2022-12-20 Q1

lunedì 19 dicembre 2022 17:54

QUADERNO 1

Data Science e Tecnologie per le Basi di Dati

I data analyst dell'Associazione Nazionale Musei Italiani sono interessati ad analizzare il ricavo medio per biglietto. In particolare, vorrebbero che le analisi affrontassero i seguenti aspetti.

Un museo ha un nome univoco e si trova in una città specifica. Vengono memorizzate anche la provincia e la regione in cui il museo risiede. La stessa città può ospitare diversi musei. Ogni museo appartiene ad una categoria specifica (ad esempio, "Arte", "Siti storici", "Storia naturale").

Un museo può avere alcuni servizi aggiuntivi disponibili per il suo pubblico. I sistemi registrano quali servizi sono disponibili per ogni museo. Esempi di servizi aggiuntivi sono "visite guidate", "audio guide", "guardaroba", "caffè", "Wi-Fi". Il numero di servizi aggiuntivi è 10 e la loro lista completa è nota.

I biglietti venduti da ogni museo sono registrati. Ci sono 3 diversi tipi di biglietti: "Full Revenue", "Reduced-student" (per studenti dai 14 ai 24 anni) e "Reduced-junior" (per giovani con meno di 14 anni).

I sistemi memorizzano anche come viene acquistato il biglietto. Un biglietto può essere acquistato in tre modalità: online, nelle biglietterie autorizzate, o direttamente all'ingresso del museo.

Le analisi devono essere effettuate considerando la data, il mese, il bimestre, il trimestre, il semestre, l'anno, se la data è un giorno lavorativo o festivo, e la fascia oraria di validità del biglietto. La fascia oraria è memorizzata in 3 intervalli di blocchi di 4 ore (08:00-12:00, 12:01-16:00, 16:01-20:00).

L'azienda è interessata alle statistiche sul ricavo medio per biglietto. L'analisi deve essere effettuata sulla base di:

- nome del museo, categoria del museo, città, provincia, regione
- servizi del museo
- tipo di biglietto (intero, ridotto-studenti, ridotto-junior)
- modalità di acquisto (online, nelle biglietterie autorizzate o all'ingresso del museo)
- data di validità del biglietto, mese, bimestre, trimestre, semestre, giorno lavorativo e fascia oraria

Homework tasks

1. Progettare il data warehouse per rispondere alle specifiche e per rispondere in modo efficiente a tutte le query fornite. Disegnare lo schema concettuale del data warehouse e lo schema logico (tabelle dei fatti e delle dimensioni).

2. Scrivere le seguenti query usando il linguaggio SQL esteso:

   a. Separatamente per ogni tipo di biglietto e per ogni mese (della validità del biglietto), analizzare: le entrate medie giornaliere, le entrate cumulative dall'inizio dell'anno, la percentuale di biglietti relativi al tipo di biglietto considerato sul numero totale di biglietti del mese

   b. Considerare i biglietti del 2021. Separatamente per ogni museo e tipo di biglietto analizzare: il ricavo medio per un biglietto, la percentuale di ricavo sul ricavo totale per la categoria di museo corrispondente, assegnare un rango al museo, per ogni tipo di biglietto, secondo il numero totale di biglietti in ordine decrescente.

3. Creare e mantenere aggiornate una vista materializzata con i comandi **CREATE MATERIALIZED VIEW** e **CREATE MATERIALIZED VIEW LOG** di ORACLE

   Considerare le seguenti query di interesse:

   1 - Separatamente per ogni tipo di biglietto e per ogni mese analizzare le entrate medie giornaliere.

   2 - Separatamente per ogni tipo di biglietto e per ogni mese analizzare le entrate cumulative dall'inizio dell'anno.

   3 - Separatamente per ogni tipo di biglietto e per ogni mese analizzare il numero totale di biglietti, le entrate totali e le entrate medie.

   4 - Separatamente per ogni tipo di biglietto e per ogni mese analizzare il numero totale di biglietti, le entrate totali e le entrate medie per l'anno 2021.

   5 - Analizzare la percentuale di biglietti relativi ad ogni tipo di biglietto e mese sul numero totale di biglietti del mese per ogni tipo di biglietto.

   3.1) Definire una vista materializzata con CREATE MATERIALIZED VIEW, in modo da ridurre il tempo di risposta delle query elencate sopra.
   3.2) Definire i log della vista materializzata con CREATE MATERIALIZED VIEW LOG, per ogni tabella in cui lo si ritiene necessario. Per quali tabelle è utile tenere traccia dei log? Si individuino *tutte e sole* le tabelle necessarie.

Inoltre, per ogni tabella si individuino *tutti e soli* gli attributi per cui è necessario tener traccia delle variazioni.

3.3) Indicare le operazioni sulla base dati (ad esempio INSERT su una specifica tabella) che causano un aggiornamento della MATERIALIZED VIEW creata

4. Aggiornamento e gestione delle viste tramite Trigger

Supponendo che il comando CREATE MATERIALIZED VIEW non sia disponibile, creare la vista materializzata definita nell'esercizio precedente e definire la procedura di aggiornamento a partire da modifiche sulla tabella dei fatti realizzata tramite trigger.

4.1 Creare la struttura della vista materializzata con CREATE TABLE VM1 (...)

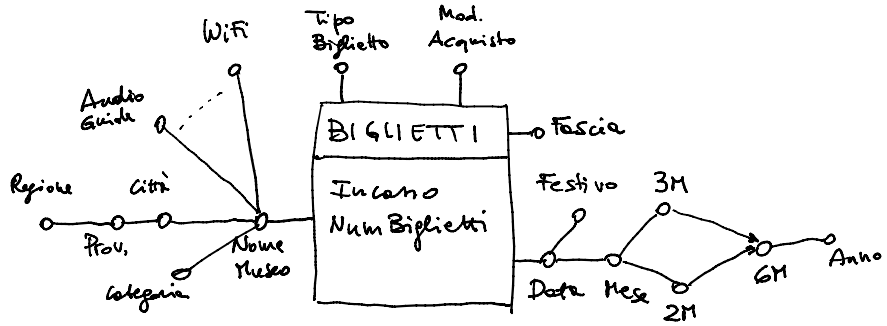
4.2 Popolare opportunamente la tabella creata con il seguente comando

```
INSERT INTO VM1 (...)
( SELECT ...
  ... )
```

4.3 Scrivere il trigger necessario per propagare le modifiche (inserimento di un nuovo record) effettuate nella tabella dei FATTI alla vista materializzata VM1.

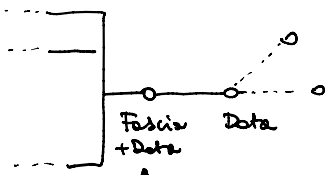
4.4 Specificare quali operazioni (ad esempio INSERT) attivano il trigger creato al punto 4.3.

(1) Schema concettuale



- Dimensione temporale
  - separazione tra data e ora (fascia oraria) [modello sopra]
  - unica gerarchia temporale Data + Fascia

Soluzione accettabile perché Fascia assume SOLO 3 VALORI



Soluzione accettabile perché Fascia assume SOLO 3 VALORI

la fascia DEVE contenere anche tutta l'informazione sulla Data

- Attributo UNIQUE NomeMuseo
  - E' la radice della gerarchia

② Schema logico

a) Junk dimension
   Mod. Acquisto + Tipo Biglietto

b) Pushdown nella tabella dei fatti

Fascia Orarie origina una propria dim. separata

MUSEO (IdMuseo, NomeMuseo, Città, Provincia, Regione, Categoria, AudioGuida, ..., WiFi)

TEMPO (IdTempo, Data, Mese, 2M, 3M, 6M, Anno, Festivo)

FASCIA (IdFascia, Fascia)

a) INFO_BIGLIETTI (IdIB, TipoBiglietto, ModAcquisto)] Junk dimension

BIGLIETTI (IdMuseo, IdTempo, IdFascia, IdIB, Incasso, NumBiglietti)

b) BIGLIETTI (IdMuseo, IdTempo, IdFascia, TipoBiglietto, ModAcquisto, Incasso, NumBiglietti)

③ Queries

Utilizzo lo schema logico b)

a) SELECT TipoBiglietto, Mese, Anno,
SUM (Incasso) / COUNT (DISTINCT Data),
SUM (SUM (Incasso)) OVER (PARTITION BY TipoBiglietto, Anno
ORDER BY Mese
ROWS UNBOUNDED PRECEDING)

100 * SUM (NumBiglietti) /
SUM (SUM (NumBiglietti)) OVER (PARTITION BY Mese)

FROM BIGLIETTI B, TEMPO T

SUM (SUM (NumBiglietti)) OVER (PARTITION BY Mese)
FROM BIGLIETTI B, TEMPO T
WHERE B.IdTempo = T.IdTempo
GROUP BY TipoBiglietto, Mese, Anno

(b)

SELECT NomeMuseo, TipoBiglietto,
SUM (Incasso)/SUM (NumBiglietti),
100* SUM(Incasso) / SUM (SUM (Incasso))
OVER (PARTITION BY TipoBiglietto, Categoria)

RANK() OVER (PARTITION BY TipoBiglietto
ORDER BY SUM(NumBiglietti) DESC)

FROM MUSEO M, BIGLIETTI B, TEMPO T
WHERE <join>
AND Anno = 2021
GROUP BY NomeMuseo, TipoBiglietto, Categoria

(4) Viste materializzate

(a) Analisi delle query

1. Aggr SUM (Incasso)
COUNT (DISTINCT Data)
GB TipoBiglietto, Mese

2. Aggr SUM (Incasso)
GB TipoBiglietto, Mese, Anno

3. Aggr SUM (Incasso)
SUM (NumBiglietti)
GB TipoBiglietto, Mese

4. Come 3. più
WHERE Anno = 2021

5. Aggr SUM (NumBiglietti)
GB TipoBiglietto, Mese

(b) Definizione vista materializzata VM1

2 possibilità

1. GB TipoBiglietto, Mese (+Anno)
Soluzione che non consente di calcolare la query 1 perché non posso mantenere l'aggregato COUNT(DISTINCT Data)

2. GB TipoBiglietto, Data (+Mese +Anno)
Consente di calcolare tutte le query
Cresce la cardinalità di VM1 di circa 30 volte

Consente di calcolare [illegible] query
Cresce la cardinalità di VM1 di circa 30 volte

⇒ Scelgo la possibilità 1

```
SELECT TipoBiglietto, Mese, Anno
       SUM (Incasso) AS TotIncasso,
       SUM (NumBiglietti) AS TotBiglietti
FROM BIGLIETTI B, TEMPO T
WHERE <join>
GROUP BY TipoBiglietto, Mese, Anno
```

VM1

ⓒ Identificatore di VM1

TipoBiglietto, Mese

ⓓ Materialized View logs

BIGLIETTI IdTempo
TipoBiglietto
Incasso
NumBiglietti

TEMPO IdTempo
Mese
Anno

ⓔ Trigger

```
CREATE OR REPLACE TRIGGER MaintView VM1
AFTER INSERT ON BIGLIETTI
FOR EACH ROW

DECLARE
  myMese, myAnno NUMBER;
  N NUMBER;

BEGIN
-- leggo Mese e Anno che mi servono per aggiornare VM1
 SELECT Mese, Anno INTO myMese, myAnno
 FROM TEMPO
 WHERE IdTempo = :NEW.IdTempo;
-- Verifico se esiste già la tupla in VM1
 SELECT COUNT(*) INTO N
 FROM VM1
 WHERE Mese = myMese
  AND TipoBiglietto = :NEW.TipoBiglietto;

 IF (N==0) THEN
-- Non esiste la tupla, la inserisco
    INSERT INTO VM1 (---)   mettere attributi
    VALUES (:NEW.TipoBiglietto
            myMese, myAnno, :NEW.Incasso,
```

```
        INSERT INTO VM1 (---)        mettere attributi
        VALUES (:NEW.TipoBiglietto
                myMese, myAnno, :NEW.Incasso,
                :NEW.NumBiglietti);

    ELSE
    -- Esiste già la tupla
        UPDATE VM1
            SET TotIncasso = TotIncasso + :NEW.Incasso
                TotBiglietti = TotBiglietti + :NEW.NumBiglietti;
          WHERE TipoBiglietto = :NEW.TipoBiglietto
            AND Mese = myMese;

    END IF;
END;
```