Data integration

Materiale Originale A/A 2014/2015 *Davide Mottin*

Aggiornato per A/A 2016/2017 *Matteo Lissandrini*

Ultimo Aggiornamento: 07.06.2017

Questa guida non sostituisce il materiale del corso, ma ne integra il contenuto cercando di aggiungere dettagli e chiarimenti laddove più necessario

- Data integration
  - Data integration
    - Federated databases
    - Data warehouse
    - Mediatori
      - Adornments
    - Esercizi possibili
  - Entity resolution
    - Algoritmo per entity resolution
      - Operazioni di Merge

L'importanza di gestire, integrare e analizzare i dati è vitale in molte applicazioni. In questo capitolo verranno introdotti i seguenti argomenti:

- Integrazione di basi di dati
- Entity Resolutions

Data integration

> **Problema**: abbiamo diverse basi di dati eterogenee che contengono cose simili ma con schemi diversi, tipi di dato diversi, rappresentazioni diverse. Come troviamo le informazioni da cercare in maniera trasparente?

Assumi di avere tre database, tutti riguardanti automobili e i loro optional, ma che immagazzinano dati in modo (strutture) diverse

```
# DB1:
NeededCars (model, color, autoTrans)
--- (autoTrans è un booleano)

# DB2:
Autos (serial, model, color)
Options (serial, option)


#DB3:
Cars (serialNo, model, color, autoTrans, ...)
--- Cars contiene ogni possibile optional come attributo
```

Assumiamo che i dati abbiano lo stesso schema, ma valori leggermente diversi negli attributi oppure che abbiamo modelli diversi ma che comunque rappresentino gli stessi identici oggetti.

Esistono tre approcci alternativi:

1. **Federated databases**: database indipendenti, ma ognuno in grado di chiamare/spedire informazioni all'altro.
2. **Warehousing**: integrazione dei dati in un unico grande database (esempio, OLAP).

3. **Mediation**: crea un'interfaccia comune di accesso ai vari dati e converte le query per ognuno dei database.

FEDERATED DATABASES

Supponiamo di avere n database con informazioni diverse e indipendenti. Un approccio potrebbe essere che ogni database effettua una query su ogni altro database.

> **Problema**: Approccio troppo dispendioso dato il numero di query da fare, ossia tutte le possibili coppie: $\mathcal{O}(n^2)$
>
> **Soluzione**: Scrivere una query diversa per ogni database.
>
> **Problema**: Difficile costruire una query del genere.

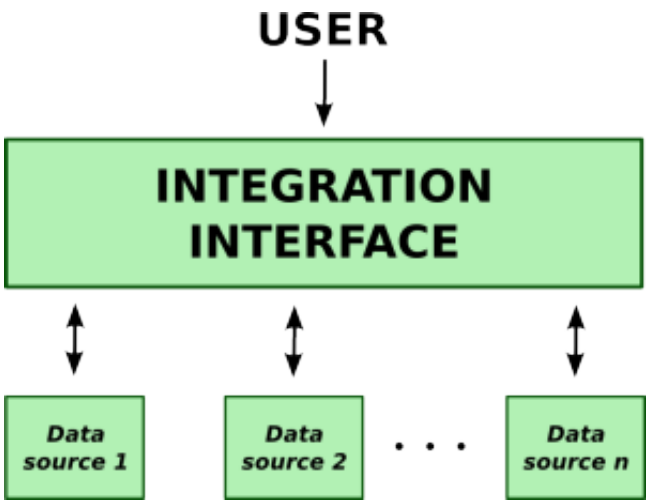
DATA WAREHOUSE

- Un'unico grande database che combina i dati di n database indipendenti.
- I dati vengono integrati e aggiornati periodicamente nella warehouse.
- Vengono utilizzate soprattutto per calcolare aggregati e per questo non devono essere necessariamente aggiornati in tempo reale.
- Vantaggio: dobbiamo scrivere una sola query.

> **Problema**: Come mappare tanti dati in un'unico database? Come mantenere i dati aggiornati?
>
> **Soluzione**: Integrare i dati in maniera manuale producendo tabelle e mediatori per ogni database, in modo tale da tradurre i dati in un'unica tabella (si ricorda che i campi dei singoli database possono avere nomi diversi).
>
> **Attenzione**: mentre inserimenti e interrogazioni alla data warehouse sono abbastanza automatici, la cancellazione è più complicata, in quanto cancellare un dato nella tabella integrata potrebbe voler dire cancellarlo in multiple tabelle (se è un dato replicato) oppure in una sola. Il problema è che se il dato è più complesso non si può sapere se si debba preservare o no in un database.

MEDIATORI

- I mediatori sono delle viste virtuali (`VIEW` in SQL) di un insieme di database.
- I mediatori non memorizzano nessun dato.
- Ogni database ha un "wrapper" o un mapping tra campi mediati e campi nel singolo database, in modo da "tradurre" le query nei diversi database.

> *Global as View*: la Warehouse presenta un interfaccia, ma non immagazzina nessun dato.
>
> 
>

Quindi il mediatore deve decidere uno schema da presentare all'utente (quante tabelle, quali attributi, di che tipo?).

In questo modo: l'utente scrive una query al mediatore, il mediatore inoltra la query ad uno o più wrapper (per ogni database necessario), ogni wrapper ritorna la risposta e il mediatore combina multiple risposte.

Per tradurre una query utente in una query per un specifico database, si possono usare *template* per wrapper (si ricorda che un wrapper è un pezzo di software che effettua la traduzione degli attributi).

Ecco un esempio dove `AutosMed` è lo schema del mediatore e la query trova tutte le auto rosse:

```
SELECT *
FROM AutosMed
WHERE color = 'red';
      =>
SELECT serialNo, model, color, autoTrans, "dealer1"
FROM Cars
WHERE color = 'red';
```

> **Problema**: le query possono diventare molto complesse ed in molti casi è impossibile trovare un mapping.
>
> **Soluzione**: restringere il problema a determinate classi di query e costruire un *wrapper generator*.

Un *wrapper generator* dato un templated e dei parametri genera il wrapper necessario.

```
SELECT *
FROM AutosMed
WHERE color = $c;
      =>
SELECT serialNo, model, color, autoTrans, "dealer1"
FROM Cars
WHERE color = $c;
```

> **Problema**: non è possibile costruire un template per ogni query. Si immagini di volere macchine di un certo colore e di un certo modello (condizioni in AND su attributi).
>
> **Soluzione**: creare dei template meno rigidi che restituiscano meno risultati (per esempio solo vincolati al colore), per poi restringere la query in un secondo momento.

Quindi se vogliamo una query su modello **e** colore, usiamo il wrapper che abbiamo su colore, e poi scriviamo una seconda query che usa il risultato della prima, e filtra in base al modello.

> **Problema**: Alcuni database hanno (o necessitano) restrizioni ai tipi di query che mettono a disposizione. Non è sempre possibile scrivere una query generica che ottenga ogni tupla, ogni vlore o ogni attibuto (privacy, costo computazionale o di rete).
>
> **Soluzione**: definire un modello che descriva quali query si possano fare o sono vietate → *adornments*

ADORNMENTS

È una sorta di linguaggio che specifica le possibilità su un database.

- `f` (free): l'attributo può essere specificato.
- `b` (bound): dobbiamo specificare un valore per l'attributo.
- `u` (unspecified): l'attributo non può essere specificato.
- `c[S]` (choice from set S): un valore deve essere specificato entro l'insieme S.
- `o[S]` (optional, from S): se specificato deve avere un valore in S, può essere omess.
- `f'` : l'attributo non può essere parte dell'output.

L'insieme degli adornments definisce le *capabilities specification*.

```
Cars(serialNo, model, color, autoTrans, navi)
Cars(u b b o[yes,no] o[yes,no])

-- Sono specificati modello e colore e, opzionalmente, trasmissione automatica e navigatore
```

Il mediatore semplicemente analizza le specifiche e decide la miglior strategia (ha una sorta di cost model interno).

ESERCIZI POSSIBILI

1. Scrivere una soluzione di integrazione per diversi database (e.g. 3 database).
2. Scrivere adornments per un particolare sistema.
3. Dato un'insieme di specifiche con adornments verificare se possono rispondere a determinate query o costruire un plan di esecuzione di una query.

Entity resolution

Molte volte alla stessa entità (oggetto, persona, stato …) ci si può riferire in modi diversi. Questo problema si verifica spesso in database diversi e ancor più spesso nel web a causa del linguaggio naturale.

> **Problema**: come capire che due espressioni si riferiscono alla stessa entità?
>
> **Soluzione**: Edit distance: approssimare le stringhe con la stringa più "vicina", cioè quella che è ottenuta trasformando (inserendo, cancellando o modificando) il minor numero di caratteri possibili.

Attraverso l'edit distance è possibile definire una somiglianza tra stringhe: le stringhe che hanno una distanza minore di τ fissato sono considerate simili.

> ⓘ **Attenzione**: la similarità non è transitiva.

ALGORITMO PER ENTITY RESOLUTION

⌖ **Problema**: dato un database, unire (merge) tuple simili.

ϟ **Soluzione**: un algoritmo che unisca tuple simili, ma solo nel caso che l'operazione di merge rispetti alcune proprietà.

Sia $\oplus$ un'operazione di merge che rispetta le seguenti proprietà:

- $t \oplus t = t$ (idenpotenza).
- $t \oplus s = s \oplus t$ (commutatività).
- $(r \oplus s) \oplus t = r \oplus (s \oplus t)$ (transitività).

mentre la relazione di similarità $\approx$ deve essere tale che:

- $r \approx r$ (riflessività).
- $r \approx s$ iff $s \approx r$ (simmetria).
- if $r \approx s$ then $r \approx (s \oplus t)$ (rappresentabilità).

Input: un insieme I di record, similarità $\approx$, una funzione di merge $\oplus$.

Ouptut: un insieme $O \subseteq I$.

Procedura: L'algoritmo parte dall'insieme dei risultati, inizialmente vuoto, e cerca di aggiungere tuple dal database che non sono simili a nessuno dei risultati già aggiunti. Questo è lo pseudocodice:

```
O := emptyset;
WHILE I is not empty DO BEGIN
   let r be any record in I;
   find, if possible, some record s in O that is similar to r;
   IF no record s exists THEN
      move r from I to O
   ELSE BEGIN
      delete r from I;
      delete s from O;
      add the merger of r and s to I;
END; END;
```

OPERAZIONI DI MERGE

Come visto prima, un primo passo richiede di decidere un metodo per riconoscere tuple simili.

> **Esempio:** Due tuple sono simili se hanno al massimo edit distance 2 per la colonna `Nome`
>
> | ID | Nome | Colore_preferito | Animale Preferito |
> |---|---|---|---|
> | 1 | Alice | Rosso | Gatto |
> | 2 | Bob | Rosso | Cane |
> | 3 | Carol | Blu | Coniglio |
> | 4 | Bilbo | Giallo | Cane |
> | 5 | Bobby | Verde | Cane |
> | 6 | Carl | Blu | Coniglio |
> | 7 | Alice | Rosso | Pappagallo |
>
> Quali tuple sono simili?
>
> ⓘ **Attenzione**: per utilizzare l'algoritmo bisogna dimostrare le proprietà per $\approx$ e $\oplus$.

Esistono due operazioni di merge molto semplici:

1. **MAKE NULL** data `e1` e `e2` , il merge è una tupla `e'` tale per cui, gli attributi identici vengono matenuti, mentre quelli diversi vengono messi a `null`
2. **SET-Value** dati `e1` e `e2` , il merge è una tupla *speciale* tale per cui, gli attributi identici vengono matenuti, mentre quelli diversi vengono in un **SET** (insieme) che include l'unione dei due.

> Calcola il risultato del merge di `Bob` , `Bilbo` e `Bobby` con entrambi i metodi. Calcola il risultato del merge di `Alice` , con l'altra tupla `Alice` con entrambi i metodi.
>
> **Problema**: se la funzione $\oplus$ non rispetta le proprietà?
>
> **Soluzione**: si definisce una *dominance relation* $r \leq s$ nella quale s contiene tutte le informazioni di r (contiene) e perciò si può ignorare r.