

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.

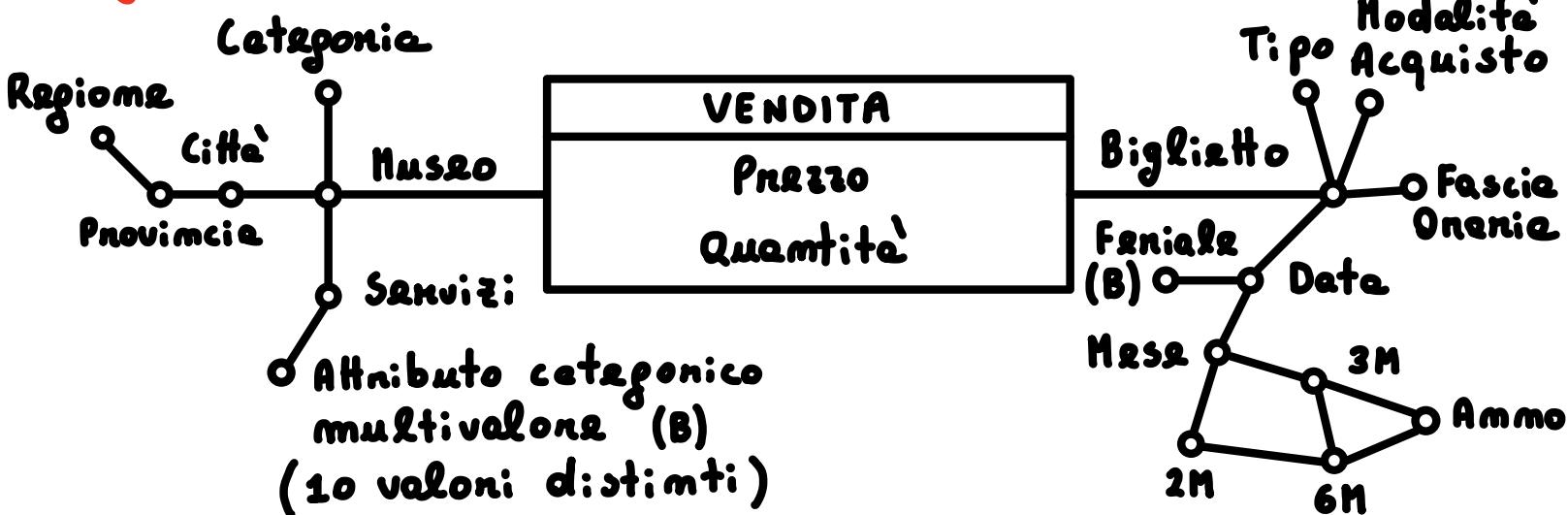
Le analisi devono essere effettuate considerando la data, il mese, il bimestre, il trimestre, il semestre, l'anno, se la data è un giorno lavorativo o festivo, e la fascia oraria di validità del biglietto. La fascia oraria è memorizzata in 3 intervalli di blocchi di 4 ore (08:00-12:00, 12:01-16:00, 16:01-20:00).

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

- 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).

Progettazione concettuale



Progettazione logica

MUSEO (Nome Museo, Città, Provincia, Regione, CATEGORIA)

SERVIZIO (Id Servizio, Nome Museo, Visite Guidate, Wi-Fi...)

BIGLIETTO (Id Biglietto, Tipo, Modalità Acquisto, Data, Mese, 2M, 3M, 6M, Ammo, Fascie (B), FASCIE)

VENDITA (Nome Museo, Id Biglietto, Prezzo, Quantità)

- 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

```

SELECT      B.Tipo, B.Mese, B.Data, B.Ammo,
            SUM(V.Prezzo)/COUNT(DISTINCT B.Data),
            SUM(SUM(V.Prezzo))
OVER(PARTITION BY B.Ammo
        ORDER BY B.Data
        ROW UNBOUNDED PRECEDING) ,
            100 * COUNT(*) / SUM(COUNT(*))
OVER(PARTITION BY B.Mese)
    
```

```

FROM      VENDITA V, BIGLIETTO B, MUSEO M
WHERE     V.NomeMuseo = M.NomeMuseo AND
          V.IdBiglietto = B.IdBiglietto
GROUP BY B.Tipo, B.Mese, B.Ammo
    
```

- 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.

```

SELECT M.Nome Museo, B.Tipo, B.Id Biglietto, M.Categorie,
V.Quantite',
SUM(SUM(V.Prezzo))/SUM(V.Quantite')
OVER(PARTITION BY B.Id Biglietto),
100 * SUM(SUM(V.Prezzo))/SUM(COUNT(*))
OVER(PARTITION BY M.Categorie),
RANK() OVER(PARTITION BY B.Tipo
ORDER BY SUM(V.Quantite')
DESC)
FROM VENDITA V, BIGLIETTO B, MUSEO M
WHERE V.Nome Museo = M.Nome Museo AND
V.Id Biglietto = B.Id Biglietto AND
B.Ambo = '2021'
GROUP BY M.Nome Museo, B.Tipo

```

- 3.1) Definire una vista materializzata con CREATE MATERIALIZED VIEW, in modo da ridurre il tempo di risposta delle query elencate sopra.

	1.	2.	3.	4.	5.
SELECT	Tipo, Mese Date	Tipo, Mese Ammo	Tipo, Mese	Tipo, Mese Ammo	Tipo, Mese
Metriche	SUM(Prezzo)	SUM(Prezzo)	SUM(Quantità) SUM(Prezzo..)	SUM(Quantità) SUM(Prezzo..)	SUM(Quantità)
FROM	VENDITA BIGLIETTO	VENDITA BIGLIETTO	VENDITA BIGLIETTO	VENDITA BIGLIETTO	VENDITA BIGLIETTO
WHERE	Id Biglietto	Id Biglietto	Id Biglietto	Id Biglietto, Ammo	Id Biglietto
GROUP BY	Tipo, Mese	Tipo, Mese	Tipo, Mese	Tipo, Mese	Tipo, Mese

3.1

CREATE MATERIALIZED VIEW MV1

BUILD IMMEDIATE

REFRESH FAST ON DEMAND

ENABLE QUERY REWRITE

AS

```

SELECT Tipo, Mese, Ammo, SUM(Prezzo) AS PrezzoTOT,
       SUM(Quantità) AS VenditeTOT
  FROM BIGLIETTO B, VENDITA V
 WHERE B.Id Biglietto = V.Id Biglietto
 GROUP BY Tipo, Mese
    
```

- 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.

**CREATE MATERIALIZED VIEW LOG ON BIGLIETTO
WITH SEQUENCE, ROWID
(IdBiglietto, Tipo, Mese, Ammo)
INCLUDING NEW VALUES**

**CREATE MATERIALIZED VIEW LOG ON VENDITA
WITH SEQUENCE, ROWID
(IdBiglietto, Prezzo, Quantità)
INCLUDING NEW VALUES**

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

Le operazioni che causano un aggiornamento su MV1 sono le operazioni di INSERT, UPDATE e DELETE.

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 (...)

```

CREATE TABLE VM1 (
    Tipo VARCHAR(20),
    Mese VARCHAR(20),
    Ammo VARCHAR(20),
    PrezzoTOT INTEGER CHECK (PrezzoTOT IS NOT NULL
                                AND PrezzoTOT > 0)
    VenditeTOT INTEGER CHECK (VenditeTOT IS NOT NULL
                                AND VenditeTOT > 0)
    PRIMARY KEY (Tipo, Mese)
)

```

4.2 Popolare opportunamente la tabella creata con il seguente comando

```

INSERT INTO VM1(Tipo, Mese, Ammo, PrezzoTOT, VenditeTOT)

```

```

SELECT Tipo, Mese, Ammo, SUM(Prezzo) AS PrezzoTOT,
       SUM(Quantità) AS VenditeTOT
FROM BIGLIETTO B, VENDITA V
WHERE B.Id Biglietto = V.Id Biglietto
GROUP BY Tipo, Mese

```

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

```
CREATE TRIGGER InsertVENDITA
AFTER INSERT ON VENDITA
FOR EACH ROW
DECLARE VarTipo VARCHAR(20), VarMese VARCHAR(20),
      N INTEGER
BEGIN
    SELECT Tipo INTO VarTipo
    FROM BIGLIETTO
    WHERE IdBiglietto = :NEW.IdBiglietto

    SELECT Mese INTO VarMese
    FROM BIGLIETTO
    WHERE IdBiglietto = :NEW.IdBiglietto

    IF (N > 0) THEN
        UPDATE VM1
        SET PrezzoTOT = PrezzoTOT + :NEW.PrezzoTOT
            VenditeTOT = VenditeTOT + :NEW.VenditeTOT
        WHERE Tipo = VarTipo AND
              Mese = VarMese
```

ELSE

INSERT INTO VM1(Tipo, Mese, Ammo, PrezzoTOT, VenditeTOT)
VALUES (VarTipo, VarMese, :NEW.Ammo, :NEW.PrezzoTOT,
:NEW.VenditeTOT);

END IF;

END;

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

L'operazione che causa l'attivazione del trigger creato
è l'operazione di **INSERT**