Informatica per Statistica
Riassunto della lezione del 06/12/2013

Igor Melatti

Introduzione alla progettazione logica di basi di dati

- Questo riassunto è da intendersi come un commento alle slide `BD2002-02.PDF`
  - non tutte le slide sono commentate
  - quelle che possono essere saltate sono esplicitamente indicate
- Slide 4: quello che cambia tra il concetto matematico di relazione, e il concetto di relazione del modello relazionale, è il fatto che quest'ultimo non è posizionale
- Slide 5: oltre alla relazione matematica e a quella del modello logico, esistono anche le relazioni del diagramma E/R
- Slides 6-8: più sistematicamente:
  - siano $D_1, \ldots, D_n$ dei *domini*, ovvero insiemi non vuoti di possibili valori
  - i domini non devono essere necessariamente distinti; potrebbero anche essere tutti uguali
  - è possibile definire l'insieme del *prodotto cartesiano* $D = \prod_{i=1}^{n} D_i = D_1 \times \ldots \times D_n$ come l'insieme delle tutte le possibili n-uple ordinate costruite sui domini $D_1, \ldots, D_n$
  - ovvero, ogni n-upla è del tipo $(d_1, \ldots, d_n)$, in modo tale che $d_1 \in D_1, \ldots, d_n \in D_n$, o più sinteticamente $\forall i \in [1, n].\ d_i \in D_i$
  - quindi $D = D_1 \times \ldots \times D_n = \{(d_1, \ldots, d_n) \mid \forall i \in [1, n].\ d_i \in D_i\}$
  - da notare che le varie n-uple dentro il prodotto cartesiano D non sono ordinate
  - invece, ogni singola n-upla è al suo interno ordinata
  - quindi, se $D_1 = \{A, B\}$ e $D_2 = \{23\}$, allora $D_1 \times D_2 = \{(A, 23), (B, 23)\} = \{(B, 23), (A, 23)\}$

  - invece, $(A, 23) \neq (23, A)$
  - un qualsiasi sottoinsieme (anche vuoto) del prodotto cartesiano $D_1 \times \ldots \times D_n$ è una *relazione matematica* tra i domini $D_1, \ldots, D_n$
  - quindi una relazione R sui domini è tale che $R \subseteq D_1 \times \ldots \times D_n$

- Slide 10: dalla relazione matematica (posizionale) a quello del modello logico dei dati (non posizionale)
  - basta aggiungere le intestazioni, ovvero un "nome" (esplicativo) da associare a ciascuna colonna
  - occorre stare attenti a definire tutte intestazioni diverse, anche per le colonne che hanno lo stesso dominio
  - ora si possono anche scambiare le colonne tra loro
  - a patto di portarsi dietro anche le intestazioni delle colonne stesse
  - quindi scambiare l'intera prima colonna con l'intera seconda colonna è ok, si ottiene cioè la stessa relazione (logica)
  - se invece si scambia la prima colonna con la seconda, ma lasciando inalterata la prima riga (intestazione), la relazione che si ottiene è diversa
  - nella relazione matematica, spostare le "colonne" (ovvero, scambiare tra loro gli elementi di tutte le n-upla) non è possibile
    * o meglio, lo si può anche fare, ma si ottiene in generale una relazione *diversa* da quella di partenza

- Slide 11: le relazioni del modello relazionale sono facilmente rappresentabili con *tabelle*
  - ovviamente anche le relazioni matematiche sono rappresentabili con tabelle, ma senza intestazioni
  - nel caso delle relazioni matematiche, si possono scambiare righe (l'ordinamento tra n-uple non c'è) ma non le colonne (l'ordinamento all'interno delle singole n-uple c'è)
  - invece, tutte le relazioni del modello relazionale possono essere rappresentate con tabelle intestate
  - si possono scambiare sia le righe che le colonne (a patto di scambiare anche le intestazioni)
  - non tutte le tabelle (intestate) rappresentano una relazione
    * una tabella intestata è semplicemente una lista di righe e colonne, senza alcun vincolo
    * potrebbe quindi avere righe uguali (vietato nelle relazioni matematiche e quindi anche in quelle logiche)
    * o potrebbe avere colonne con intestazioni uguali

    - o potrebbe avere colonne con valori non tratti dallo stesso dominio (un numero in una riga, una stringa in un'altra)
    - se nessuna di queste tre condizioni si avvera, allora la tabella può essere vista come la rappresentazione di una relazione
  - un DBMS (e quindi la base di dati che esso gestisce) si dice *relazionale* se il suo modello logico è quello relazionale
  - un DBMS relazionale (e la base di dati che esso gestisce) può essere visto come un insieme di tabelle

- Slides 12-14: spesso le tabelle di una base di dati sono connesse tra loro, nel senso che i dati di una tabella fanno riferimento a dati contenuti in un'altra
  - questi riferimenti si fanno per valore
  - nella tabella Esami si può mettere solo la matricola dello studente, tanto tutte le informazioni sono nella tabella studente
  - quindi ciascuna riga della tabella esami fa riferimento ad una qualche riga della tabella studenti
  - in particolare, le prime due righe della tabella esami fanno riferimento alla prima riga della tabella studenti
  - per esplicitare questo legame, il modello relazionale si limita a piazzare lo stesso dato (3456 nell'esempio) in entrambe le tabelle
  - altri modelli logici dei dati fanno scelte diverse, questo è l'aspetto saliente del modello relazionale

- Slide 15: saltare

- Slide 16: definizioni formali di schema di relazione e di base di dati
  - per rappresentare lo schema di una relazione (e quindi una tabella) è sufficiente il nome della tabella e delle intestazioni di tutte le sue colonne (*attributi*, diversi ma collegati agli attributi degli schemi E/R)
  - la convenzione è dare il nome della tabella e tra parentesi i nomi dei suoi attributi
  - ad essere precisi occorre dire anche quali sono i domini, ma quest'informazione viene data in annotazioni a parte
  - riprendendo l'esempio nelle slides 9-10, si ha:
  - siano $D_1 = D_2 = \{p \mid p$ è una sequenza (finita) di lettere dell'alfabeto$\}$ (ovvero, D_1 e D_2 sono gli insiemi di tutte le possibili stringhe) e $D_3 = D_4 = \mathbb{N}$, allora la relazione matematica corrispondente è $D_1 \times D_2 \times D_3 \times D_4 \supseteq R = \{$(Juve, Lazio, 3, 1), (Lazio, Milan, 2, 0), (Juve, Roma, 0, 2), (Roma, Milan, 0, 1)$\}$

  - lo schema della (o meglio, di una) corrispondente relazione nel modello logico è Partita(SquadraInCasa, SquadraFuoriCasa, GolInCasa, GolFuoriCasa)
  - lo schema della base di dati, in questo esempio, è Campionato = {Partita(SquadraInCasa, SquadraFuoriCasa, GolInCasa, GolFuoriCasa)}
  - supponendo che nella base di dati Campionato ci sia un'altra tabella con i dati delle squadre, allora si avrebbe uno schema di basi di dati simile a questo: Campionato = {Partita(SquadraInCasa, SquadraFuoriCasa, GolInCasa, GolFuoriCasa), Squadra(Nome, ColoriSociali, Città, NomeStadio)}

- Slide 17: riprendendo l'esempio dato appena sopra, si ha quanto segue
  - si dia un nome a ciascuna n-upla logica (normalmente, le n-uple logiche sono chiamate *tuple*): t_1 = (Juve, Lazio, 3, 1), t_2 = (Lazio, Milan, 2, 0), t_3 = (Juve, Roma, 0, 2), t_4 = (Roma, Milan, 0, 1)
  - si avrà quindi t_1[SquadraInCasa] = Juve, t_4[SquadraFuoriCasa] = Milan, t_3[GolInCasa] = 0, etc

- Slide 18: le istanze, ovvero i contenuti (n-uple o meglio tuple) delle relazioni; riprendendo ancora l'esempio di cui sopra si ha:
  - l'istanza della relazione logica Partita presentata sopra è descritta matematicamente come $\{t_1, t_2, t_3, t_4\}$
  - l'istanza della base di dati Campionato sarà $\{\{t_1, t_2, t_3, t_4\}, \{t_5, t_6, t_7, t_8\}\}$, dove ad esempio t_5[Nome] = Juve, t_5[ColoriSociali] = Bianco_e_Nero, t_5[Città] = Torino, t_5[NomeStadio] = Olimpico
  - **esercizio**: descrivere possibili valori per t_6, t_7, t_8, sulla base degli esempi appena discussi

- Slide 19-23: saltare

- Slide 26: per esempio, si supponga di avere un attributo Età per una tabella Persona; il valore effettivo per questo attributo potrebbe non essere sempre noto nel momento in cui si caricano i valori nella base di dati
  - mettere un valore "non utilizzato" potrebbe voler dire mettere ad esempio 150, tanto nessuno vive così tanto
  - quindi quando il dato sull'età non è conosciuto, si mette 150
  - e se poi la medicina compie un improvviso balzo da gigante?

- Slide 27: tutti i domini delle basi di dati sono considerati estesi dal valore speciale NULL

  - è garantito che sia *diverso* da tutti gli altri elementi del dominio (ci pensa il DBMS)
  - quindi, se il dominio è quello delle stringhe, 'NULL' (la stringa composta dalle lettere N, U, L, L) è un valore diverso da NULL
  - infatti, i linguaggi di interrogazione (come l'SQL) distinguono tra i due casi, grazie alle virgolette ('NULL' è una stringa, NULL è il valore NULL)
  - in ogni caso, occorre dire che alcuni attributi non possono contenere valori NULL

- Slide 29, ovvero:

  - valore sconosciuto: il valore corrispondente esiste, ma non lo si conosce (ad esempio, l'Età per le persone, tutte ne hanno una, ma potrebbe non essere nota in alcuni casi)
  - valore inesistente: il valore corrispondente non esiste nel caso in esame (ad esempio, un attributo SecondoNome può non esistere in alcuni casi, in quanto non tutte le persone hanno un secondo nome)
  - valore senza informazione: non si sa se esiste oppure no, ed ovviamente, nel caso esistesse, non se ne conoscerebbe il valore (ad esempio, un attributo SecondoNome in alcuni casi può essere escluso, ovvero si sa che non c'è, e siamo nel caso immediatamente precedente, oppure si potrebbe non sapere se la specifica persona in questione ha o no un secondo nome, e allora siamo in questo caso del valore senza informazione)

- Slide 32: data un'istanza di una base di dati (quindi *tutte* le tuple di *tutte* le relazioni), occorre che essa sia consistente rispetto ad alcuni *vincoli*

  - ovvero, si dà al progettista di basi di dati la possibilità di dire che proprietà deve avere l'istanza attuale della base di dati per essere corretta
  - ogni proprietà è una funzione booleana: se applicata all'attuale istanza, può ritornare o il valore vero (e allora l'istanza è ok) o il valore falso (e allora l'istanza ha qualcosa che non va, come gli esempi di slide 31)

- Slide 34: ad esempio "tutti gli studenti dell'Università devono essere inseriti nella tabella Studente"; si tratta di una proprietà che esula dai compiti del progettista

- Slide 35: gli intrarelazionali (detti anche vincoli di ennupla o tupla) guardano ogni singola tupla di una singola tabella separatamente, senza "incrociare" le informazioni; invece quelle interrelazionali (detti anche vincoli di integrità referenziale) guardano alle corrispondenze tra (attributi di) tabelle

- Slides 40-48
  - si parte da una tabella $R(A_1, \ldots, A_n)$, e si vuole trovare una chiave
  - si comincia col trovare una superchiave: un sottoinsieme degli attributi, ovvero un $K \subseteq \{A_1, \ldots, A_n\}$, che identifichi sempre una solo possibile tupla di valori di R
  - in formule, se $t_1 \neq t_2$ sono due tuple distinte di un'istanza di R, allora $t_1[K] \neq t_2[K]$
  - nell'esempio di slide 40, ci sono due proposte di superchiave: $K_1 =$ {Matricola} e $K_2 =$ {Cognome, Nome, Nascita}
  - lo si può vedere anche così: se si prendono due qualsiasi tuple (righe) e ne si guarda solo la Matricola, si ottengono sempre valori diversi ({27655, 78763, 65432, 87654, 67653})
  - ugualmente per K_2: si ottengono {(Rossi, Mario, 5/12/78), (Rossi, Mario, 3/11/76), (Neri, Piero, 10/7/79), (Neri, Mario, 3/11/76), (Rossi, Piero, 5/12/78)}
  - invece, per $K_3 =$ {Cognome, Nascita} si otterrebbe {(Rossi, 5/12/78), (Rossi, 3/11/76), (Neri, 10/7/79), (Neri, 3/11/76), (Rossi, 5/12/78)}, che ha una ripetizione (prima e ultima tupla), quindi K_3 non è superchiave
  - in realtà, questo lavoro non va fatto su un'istanza in particolare come è stato fatto qui, ma occorre farlo in generale, ovvero immaginando i casi possibili che si possono presentare
  - ad esempio, è possibile che si presenti il caso in cui due studenti diversi abbiano uguale nome, cognome e data di nascita, quindi in realtà K_2 non è una superchiave (lo è "per caso", come si dice a slide 45 e 47)
  - individuata una superchiave K, occorre vedere se è minimale, ovvero se togliendo un qualche attributo a K si ottiene comunque una superchiave
  - se non si può togliere nessun attributo, allora K è chiave
  - ovviamente, se una superchiave K ha un solo attributo, allora è chiave
  - questo ovviamente non toglie che esistano anche chiavi con più attributi
  - esiste sempre una superchiave: basta prendere *tutti* gli attributi, e per definizione di relazione logica tutte le tuple saranno diverse...
    - precisazione importante: può succedere che una superchiave non esista, e quindi tipicamente si aggiunge un attributo "codice" o simile
    - questo non contraddice il fatto che esiste sempre una superchiave

    - infatti, se la superchiave non esiste, allora a rigor di definizione la relazione per come la si era definita non era effettivamente una relazione
    - cioè, non soddisfava la definizione di relazione del modello logico dei dati, che prevede appunto che le righe (tuple) siano tutte diverse tra loro
    - una volta aggiunto l'attributo "codice" o simile, si ottiene finalmente una relazione logica con tutti i crismi
  - per una stessa tabella, possono esistere più superchiavi e più chiavi

- Slides 49-52: non è realistico imporre che tutte le chiavi di una tabella non possano prendere il valore NULL
  - è però realistico, e lo si fa sempre, scegliere una particolare chiave, quella concettualmente più importante, definirla come *chiave primaria* e impedire che lì ci siano valori NULL

- Slides 53-55: l'integrità referenziale è un vincolo tra tabelle (quindi interrelazionale) che impone che un attributo (in generale un insieme di attributi) di una tabella abbia solo valori che siano già presenti in un altro (insieme di) attributi di un'altra tabella
  - conta quindi la "direzione" del vincolo
  - slide 54: tutto ok, ogni valore di Infrazioni.Vigile è contenuto in Vigili.Matricola (non ha importanza che il viceversa non valga, ovvero che 7543 non sia in Infrazioni.Vigile)
  - slide 55: due attributi, occorre che la coppia di valori sia presente
  - se per esempio la tabella infrazioni contenesse una tupla t t.c. t[Prov] = TO e t[Numero] = 839548, allora il vincolo non sarebbe più valido, anche se, presi separatamente, sia TO che 839548 sono presenti in Auto

- Slides 59-65: saltare