Informatica per Statistica
Riassunto della lezione dell'11/12/2013

Igor Melatti

Introduzione alla progettazione logica di basi di dati

- Questo riassunto è da intendersi come un commento alle slide `BD2002-08.PDF`
  - non tutte le slide sono commentate
  - quelle che possono essere saltate sono esplicitamente indicate
- Slide 4: progettazione logica, prende un diagramma E/R, lo ristruttura, e poi costruisce un modello logico corrispondente
  - il "modello logico" si riferisce al fatto che occorre sapere se si vuole un modello logico relazionale, o reticolare, o ad oggetti...
  - qui si suppone sempre che sia quello relazionale
  - il carico applicativo serve nella fase di ristrutturazione
  - lo "schema logico" in output è uno schema di una base di dati
  - la "documentazione associata" serve per definire i domini, i vincoli, e le interrogazioni
- Slide 5-6: prima di effettuare la traduzione vera e propria, occorre considerare gli aspetti prestazionali, e se necessario *ristrutturare* il diagramma E/R
  - qui per "ristrutturare" si intende modificare opportunamente il diagramma E/R per tener conto di aspetti prestazionali e non
  - quindi la fase di ristrutturazione prende in input un diagramma E/R (e informazioni aggiuntive sui carichi di lavoro previsti) e restituisce in output un nuovo diagramma E/R
- Slides 7-14: come, in generale, si valutano (o meglio, si stimano) le prestazioni su un diagramma E/R provvisto di tabelle dei volumi

- una tabella dei volumi (ad es., in slide 11) indica, per ogni entità e per ogni relazione del diagramma E/R, quale sarà il numero atteso di occorrenze
  - ovviamente, le relazioni dovranno avere un numero in accordo con le rispettive cardinalità
  - nell'esempio di slide 11, la cardinalità della relazione "Direzione" è necessariamente uguale a quella dell'entità "Dipartimento", dato che, dalla parte di "Dipartimento", "Direzione" ha cardinalità (1, 1)
  - la tabella fornisce indirettamente la stima del rapporto delle occorrenze delle entità coinvolte in una relazione
  - ad esempio, in media 80 impiegati su 2000 sono direttori di dipartimento, in accordo con la cardinalità (0, 1) della relazione "Direzione" dalla parte di "Impiegato" (il che vuole dire che solo un sottoinsieme proprio degli impiegati sono direttori...)
  - come ulteriore esempio, in media si hanno 3 progetti per ogni impiegato, e 12 impiegati per ogni progetto
  - non occorre dare i volumi degli attributi (che sono gli stessi dell'entità o relazione corrispondente), o di altri costrutti dello schema E/R
- questa valutazione viene fatta nel momento in cui è necessario fare una qualche scelta all'interno del processo di ristrutturazione
  - esempi: c'è un ridondanza, conviene mantenerla o eliminarla?
  - c'è una generalizzazione, come conviene eliminarla?
- in generale, in questi casi sono possibili almeno due scelte
- per entrambe, si stimano le prestazioni in termini di spazio (ovvero, di memoria occupata) e di tempo (necessario alle operazioni coinvolte nella parte di diagramma E/R che si sta ristrutturando)
- siccome qui si ha a disposizione la sola tabella dei volumi, che parla di occorrenze per entità e relazioni, entrambi questi aspetti sono tradotti in termini di numero di occorrenze
- quindi, per valutare lo spazio di un diagramma E/R, si considerano le occorrenze totali corrispondenti
  - non è difficile risalire da questo alla quantità di memoria (in termini di bytes) effettivamente usata
- mentre, per valutare le prestazioni, si prendono le operazioni che la base di dati deve offrire (e che saranno, alla fine della progettazione logica, tradotte in *query* SQL)
- si considerano le entità e le relazioni che dovranno essere necessariamente considerate per eseguire una data operazione
- si costruisce una tabella degli accessi, che fornisce la stima delle prestazioni richieste

  - per un esempio più concreto, vedere più avanti (slides 22–27)

- Slide 16: una ridondanza è un concetto (un attributo o una relazione) che possono essere eliminati senza perdita di informazione
  - questo perché la stessa informazione è contenuta altrove, oppure è ricavabile tramite calcoli
- Slide 17: ovviamente, mantenere una ridondanza è uno "spreco" di spazio; può però valerne la pena, se semplifica alcune interrogazioni
  - occorre vedere caso per caso cosa succede alle prestazioni sia mantenendo che eliminando le ridondanze
- Slide 18: le ridondanze possono riguardare solo attributi o relazioni, difficilmente entità
- Slides 22-27: analisi di una ridondanza
  - se si eliminasse l'attributo NumeroAbitanti non si perderebbe informazione: per sapere il numero di abitanti di una città basterebbe contare quante volte quella città è coinvolta in istanze della relazione Residenza
  - quindi NumeroAbitanti è ridondante
  - per decidere se tenerlo oppure no, si procede come segue
  - prima di tutto, si determina, sulla base della tabella dei volumi (o carichi), quanta memoria si risparmierebbe eliminando la ridondanza
    * o equivalentemente, quanta memoria si sprecherebbe mantenendo la ridondanza
    * nell'esempio dato: assumendo che ogni attributo NumeroAbitanti occupi i 4 bytes standard per rappresentare un intero, si ha che il risparmio di memoria è di $200 \times 4 = 800$ bytes (che ci siano 200 città lo si può vedere dalla tabella dei volumi di slide 23)
  - dopodiché, si prendono tutte le operazioni che coinvolgono NumeroAbitanti
    * nell'esempio dato: le due operazioni di slide 23
    * per ogni operazione occorre saper valutare che tipo di accessi richiede sui concetti coinvolti
    * un'operazione che richieda di inserire un nuovo elemento di un dato concetto va valutata come un'operazione di *scrittura* ("S" nelle tabelle nelle slides 24 e 25)
    * un'operazione che richieda semplicemente di restituire dei dati senza modificarli in alcun modo (interrogazione semplice) va valutata come un'operazione di *lettura* ("L' nelle tabelle nelle slides 24 e 25)

    - un'operazione che debba modificare dei dati di un elemento già presente in un concetto richiede 2 accessi, uno in lettura (per trovare l'elemento da modificare tra tutti quelli presenti) e uno in scrittura (per modificarlo opportunamente)

- ora è possibile valutare:
    1. quanto "costano" queste operazioni mantenendo la ridondanza?
        - slide 24: per realizzare l'operazione 1 con la ridondanza, occorre accedere 1 volta in scrittura a Persona (per inserire la persona), 1 volta in scrittura a Residenza (per associarla con la città di residenza), 1 volta in lettura a Città (per trovare la città di residenza tra quelli già presenti) e 1 in scrittura a Città (per modificare il numero di abitanti, che va incrementato di uno)
        - slide 26: costo dell'operazione 1, 3 accessi in scrittura e 1 in lettura per ogni operazione; essendoci 500 chiamate all'operazione 1 al giorno, si hanno 1500 accessi in scrittura e 500 in lettura al giorno
        - slide 24: per realizzare l'operazione 2 con la ridondanza, occorre accedere 1 volta in lettura a Città (per trovare la città richiesta e stamparne tutti gli attributi)
        - slide 26: costo dell'operazione 2, 1 accesso in lettura per ogni operazione; essendoci 2 chiamate all'operazione 2 al giorno, si hanno 2 accessi in lettura al giorno
        - per farne un numero unico, dato che gli accessi in scrittura sono più difficili da realizzare per i DBMS, si contano doppi, e quindi si ha che il costo combinato dell'operazione 1 e 2 è di 3502 accessi al giorno
        - quindi l'operazione 2 è trascurabile rispetto all'operazione 1
    2. quanto "costano" queste operazioni eliminando la ridondanza?
        - slide 25: per realizzare l'operazione 1 senza la ridondanza, occorre accedere 1 volta in scrittura a Persona (per inserire la persona) e 1 volta in scrittura a Residenza (per associarla con la città di residenza); non è più necessario modificare la città nell'entità Città, in quanto non c'è più il numero di abitanti
        - slide 27: costo dell'operazione 1, 2 accessi in scrittura per ogni operazione; essendoci 500 chiamate all'operazione 1 al giorno, si hanno 1000 accessi in scrittura al giorno
        - slide 25: per realizzare l'operazione 2 senza la ridondanza, occorre accedere 1 volta in lettura a Città (per trovare la città richiesta e stamparne tutti gli attributi, che però non comprendono il numero di abitanti richiesto dall'operazione 2), e accedere a Residenza tante volte quante sono le associazioni con la città richiesta ivi presenti (se ci sono 200 città e

1000000 di persone come risulta dalla slide 23, allora in media ci saranno 1000000/200 = 5000 abitanti per città, e questo valore viene preso come il numero di accessi necessario)

    - slide 27: costo dell'operazione 2, 5001 accessi in lettura per ogni operazione; essendoci 2 chiamate all'operazione 2 al giorno, si hanno 10002 accessi al giorno
    - per farne un numero unico (contando nuovamente doppi gli accessi in scrittura), il costo combinato dell'operazione 1 e 2 è di 12002 accessi al giorno
  - dopodiché si può finalmente valutare se il risparmio di memoria è giustificato dal (probabile) peggioramento delle operazioni
    - qui, per risparmiare 800 bytes (meno di un KB) si fanno 8500 accessi in più al giorno
    - non ne vale la pena

- Slide 32: contrariamente da quanto possa sembrare, A11, A12 e TIPO *non* sono identificatori primari dell'entità E0; sono colorati solo per mostrare che sono cambiati da prima a dopo la ristrutturazione
- Slide 34: qui A01 è invece identificatore primario sia per E1 che per E2
- Slide 38: i numeri 1, 2, e 3 si riferiscono agli stessi della slide 30
- Slide 39-40: saltare
- Slides 44-45: partizionamento verticale di entità
- Slides 46-47: eliminazione di attributi multivalore
- Slides 48-49: accorpamento di entità e relazioni in un'unica entità
- Slides 50-51: partizionamento orizzontale di una relazione
- Slides 55-seguenti: finita la ristrutturazione, si può scrivere lo schema della base di dati
  - oltre che allo schema, occorre scrivere le annotazioni, ovvero domini, vincoli etc etc
  - negli esempio di queste slides, le annotazioni sono date per scontate
- Slide 55:
  - è praticamente sempre vero che un'entità diventa una relazione logica avente gli stessi attributi dell'entità di stessa
  - fa eccezione il caso (raro) di entità collegate da una relazione 1-1
  - per le relazioni concettuali occorre vedere caso per caso: possono diventare relazioni logiche a sé stanti o essere "inglobate" in altri relazioni

- Slides 56-58: relazioni molti a molti
  - sono quelle con la traduzione più "lineare": una relazione logica per ciascuno dei tre concetti (2 entità e una relazione) coinvolti
  - le chiavi sono ricavate da quelle delle entità
  - i nomi degli attributi della relazione logica corrispondente alla relazione concettuale possono essere cambiati (ad es. Impiegato invece di Matricola in Partecipazione)
  - il cambio dei nomi per le relazioni logiche corrispondenti a relazioni concettuali è molto comune (vedere anche gli esempi nelle slides seguenti)
- Slides 61-62: relazioni uno a molti
  - si potrebbe fare come per le relazioni molti a molti (slide 61), ma sarebbe inutilmente ridondante
  - infatti, dato che per ogni giocatore c'è una sola squadra, è inutile ripetere la chiave di quel giocatore in un'altra tabella: tanto vale rimanere sulla tabella dei giocatori ed aggiungere la squadra e l'ingaggio
  - se la relazione fosse stata molti a molti non si sarebbe potuto fare: occorrerebbe ripetere su righe diverse lo stesso giocatore (per assegnargli le diverse squadre), ma questo vorrebbe dire ripetere la chiave su righe diverse...
- Slide 64: ci sono quindi quattro possibilità:
  1. due relazioni Direttore e Dipartimento, e Direttore contiene anche DataInizio e la chiave di Dipartimento (Direttore incorpora Direzione)
  2. due relazioni Direttore e Dipartimento, e Dipartimento contiene anche DataInizio e la chiave di Direttore (Dipartimento incorpora Direzione)
  3. una sola relazione Direttore che contiene anche tutti gli attributi di Dipartimento e DataInizio (Direttore incorpora Direzione e Dipartimento)
  4. una sola relazione Dipartimento che contiene anche tutti gli attributi di Direttore e DataInizio (Dipartimento incorpora Direzione e Direttore)
- Slide 65: errore nello schema E/R, "Direttore" è in realtà "Impiegato"
  1. in questo caso si preferisce incorporare Direzione in Dipartimento, così da minimizzare il numero di NULL (ovvero di memoria sprecata)

- Slide 66: qui si possono anche mettere 3 relazioni separate, una per ogni concetto
- Slides 69-71: saltare