappresentato nello schema di database dallo schema `SPEDIZIONI`.

4.2. Costruzione dell'albero degli attributi

Data una porzione di interesse di uno schema di database e uno schema relazionale `F`, chiamiamo *albero degli attributi* l'albero che soddisfa i seguenti requisiti:

- ogni vertice corrisponde a un attributo dello schema;
- la radice corrisponde alla chiave primaria di `F`;
- per ogni vertice *v*, l'attributo corrispondente determina funzionalmente tutti gli attributi corrispondenti ai discendenti di *v*.

L'albero degli attributi corrispondente a `F` può essere costruito in modo automatico applicando la seguente procedura ricorsiva:

```
root=nuovoVertice(F);
// nuovoVertice(<schema>) restituisce un nuovo vertice etichettato con il nome
// dello schema; nuovoVertice(<insieme di attributi>) restituisce un nuovo
// vertice etichettato con la concatenazione dei nomi degli attributi
// nell'insieme
traduci(F,root);
```

dove, denotando con `pk(R)` e con `fk(R,S)` l'insieme degli attributi di `R` che formano,

rispettivamente, la chiave primaria di R e una chiave importata da S:

```
traduci(R,v):
// R è lo schema relazionale corrente, v il vertice corrente
{ per ogni attributo a∈R | (a≠pk(R) ∧ (∄S | a∈fk(R,S)))
     aggiungiFiglio(v,nuovoVertice(a));  // aggiunge un figlio al vertice v
  per ogni insieme di attributi A⊂R | (∃S | A=fk(R,S))
  {  next=nuovoVertice(A);
     aggiungiFiglio(v,next);
     traduci(S,next);
  }
  per ogni schema T | pk(T)=fk(T,R)
  {  per ogni attributo a∈T | (a∉pk(R) ∧ (∄S | a∈fk(T,S)))
       aggiungiFiglio(v,nuovoVertice(a));
     per ogni insieme di attributi A⊂T | (∃S | A=fk(T,S))
     {  next=nuovoVertice(A);
        aggiungiFiglio(v,next);
        traduci(S,next);
     }
  }
}
```

La procedura `traduci` costruisce l'albero seguendo le dipendenze funzionali rappresentate nello schema. Il primo ciclo considera le dipendenze tra la chiave primaria di R e ogni altro attributo di R (inclusi, se la chiave è composta, i singoli attributi che la compongono ma esclusi gli attributi che fanno parte di chiavi importate, che vengono considerati al passo successivo). Il secondo ciclo tratta le dipendenze tra la chiave primaria e ogni chiave importata da uno schema S, innescando la ricorsione su S. Il terzo ciclo gestisce le situazioni del tipo:

$R(\underline{k_R},...)$
$T(\underline{k_T}:R,...k_S:S)$
$S(\underline{k_S},...)$

(ad esempio PRODOTTI, PROD_DIETETICI e DIETE) in cui la relazione uno-a-molti tra R e S è rappresentata tramite un terzo schema T.

Alcune considerazioni aggiuntive:

- Lo schema logico può contenere un ciclo di chiavi importate; l'esempio più semplice è uno schema di relazione che include una chiave importata dallo schema stesso. In questo caso, la procedura `traduci` entrerebbe in *loop* su questo ciclo generando un ramo di lunghezza infinita. Poiché rappresentare una associazione ricorsiva sullo schema logico del DW non è possibile, il *loop* deve essere riconosciuto e interrotto.

- Via via che la procedura `traduci` "esplora" uno schema di database ciclico, lo stesso schema relazionale `R` può essere raggiunto due volte attraverso percorsi diversi, generando così due vertici omologhi *v′* e *v″* nell'albero. Se ciascuna tupla di `F` determina esattamente una tupla di `R` qualunque dei due percorsi venga seguito, *v′* e *v″* possono essere fusi in un unico vertice *v* in cui entrano due archi; lo stesso vale per ogni coppia di vertici omologhi discendenti da *v′* e *v″*. Altrimenti, *v′* e *v″* devono essere mantenuti distinti.
- Qualora sia noto che un attributo di uno schema relazionale può assumere valori nulli, l'arco corrispondente dell'albero deve essere contrassegnato da un trattino a significare che la relazione è opzionale.
- Le relazioni -a-molti, descritte da schemi con due o più chiavi importate facenti tutte parte della chiave primaria (ad esempio `PROD_IN_MAGAZZ`), non vengono inserite nell'albero degli attributi poiché rappresentarle nello schema logico del DW sarebbe impossibile senza violare la prima forma normale.

L'albero degli attributi corrispondente allo schema delle spedizioni è riportato in Figura 4.

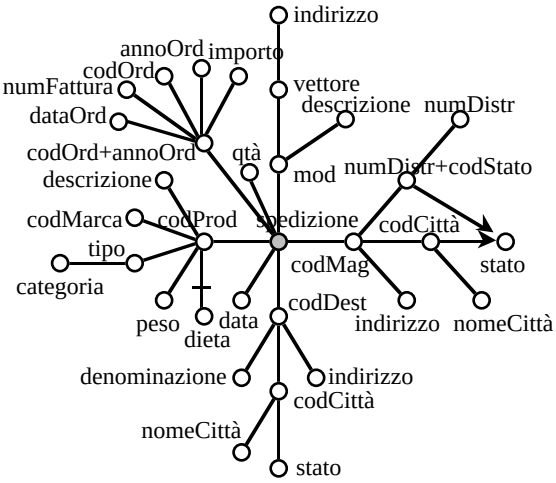

Figure 4. Albero degli attributi per lo schema delle spedizioni (in grigio la radice).

4.3. Potatura e innesto dell'albero degli attributi

In genere, non tutti gli attributi dell'albero sono d'interesse per il DW. Quindi, l'albero può essere "potato" e "innestato" al fine di eliminare i livelli di dettaglio non necessari.

La *potatura* di un vertice *v* si effettua eliminando il sottoalbero con radice in *v*. Gli attributi

eliminati non verranno inclusi nello schema di fatto, quindi non potranno essere usati per aggregare i dati.

L'*innesto* viene utilizzato quando, sebbene un vertice esprima un'informazione non interessante, i suoi discendenti devono essere mantenuti. L'innesto del vertice *v*, con padre *v'*, viene effettuato collegando tutti i figli di *v* direttamente a *v'* ed eliminando *v*. Come risultato verrà perduto il livello di aggregazione corrispondente all'attributo *v*, ma non i livelli corrispondenti ai suoi discendenti. Si noti che innestare un figlio della radice significa aumentare la granularità delle istanze di fatto e, se il vertice innestato ha più di un figlio, può portare all'aumento del numero di dimensioni nello schema di fatto.

Vale la pena riportare una considerazione riguardante le chiavi composte. Sia `R` uno schema con chiave primaria composta dagli attributi $a_1,...a_m$. L'algoritmo delineato nella Sezione 4.2 traduce `R` in un vertice $c=a_1+...a_m$ con figli $a_1,...a_m$. Essenzialmente, si possono presentare due situazioni:

- Se la granularità di `R` deve essere preservata nello schema di fatto, *c* viene mantenuto mentre uno o più dei suoi figli possono essere potati; ad esempio, il vertice `numDistretto+stato` è mantenuto poiché l'aggregazione deve essere effettuata a livello di ogni singolo distretto, ma `numDistretto` può essere potato poiché non esprime aggregazioni interessanti.
- Altrimenti, se la granularità espressa da `R` è troppo fine, *c* può essere innestato e alcuni o tutti i suoi figli mantenuti; ad esempio, il vertice `codOrd+annoOrd` viene innestato e i suoi figli `annoOrd` e `codOrd` vengono potati.

Effettuando inoltre la potatura di `descrizione` e `denominazione`, l'albero degli attributi si trasforma come mostrato in Figura 5.

4.4. Definizione delle dimensioni

Le dimensioni determinano in che modo le istanze di fatto potranno essere aggregate significativamente per il processo decisionale. Le dimensioni devono essere scelte nell'albero degli attributi tra i vertici figli della radice; possono corrispondere ad attributi discreti o a intervalli di valori di attributi discreti o continui. La loro scelta è cruciale per il progetto del DW poiché definisce la

granularità delle istanze di fatto primarie.

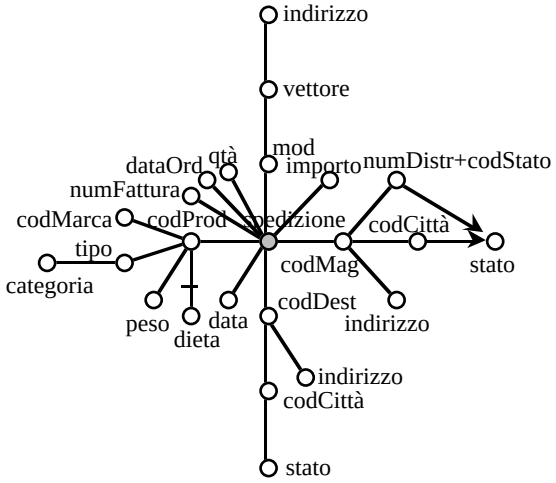

Figure 5. Albero degli attributi per le spedizioni dopo potatura e innesto.

Ciascuna istanza primaria "riassume" tutte le tuple di `F` corrispondenti a una combinazione di valori delle dimensioni. Se tra le dimensioni compaiono tutti gli attributi che costituiscono una chiave candidata di `F`, ogni istanza primaria corrisponde a una tupla; spesso, uno o più degli attributi che identificano `F` vengono potati o innestati, cosicché ciascuna istanza primaria può corrispondere a più tuple.

Nell'esempio, come dimensioni vengono scelti gli attributi `codProd`, `codDest`, `data`, `mod` e `codMag`; di conseguenza, la granularità delle istanze primarie coincide con quella di `SPEDIZIONI`.

È universalmente riconosciuto che il tempo è una dimensione chiave per i DW. In uno schema di database che descrive l'evoluzione del dominio applicativo, uno o più attributi temporali sono necessariamente presenti, e sono ovvi candidati per definire una dimensione. Se dopo la traduzione il tempo appare nell'albero come figlio di un vertice diverso dalla radice, conviene valutare la possibilità di effettuare un innesto che lo renda una dimensione.

4.5. Definizione delle misure

Le misure vengono definite applicando, ad attributi numerici dell'albero, funzioni di aggregazione che operano su tutte le tuple di `F` corrispondenti a ciascuna istanza di fatto primaria. Tipicamente si tratta di somma/media/massimo/minimo di espressioni oppure del conteggio del numero di tuple. Uno schema di fatto può non avere misure, qualora l'unica informazione di interesse sia il verificarsi del fatto.

Le misure prescelte vengono riportate sullo schema di fatto. A questo punto, è utile definire un *glossario* che associ ciascuna misura a un'espressione che descriva come essa può essere calcolata a partire dagli attributi dello schema relazionale. Con riferimento all'esempio delle spedizioni, il glossario può essere compilato come segue:

```
qtà spedita = SPEDIZIONI.quantità
importo = ORDINI.importo
```

tenendo presente che, per come è definito lo schema logico, ogni istanza di fatto primaria conterrà al più una tupla di `SPEDIZIONI`. Se invece la granularità delle istanze fosse stata superiore a quella di `SPEDIZIONI` (ad esempio perché l'attributo `destinazione` non è stato scelto come dimensione), il glossario avrebbe dovuto evidenziare una funzione di aggregazione:

```
qtà spedita = SUM(SPEDIZIONI.quantità)
importo = SUM(ORDINI.importo)
```

In questa fase possono infine essere identificate le eventuali non additività presenti nello schema.

4.6. Definizione delle gerarchie

L'ultimo passo nella costruzione dello schema di fatto è la definizione delle gerarchie sulle dimensioni. Lungo ogni gerarchia, gli attributi devono essere disposti in alberi in cui ogni vertice è legato ai suoi discendenti da una associazione -a-uno.

L'albero degli attributi esprime già un'organizzazione plausibile per le gerarchie; in questa fase è ancora possibile potare e innestare l'albero per eliminare dettagli irrilevanti. È anche possibile aggiungere nuovi livelli di aggregazione definendo opportuni intervalli per attributi numerici; tipicamente ciò viene fatto sulla dimensione tempo.

A questo punto, gli attributi che non verranno usati per l'aggregazione ma solo per motivi informativi possono essere contrassegnati come non dimensionali. Si noti che attributi non numerici figli della radice ma non prescelti come dimensioni devono necessariamente essere potati (se la granularità delle isatnze di fatto primarie è maggiore di quella del fatto) oppure essere rappresentati come non dimensionali (se le due granularità coincidono).

5. Conclusioni

In questo lavoro abbiamo delineato una metodologia per il progetto concettuale di data warehouse a partire dagli schemi logici che descrivono una base di dati operazionale. Il modello concettuale adottato risulta indipendente dal modello logico prescelto per l'implementazione (multidimensionale o relazionale); il passaggio dallo schema dimensionale allo schema logico del DW richiede una metodologia di progettazione logica. Come per i sistemi informativi operazionali, il progetto logico di un DW deve essere basato sulla stima del volume dei dati e del carico di lavoro: il primo verrà calcolato considerando la sparsità dei fatti e le cardinalità degli attributi dimensionali, il secondo sarà espresso in termini di interrogazioni e loro frequenze.

Correntemente stiamo mettendo a punto una metodologia per il progetto logico; tra gli argomenti specifici allo studio citiamo:

- *Partizionamento del DW in data mart integrati*.
- *Materializzazione delle viste*. Questo problema coinvolge l'intero schema dimensionale; infatti la presenza di interrogazioni di drill-across rende auspicabile un'ottimizzazione incrociata tra gli schemi di fatto.
- *Traduzione in fact table e dimension table*.
- *Partizionamento verticale delle fact table*. I tempi di risposta possono essere ridotti sulla base dell'insieme di misure richieste da ciascuna interrogazione.
- *Partizionamento orizzontale delle fact table*. I tempi di risposta possono essere ridotti sulla base della selettività di ciascuna interrogazione.

References

[1] R. Agrawal, A. Gupta, S. Sarawagi. Modeling multidimensional databases. *IBM Research Report*, IBM Almaden Research Center, 1995.

[2] E. Baralis, S. Paraboschi, E. Teniente. Materialized view selection in multidimensional database. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, Athens, Greece, 1997, pp. 156-165.

[3] S. Chaudhuri, K. Shim. Including group-by in query optimization. In *Proc. 20th Int. Conf. on Very Large Data Bases*, pp. 354-366, 1994.

[4] G. Colliat. OLAP, relational and multidimensional database systems. *SIGMOD Record*, vol. 25, n. 3, pp. 64-69, 1996.

[5] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth. Data mining and knowledge discovery in databases: an overview. *Comm. of the ACM*, vol. 39, n.11, 1996.

[6] M. Golfarelli, D. Maio, S. Rizzi. Conceptual design of data warehouses from E/R schemes. In *Proc. Hawaii Int. Conf. on System Sciences*, vol. VII, Kona, Hawaii, pp. 334-343, 1998.

[7] A. Gupta, V. Harinarayan, D. Quass. Aggregate-query processing in data-warehousing environments. In *Proc. 21th Int. Conf. on Very Large Data Bases*, Zurich, Switzerland, 1995.

[8] H. Gupta, V. Harinarayan, A. Rajaraman. Index selection for OLAP. In *Proc. Int. Conf. Data Engineering*, Binghamton, UK, 1997.

[9] M. Gyssens, L.V.S. Lakshmanan. A foundation for multi-dimensional databases. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, Athens, Greece, 1997, pp. 106-115.

[10] V. Harinarayan, A. Rajaraman, J. Ulman. Implementing Data Cubes Efficiently. In *Proc. of ACM Sigmod Conf.*, Montreal, Canada, 1996.

[11] T. Johnson, D. Shasha. Hierarchically split cube forests for decision support: description and tuned design. *Bullettin of Technical Committee on Data Engineering*.vol. 20, n. 1, 1997.

[12] R. Kimball. *The data warehouse toolkit*. John Wiley & Sons, 1996.

[13] D. Lomet, B. Salzberg. The Hb-Tree: a multidimensional indexing method with good guaranteed performance. *ACM Trans. On Database Systems*, vol. 15, n. 44, pp.625-658, 1990.

[14] F. McGuff. Data modeling for data warehouses. October 1996. http://members.aol.com/ fmcguff/dwmodel/dwmodel.htm.

[15] J. Widom. Research Problems in Data Warehousing. In *Proc. 4th Int. Conf. on Information and Knowledge Management*, Nov. 1995.

[16] Y. Zhuge, H. Garcia-Molina, J. L. Wiener. The Strobe Algorithms for Multi-Source Warehouse Consistency. In *Proc. Conference on Parallel and Distributed Information Systems*, Miami Beach, FL, 1996.