Appunti Database

Brendon Mendicino

October 12, 2022

Contents

1 Introduzione **3**

2 Data Warehouse **3**

3 Analisi **4**
3.1 Finestra di calcolo 5
3.2 Sintassi ORACLE 7
3.3 Esercizi 8

1 Introduzione

KDD: Knowledge Discovery from Data

Tecniche di data mining

- Regole di associazione: usate per trovare delle relazioni frequenti all'interno del database. Ad esempio: chi compra pannolini compra anche birra, il 2% degli scontrini contegono entrambe gli oggetti, il 30% degli scontrini che contengono pannalini contengono anche birra. Grazie alle regole di associazione si possono fare dei tipi di analisi come la basket analisys, ma puo essere utile anche per le raccomanadazioni.
- Classificazione: i classificatori predicono etichette discrete, esempio: nella posta elettronica alcune mail vengono segnate come spam. La classificazione definisce un modello per definire le predizioni, a volte non è sempre possibile creare dei modelli interpretabili ovvero dare una ragione per una determinata scelta.
- Clustering: gli algoritmi creano dei gruppi che raggruppano gli oggetti in esame, senza però dare delle motivazioni delle scelte effettuate.

2 Data Warehouse

Un DW è una base dati di supporto alla decisioni, che è mantenuta separatamente dalla base di dati operativa dell'azienda. I dati al suo interno sono:

- orientati ai soggetti di interesse;
- integrati e consistenti;
- dipendenti dal tempo;
- non volatili;
- utilizzati per il supporto alle decisioni aziendali;

Per la progettazione concettuale di un DW, non esiste un formalismo universale, il modello ER non è adatto ma viene invece utilizzata il modello **Dimensional Fact Model**.

Il DFM è composto da:

- Fatto: modella un insieme di eventi di interesse, che evolvono nel tempo (che può overe diversa granuralità).
- Dimensioni: sono gli attribuiti di un fatto, generalmento sono categorici.

- Misure: discrive una caratteristica numerica di un fatto.

Sulle dimensioni si possono definire delle gerarachie, che definiscono di fatto una dipendenza funzionale tra gli attributi, quindi di 1 a n. Ad esempio: **data** ha un arco **mese**, una data ha uno ed un solo mese (1 a n).

I costrutti avanazati sono:

- archi opzionali;
- dimensioni opzionali;
- attributo descrittivo: sono delle informazioni utili all'utente ma su cui non verteranno le interragazioni (ad esempio non si farà mai la group by su un indirizzo);
- non-additività: non si può fare la somma sulla metrica, il motivo è che non è modellato in modo tale da fare la somma;
- Fatto: fenomeno di studio;
- Misure: attributi del fatto;
- Dimensioni: tabelle collegate al fatto;

Schema a stella:

Snoflawke scheme: si esplicitano le dipendenze funzionali, questo però comporta un aumento delle operazioni di join.

Nella realtà lo snowflake è raramente utilizzato, il motivo è che il costo delle join può diventare oneroso. Un caso di utilizzo dello snowflake è quando si hanno dei dati condivisi.

Archi multipli:

Dimensioni degeneri: sono delle dimensioni con un solo attributo, questo si perchè nello stato attuale non si hanno delle specifiche per quell'attributo ma nel futuro si potrebbe facilmente estendere. Un'altra soluzione potrebbe essere un push down delle dimensioni degeri nella tabella dei fatti.

Junk Dimension: si può creare una dimensione che contenga tutte le dimensioni degeneri, le informazioni sono collegate semanticamente, è anche possibile unire delle informazioni scorrelate ma non è una scelta poco corretta, una soluzione potrebbe essere avere più junk dimensions.

3 Analisi

Sfruttando solo l'SQL è molto difficile fare delle analisi su un dw, infatti volendo calcolare delle operazioni per due argomenti diversi si devono fare più query. Estendendo il SQL si può, ad esempio, effettuare più operazioni leggendo una sola volta la tabella, ed effettuando il minor numero di join possibile.

Analisi OLAP I tipi di operazione sono:

- roll up: riducendo il livello di dettaglio del dato, ovvero eliminare una o più clausole della groupby o navigare la gerarchia verso l'esterno;
- drill down: si aumenta il livello di dettaglio oppure si aggiunge una dimensione di analisi;
- slice and dice: consentono di ridurre il volume dei dati selezionando un sottogruppo dei dati di partenza;
- tabelle pivot: come viene mostrato il dato;
- ordinamento: ordinamento in base agli attributi;

Queste operazioni possono essere fatte con più o una query.

3.1 Finestra di calcolo

Una finestra di calcolo fa dei calcoli a partire da una query sottostante, la finestra ha 3 operazioni sottostanti:

- partizionamento (**partition by**): partizionamnto dei dati, divide i record in gruppi a partire dall'attributo selezionato;
- ordinamento (**order by**): si definisce il criterio di ordinamento delle righe all'interno dei partizionamenti;
- finestra di aggregazione (**over**): porzione di dati, specifica per ogni riga di dato, su cui effettuare dei calcoli;

Example 3.1

Data la tabella Vendite(Città, Mese, Importo), calcolare per ogni città la media delle vendite per il mese corrente ed i due precedenti.

```
SELECT Citta, Mese, Importo,
    AVG(Importo) OVER (PARTITION BY Citta)
                        ORDER BY Mese
                        ROWS 2 PRECEDING)
    AS MediaMobile
FROM Vendite;
```

Quando la finestra è incompleta il calcolo è effettuato sulla parte presente, è possibile specificare che se la riga non è presente il risultato deve essere NULL.

Si può definire un intervallo fisico, superiore o inferiore.

```
    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
```

È possibile definire la tupla currente e quella che la precedono e che la seguono

```
ROWS UNBOUNDED PRECEDING (o FOLLOWING)
```

Il rggruppamento fisico è specifico per quando i dati non hanno delle interruzioni.

Per definire un intervallo logico si utilizza il costrutto **range**.

```
SELECT Citta, Mese, Importo,
    Importo / SUM(Importo) OVER () AS PerOverMax,
    Importo / SUM(Importo) OVER (PARTITION BY Citta) AS PerOverCity,
    Importo / SUM(Importo) OVER (PARTITION BY Mese) AS PerOverMonth
FROM Vendite
```

Se una **group by** è presente all'interno della query allora, tutte le entry che possono comparire nella finestra di calcolo sono solo quelle che compaiono nella group by.

Funzione di ranking La funzione di ranking serve a creare delle classifiche

- **rank()**: la funzione rank in presenza di più oggetti nella stessa posizione salta al prossimo record;
- **denserank()**: la funzione denserank tiene tutte righe con la stessa posizione;

...

```
SELECT Citta, Mese, SUM(Importo) AS TotMese,
    RANK() OVER (PARTITION BY Citta
                 ORDER BY SUM(Import) DESC)

FROM Vendite, ...
WHERE ...
GROUP BY Citta, Mese
```

Estensione della group by

- **rollup**: consente di calcolare le aggragazioni su tutti i possibili gruppi, eliminando una colonna alla volta, da destra verso sinistra, esempio: calcola le vendite per: (Citta, Mese, Prodotto), (Citta, Mese), (Citta):

  ```
  SELECT Citta, Mese, Prodotto, SUM(Importo) AS TotVendite
  FROM ...
  WHERE ...
  GROUP BY ROLLUP (Citta, Mese, Prodotto)

  ```

- **cube**: consente di calcolare tutte le possibili combinazioni del ragrruppamento;
- **grouping sets**: serve a definire degli aggregati su gruppi specifici definiti dall'utente;

3.2 Sintassi ORACLE

Raggruppamento fisico:

Example 3.2

Selezionare, separatamente per ogni città, per ogni data l'importo e la media dell'importo dei due giorni precedenti.

```
select citta, data, importo,
    avg(importo) over (partition by citta
                       order by data
                       rows 2 preceding
    ) as mediaMobile
from vendite
order by citta, data;

```

Raggruppamento logico:

Example 3.3

```
select citta, data, importo,
    avg(importo) over (PARTITION BY citta
                       ORDER BY data
                       RANGE BETWEEN INTERVAL '2'
                       DAY PRECEDING AND CURRENT ROW
    ) as mediaUltimi3Giorni
from vendite
order by citta, data;

```

Example 3.4

```
select COD_A, sum(Q) as sommaPerArticolo,
    rank() over (order by sum(Q) desc) as graduatoria
from FAP
group by COD_A

```

All'interno di oracle sono preseti delle funzionalità aggiuntive oltre alla funzione di rank, come:

ROW_NUMBER Assegno un numero progressivo ad ogni elemento in una partizione.

```
select tipo, peso,
    row_number over (partition by tipo
                     order by tipo)
```

```
from ...
where ...;
```

CUME_DIST Consente di calcolare le distribuzine cumulativa all'interno di una partizione, permette di definire un valore sulla distribuzione dei valori.

NTILE(n) Una funzione che da la possibilità di dividere le partizioni in sottogruppi

```
select tipo, perso,
    ntile(3) over (partition by tipo order by peso) as ntile3peso
from ...
where ...;
```

3.3 Esercizi

Cliente(CodCliente, Cliente, Provincia, Regione)
Categoria(CodCat, Categoria)
Agente(CodAgente, Agente, Agenzia)
Tempo(CodTempo, Mese, Trimestre, Semestre, Anno)
Fatturato(CodTempo, CodCliente, CodCatArticolo, CodAgente, TotFatturato, NumArticoli, TotSconto)

1. Visualizzare per ogni categoria di articoli

- la categoria
- la quantità totale fatturata per la categoria in esame
- il fatturato totale associato alla categoria in esame
- il rank della categoria in funzione della quantità totale fatturata
- il rank della categoria in funzione del fatturato totale

```
select categoria, sum(numArticoli),
    sum(totFatturato),
    rank() over (order by sum(numArticoli) desc),
    rank() over (order by sum(totFatturato) desc)
from fatturato f, categoria c
where f.codCatArticolo = c.codCat
group by categoria;
```

2. Visualizzare per ogni provincia

- la provincia

- la regione della provincia
- il fatturato totale associato alla provincia
- il rank della provincia in funzione del fatturato totale, separato per regione

```
select provincia, regione,
    sum(totFatturato) as fatturatoPerProvincia,
    rank() over (partion by regione
                 order by sum(totFatturato) desc
    ) as rankFatturatoPerRegione
from cliente c, fatturato f
where c.codCliente = f.codCliente
group by provincia, regione;
```

3. Visualizzare per ogni provincia e mese
   - la provincia
   - la regione della provincia
   - il mese
   - il fatturato totale associato alla provincia nel mese in esame
   - il rank della provincia in funzione del fatturato totale, separato per mese

```
select provincia, regione, mese,
    sum(totFatturato) as fatturatoPerProvinciaPerMese,
    rank() over (partition by mese
                 order by sum(totFatturato)
    ) as rankFatturatoPerMese
from cliente c, fatturato f, tempo t
where c.codCliente = f.codCliente and t.codTempo = f.codTempo
group by provincia, regione, mese;
```

4. Visualizzare per ogni regione e mese
   - la regione
   - il mese
   - il fatturato totale associato alla regione nel mese in esame
   - l'incasso cumulativo al trascorrere dei mesi, separato per ogni regione
   - l'incasso cumulativo al trascorrere dei mesi, separato per ogni anno e regione

```sql
select regione, mese,
    sum(TotFatturato) as fatturatoPerMese,
    sum(TotFatturato) over (
        partition by regione
        order by mese
    ) as incassoCumulativoTot,
    sum(TotFatturato) over (
        partition by regione, anno
        order by mese
    ) as incassoCumulativoPerAnno
from cliente c, fatturato f, tempo t
where c.CodCliente = f.CodCliente and t.CodTempo = f.CodTempo
group by regione, mese, anno;
```