

Appunti Database

Brendon Mendicino

October 19, 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
4	Viste materializzate	11
4.1	Documentazione Oracle	11
5	Progettazione fisica	13
6	Alimentazione dei Data Warehouse	14
6.1	Estrazione	14
6.2	Pulitura	14
6.3	Trasformazione	14
6.4	Caricamento	15

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 contengono entrambe gli oggetti, il 30% degli scontrini che contengono pannolini contengono anche birra. Grazie alle regole di associazione si possono fare dei tipi di analisi come la basket analysis, ma può essere utile anche per le raccomandazioni.
- 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 alle 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 avere diversa granularità).
- Dimensioni: sono gli attributi di un fatto, generalmente sono categorici.

- Misure: descrive una caratteristica numerica di un fatto.

Sulle dimensioni si possono definire delle gerarchie, 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 avanzati sono:

- archi opzionali;
- dimensioni opzionali;
- attributo descrittivo: sono delle informazioni utili all'utente ma su cui non verteranno le interrogazioni (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:

Snowflake 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 degeneri 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**): partizionamento 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.

```
1 SELECT Città, Mese, Importo,
2     AVG(Importo) OVER (PARTITION BY Città)
3                        ORDER BY Mese
4                        ROWS 2 PRECEDING)
5     AS MediaMobile
6 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.

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

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

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

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

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

```
1 SELECT Citta, Mese, Importo,
2        Importo / SUM(Importo) OVER () AS PerOverMax,
3        Importo / SUM(Importo) OVER (PARTITION BY Citta) AS PerOverCity,
4        Importo / SUM(Importo) OVER (PARTITION BY Mese) AS PerOverMonth
5 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;

...

```
1 SELECT Citta, Mese, SUM(Importo) AS TotMese,
2        RANK() OVER (PARTITION BY Citta
3                     ORDER BY SUM(Importo) DESC)
4
5 FROM Vendite, ...
6 WHERE ...
7 GROUP BY Citta, Mese
```

Estensione della group by

- **rollup**: consente di calcolare le aggregazioni 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):

```
1 SELECT Citta, Mese, Prodotto, SUM(Importo) AS TotVendite
2 FROM ...
3 WHERE ...
4 GROUP BY ROLLUP (Citta, Mese, Prodotto)
5
```

- **cube**: consente di calcolare tutte le possibili combinazioni del raggruppamento;
- **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.

```
1 select citta, data, importo,
2     avg(importo) over (partition by citta
3                        order by data
4                        rows 2 preceding
5                        ) as mediaMobile
6 from vendite
7 order by citta, data;
```

Raggruppamento logico:

Example 3.3

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

Example 3.4

```
1 select COD_A, sum(Q) as sommaPerArticolo,
2     rank() over (order by sum(Q) desc) as graduatoria
3 from FAP
4 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.

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

```
4 from ...
5 where ...;
```

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

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

```
1 select tipo, perso,
2     ntile(3) over (partition by tipo order by peso) as ntile3peso
3 from ...
4 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

```
1 select categoria, sum(numArticoli),
2     sum(totFatturato),
3     rank() over (order by sum(numArticoli) desc),
4     rank() over (order by sum(totFatturato) desc)
5 from fatturato f, categoria c
6 where f.codCatArticolo = c.codCat
7 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

```
1 select provincia, regione,
2     sum(totFatturato) as fatturatoPerProvincia,
3     rank() over (partition by regione
4                  order by sum(totFatturato) desc
5                  ) as rankFatturatoPerRegione
6 from cliente c, fatturato f
7 where c.codCliente = f.codCliente
8 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

```
1 select provincia, regione, mese,
2     sum(totFatturato) as fatturatoPerProvinciaPerMese,
3     rank() over (partition by mese
4                  order by sum(totFatturato)
5                  ) as rankFatturatoPerMese
6 from cliente c, fatturato f, tempo t
7 where c.codCliente = f.codCliente and t.codTempo = f.codTempo
8 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

```
1 select regione, mese,
2     sum(TotFatturato) as fatturatoPerMese,
3     sum(TotFatturato) over (
4         partition by regione
5         order by mese
6         rows unbounde preceding
7     ) as incassoCumulativoTot,
8     sum(TotFatturato) over (
9         partition by regione, anno
10        order by mese
11        rows unbounded preceding
12    ) as incassoCumulativoPerAnno
13 from cliente c, fatturato f, tempo t
14 where c.CodCliente = f.CodCliente and t.CodTempo = f.CodTempo
15 group by regione, mese, anno;
```

4 Viste materializzate

Le viste materializzate sono necessarie per ridurre la lentezza delle operazioni di group by per grandi moli di dati, le viste materializzate sono dei sommari precalcolati della tabella dei fatti.

Le VM usano con il costrutto principale la group by, quando si crea una VM è conveniente includere anche le dimensioni a granularità superiori, in modo da poter riutilizzare la tabella.

Per rappresentare le dipendenze delle viste materializzate si utilizza un **reticolo multidimensionale**. Più ci si trova in alto al reticolo più ci si avvicina alle dimensioni della tabella dei fatti, più si va in basso più si trova una granularità maggiore.

La scelta delle viste tra tutte le possibili combinazioni è data da:

- si sceglie una sola vista da cui è possibile raggiungere tutti gli attributi;
- creo una vista per ogni query;
- scelgo delle viste intermedie che possono portare a rispondere a più query;

4.1 Documentazione Oracle

Riducono i tempi di esecuzione delle group by e non si eseguono più le join. Nel DBMS Oracle esiste la **query rewriting**, che permette grazie all'ottimizzatore di interpretare le query e se i risultati corrispondono alle condizioni di creazione delle viste, allora la query viene riscritta con la vista.

```
1 create materialized view NAME
2 [build {immediate|deferred}]
3 [refresh {complete|fast|force|never}
4         {on commit|on demand}]
5 [enable query rewrite]
6 as
7     QUERY
```

- immediate: lo schema della tabella viene popolata immediatamente, dato dallo schema di attributi presenti nella select;
- deferred: la vista viene creata, ma viene popolata successivamente;
- complete: i dati vengono presi interamente dal database;
- fast: i dati vengono presi in modo incrementale;
- force: se possibile viene eseguito il refresh in modalità fast, oppure in modalità complete;
- never: la vista non viene mai aggiornata;

- on commit: ogni volta che viene fatto un commit sulla tabella della query anche la vista viene aggiornata;
- on demand: viene definito dall'utente quando aggiornare la vista;
- enable query rewrite: abilita il dbms ad usare la vista per accelerare le query;

Per effettuare il refresh esistono dei tipi di job (a differenza del tipo di prodotto). Quando abbiamo bisogno del fast refresh, la tabella ha bisogno delle informazioni aggiuntive, ovvero dei file di log che ci informano delle nuove informazioni aggiunte al db, la **materialized view log** è associata ad una tabella che ha subito delle variazioni:

```
1 create materialized view log on
2   TABELLA
3 with sequence, rowid
4   (Attributo, ...)
5 including new values;
```

- sequence: istante temporale in cui è avvenuta la modifica;
- rowid: indica la tupla che ha subito una modifica;

Su queste keyword si deve definire una lista di attributi da monitorare, si aggiunge **including new values** per supportare l'inserimento di nuove tuple.

5 Progettazione fisica

Fare una progettazione fisica comporta analizzare il carico di lavoro e definire delle strutture fisiche accessorie per velocizzare le operazioni. Si possono definire delle viste oppure degli indici, ad esempio: indici bitmap, indice di join, ...

La progettazione fisica è dipendente dal carico di lavoro.

La progettazione fisica è caratterizzata da una fase di tuning, utilizzata per testare gli indici e le viste create e decidere se mantenerli o meno.

Gli indici si possono creare sugli attributi che vengono selezionati più frequentemente, se il dominio è ridotto (come quelli categorici dei DW) si utilizza un bitmap, altrimenti un B-tree.

6 Alimentazione dei Data Warehouse

Essendo dei dati derivati, la prima operazione da effettuare è l'ETL, se questo è complesso si va a definire un area di staging in cui il dato viene mantenuto temporaneamente. Il processo di ETL va gestito sia per il popolamento del DW sia per quando verrà aggiornato con dati nuovi.

6.1 Estrazione

L'estrazione statica è la prima estrazione effettuata per popolare il DW. Per fare l'estrazione incrementale si possono:

- creare delle applicazioni ad hoc per i sistemi legacy;
- usando dei log, che non vanno ad interferire con il carico del db;
- usando dei trigger: sono procedure che si attivano quando degli si effettuano delle operazioni specifiche;
- basata su timestamp: dove i recordi hanno il timestamp dell'ultima modifica effettuata su di essi;

6.2 Pulitura

Quando si effettua una estrazione ci si potrebbe trovare di fronte a:

- dati duplicati;
- dati mancanti;
- campo non previsto;
- valori errati o impossibili;
- inconsistenza del valore;

Ogni errore richiede una tecnica specifica per essere risolto, le più comuni sono l'uso di dizionari per controllare errori di battitura, oppure il **join approssimato**, ad esempio: due database non hanno una chiave condivisa per identificare un utente dall'ordine effettuato, allora per fare una join si dovranno prendere i campi comuni, controllandone sempre la consistenza, oppure i problemi di **merge/purge**, ad esempio: facendo il merge di due db le informazioni potrebbero essere duplicate ...

6.3 Trasformazione

Conversione dei dati nel formato di quelli presenti nel data warehouse.

6.4 Caricamento

In fase di caricamento i dati si caricano nel seguente ordine:

- dimensioni;
- fatti;
- indici e viste;

Problem 6.1 – Progettazione Magazzini

Tabelle:

Tempo(codT, data, mese, 3m, 4m, 6m, anno)

Magazzino(codMa, magazzino, città, provincia, regione)

Modello(codMo, modello, categoria)

UsoMtqMagazzino(codMa, codT, mtqLiberi, mtqTot)

UsoProdMagazzino(codMa, codMo, codT, numeroProdottiTotale, valoreTotaleProdotti)

Query:

1. Relativamente al primo trimestre dell'anno 2013, considerando solo i magazzini della città di Torino, trovare per ogni coppia (magazzino, data) il valore complessivo di prodotti presenti in tale data nel magazzino e il valore complessivo medio giornaliero di prodotti presenti nel magazzino nel corso della settimana precedente la data in esame (data in esame inclusa):

```

1 select magazzino, data,
2     sum(valoreTotProdotti) as valoreTot,
3     avg(sum(valoreTotProdotti)) over (
4         partition by magazzino
5         order by data
6         range between interval '7'
7         day preceding and current row
8     ) as valoreMedioSuGiornoCorrenteESettimanaPrecedente
9
10 from UsoProdMagazzino u, Tempo t, Magazzino m
11 where u.codT = t.codT and
12        u.codMa = m.codMa and
13        città = 'torino' and
14        anno = 2013 and
15        3m = 1
16 group by magazzino, data;
```

2. Relativamente all'anno 2004, trovare per ogni coppia(città,data) la percentuale di superficie liberagiornaliera nella città. Associare ad ogni coppia un attributo di rank legato alla percentuale disuperficie libera giornaliera nella città (1 per la coppia con la più bassa percentuale di superficie libera giornaliera).

```
1 select citta, data,
2       sum(mtqLiberi) / sum(mtqTot) * 100 as
   percentualeMtqLiberi,
3       rank() over (
4         order by sum(mtqLiberi) / sum(mtqTot) * 100
5       ) as rankLowestPercentuale
6 from Tempo t, Magazzino m, UsoMtqMagazzino u
7 where t.codT = u.codT and
8       m.codMa = u.codMa and
9       anno = 2004
10 group by citta, data;
```

3. Relativamente ai primi sei mesi dell'anno 2014, trovare per ogni coppia (magazzino,data) la percentuale di superficie libera giornaliera.

```
1 select magazzino, data,
2       100 * sum(mtqLiberi) / sum(mtqTot) as
   percentualeMtqLiberi
3 from Tempo t, Magazzino m, UsoMtqMagazzino u
4 where t.codT = u.codT and
5       m.codMa = u.codMa and
6       anno = 2014 and
7       mese <= 6
8 group by magazzino, data;
```

4. Relativamente all'anno 2013, trovare per ogni coppia (magazzino,mese) il valore complessivo medio giornaliero di prodotti presenti.

```
1 select distinct magazzino, mese,
2       avg(sum(valoreTotProdotti)) over (
3         partition by magazzino, mese
4       ) as valoreMedioGiornalieroComplessivo
5 from UsoProdMagazzino u, Tempo t, Magazzino m
6 where u.codMa = m.codMa and
7       u.codT = t.codT and
8       anno = 2013
9 group by magazzino, mese, data;
```

5. Relativamente all'anno 2015, trovare per ogni regione il valore complessivo medio giornaliero di prodotti presenti nella regione.

```
1 select regione, mese,
2       avg(sum(valoreTotProdotti)) over (
3         partition by regione, mese
4       ) as valoreMedioGiornalieroComplessivo
5 from UsoProdMagazzino u, Tempo t, Magazzino m
6 where u.codMa = m.codMa and
7       u.codT = t.codT and
8       anno = 2015
9 group by regione, mese, data;
```


6. Relativamente all'anno 2014, trovare per ogni coppia(mese, regione) la percentuale di superficie libera giornaliera nella regione.

```
1 select regione, mese,
2     avg (100 * sum(mtqLiberi) / sum(mtqTot)) over (
3         partition by regione
4     )
5 from UsoMtqMagazzino u, Tempo t, Magazzino m
6 where u.codMa = m.codMa and
7       u.codT = t.codT and
8       anno = 2014
9 group by regione, mese, data;
```