QUADERNO 1

Data Science e Tecnologie per le Basi di Dati

2023-2024

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 e tipo di biglietto corrispondenti, 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:

    - Analizzare le entrate medie mensili relative ad ogni tipo di biglietto e per ogni semestre.

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

    - Considerando solo i biglietti acquistati online, separatamente per ogni tipo di biglietto e per ogni mese analizzare il numero totale di biglietti, le entrate totali e le entrate medie per biglietto.

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

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

    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

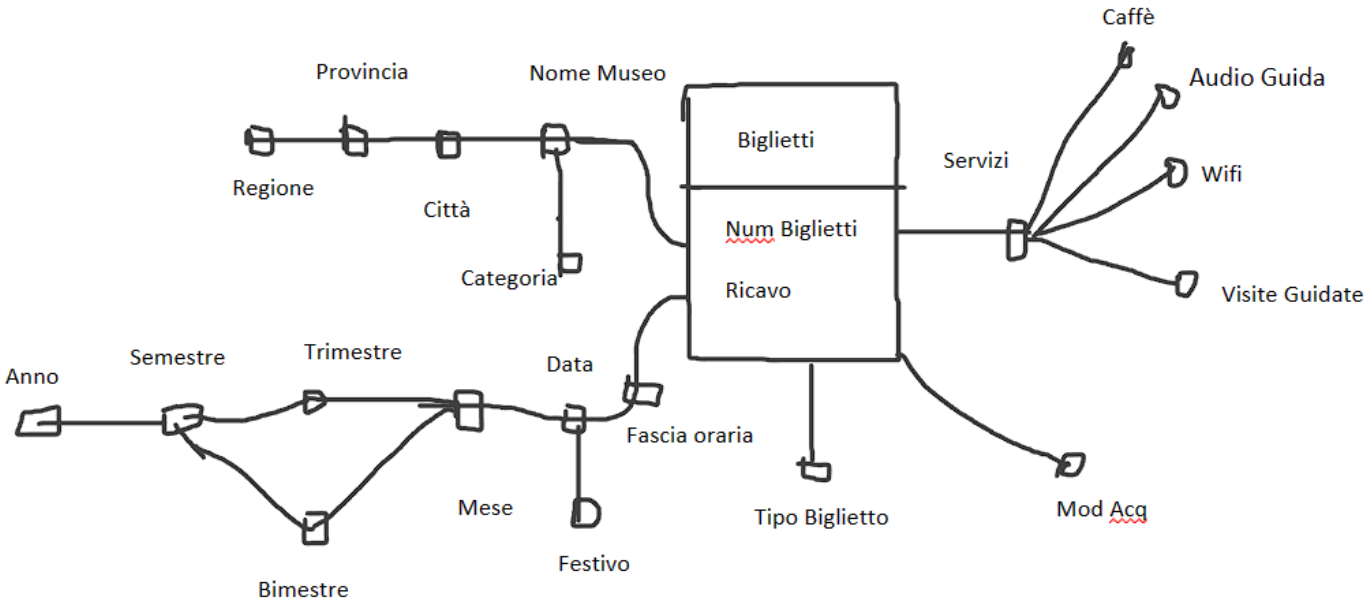
```
INSERT INTO VM1 (...)
( SELECT …
  ... )
```

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.

PARTE 1

SCHEMA CONCETTUALE



SCHEMA LOGICO

SERVIZI(ID_SERVIZIO,WI-FI, CAFFE,AUDIO_GUIDE,VISITE_GUIDATE,...)
INFO_BIGLIETTO(ID_INFO,TIPO_BIGLIETTO,MOD_ACQ)
MUSEO(ID_MUSEO,NOME_MUSEO,CATEGORIA,CITTA,PROVINCIA,REGIONE)
TEMPO(ID_TEMPO,DATA,FESTIVO,FASCIA_ORARIA,MESE,BIMESTRE,TRIMESTRE,SEMESTRE,ANNO)
BIGLIETTI(ID_TEMPO,ID_MUSEO,ID_INFO,ID_SERVIZIO,NUM_BIGLIETTI,RICAVO)

PARTE 2

A)

```
SELECT MESE, TIPO_BIGLIETTO,  SUM(RICAVO)/COUNT(DISTINCT DATA) AS MEDIA_GIORNALIERA,
 SUM(SUM(RICAVO)) OVER (PARTITIONBY ANNO,TIPO_BILGIETTO
        ORDERBY MESE
        ROWS UNBOUNDED PREDCEDING),
100X SUM(NUM_BIGLIETTI)/ SUM(SUM(NUM_BIGLIETTI)) OVER (PARTITION BY MESE) AS PERCENTUALE
FROM INFO_BILGIETTO, BIGLIETTI, TEMPO
WHERE INFO_BIGLIETTO.ID_INFO= BIGLIETTI.ID_INFO AND TEMPO.ID_TEMPO = BIGLIETTI.ID_TEMPO
GROUP BY MESE,TIPO_BIGLIETTO
```

B)

```
SELECT NOME_MUSEO, TIPO_BIGLIETTO, SUM(RICAVO) /SUM(NUM_BIGLIETTI) AS MEDIA_BIGLIETTO,
        100 X SUM(RICAVO)/SUM(SUM(RICAVO)) OVER (PARTITION BY TIPO_BIGLIETTO, CATEGORIA),
        RANK () OVER (PARTITION BY TIPO_BIGLIETTO,ORDER BY SUM(NUM_BIGLIETTI) DESC)
FROM MUSEO, INFO_BIGLIETTO, BIGLIETTI, TEMPO
WHERE ANNO=2021 AND INFO_BIGLIETTO.ID_INFO= BIGLIETTI.ID_INFO AND TEMPO.ID_TEMPO =
BIGLIETTI.ID_TEMPO AND MUSEO.ID_MUSEO= BIGLIETTI.ID_MUSEO
GROUP BY NOME_MUSEO, TIPO_BIGLIETTO, CATEGORIA
```

PARTE 3

1-SEL tipo di biglietto e per ogni semestre, SUM(RICAVO)/SUM()
FROM BIGLIETTI, INFO_BIGLIETTO, TEMPO
GR TIPO_B, SEMESTRE

2-SEL tipo di biglietto, MESE, SUM(VAL_B) OVER (PARTITION BY ANNO
ROWS UNBOUNDED PRECEDING)
FROM BIGLIETTI, INFO_BIGLIETTO, TEMPO
GR TIPO_B, MESE

3-SEL tipo di biglietto, MESE, SUM(NUM_B), SUM(VAL_B), SUM(VAL_B)/ SUM(NUM_B)
FROM BIGLIETTI, INFO_BIGLIETTO, TEMPO
WHERE MOD_ACQ=ONLINE
GR TIPO_B, SEMESTRE

4-SEL tipo di biglietto, MESE, SUM(NUM_B), SUM(VAL_B), SUM(VAL_B)/ SUM(NUM_B)
FROM BIGLIETTI, INFO_BIGLIETTO, TEMPO
WHERE anno=2021.
GR TIPO_B, SEMESTRE

5-SEL tipo di biglietto, MESE, 100 X SUM(NUM_BIGLI)/SUM(NUM_B) OVER (PARTITION BY MESE)
FROM BIGLIETTI, INFO_BIGLIETTO, TEMPO
GR TIPO_B, SEMESTRE

3.1 VISTA

```
CREATE MATERIALIZE VIEW VM1
BUILD IMMEDIATE AS
SELECT MESE, SEMESTRE, ANNO, TIPO_BIGLEITTO, MOD_ACQ, SUM(NUM_BIGLIETTI) AS TOT_BIGLIETTI,
SUM(RICAVO) AS TOT_RICAVO
FROM BIGLIETTI, TEMPO, INFO_BIGLIETTO
WHERE BIGLIETTI.ID_TEMPO=TEMPO.ID_TEMPO AND BIGLIETTI.ID_INFO = INFO_BIGLIETTO.ID_INFO
GROUP BY MESE, SEMESTRE, ANNO, TIPO_BIGLEITTO, MOD_ACQ
```

3.2 VISTA LOG

Le tabelle da tener traccia con i log per la vista materializzata sono:
BIGLIETTI : ID_TEMPO ,ID_INFO ,NUM_BIGLIETTI,RICAVO
INFO_BIGLIETTO: ID_INFO,TIPO_BIGLIETTO,MOD_ACQ
TEMPO: ID_TEMPO, MESE, SEMESTRE, ANNO

```
-CREATE MATERIALIZE VIEW LOG ON BIGLIETTI
WITH SEQUENCE,ROWID
(ID_TEMPO ,ID_INFO ,NUM_BIGLIETTI,RICAVO)
INCLUDING NEW VALUES

-CREATE MATERIALIZE VIEW LOG ON INFO_BIGLIETTO
 WITH SEQUENCE,ROWID
(ID_INFO,TIPO_BIGLIETTO,MOD_ACQ)
 INCLUDING NEW VALUES
```

-CREATE MATERIALIZE VIEW LOG ON TEMPO
WITH SEQUENCE,ROWID
(ID_TEMPO, MESE, SEMESTRE, ANNO)
INCLUDING NEW VALUES

3.3

Le operazioni sulla base dati che causano un aggiornamento sulla materialized view sono le istruzioni di INSERT,UPDATE, DELETE nelle tabelle: BIGLIETTI, INFO_BIGLIETTO, TEMPO

PARTE 4

1.

```
CREATE TABLE VM1(
MESE VARCHAR(20)CHECK (MESE IS NOT NULL),
SEMESTREE VARCHAR(20)CHECK (SEMESTRE IS NOT NULL),
ANNO VARCHAR(20) CHECK (ANNO IS NOT NULL),
TIPO_BIGLIETTO VARCHAR(20) CHECK (TIPO_BIGLIETTO IS NOT NULL),
MOD_ACQ VARCHAR(20) CHECK (MOD_ACQ IS NOT NULL),
TOT_BIGLIETTI INTEGER,
TOT_RICAVO INTEGER
);
```

2.

```
INSERT INTO VM1( MESE,SEMESTRE, ANNO,TIPO_BIGLIETTO, MOD_ACQ, SUM(NUM_BIGLIETTI) AS TOT_BIGLIETTI,
SUM(RICAVO) AS TOT_RICAVO)
(SELECT MESE,SEMESTRE, ANNO,TIPO_BIGLIETTO, MOD_ACQ, SUM(NUM_BIGLIETTI), SUM(RICAVO)
FROM BIGLIETTI, TEMPO, INFO_BIGLIETTO
WHERE BIGLIETTI.ID_TEMPO=TEMPO.ID_TEMPO AND BIGLIETTI.ID_INFO = INFO_BIGLIETTO.ID_INFO
GROUP BY MESE,SEMESTRE, ANNO,TIPO_BIGLIETTO, MOD_ACQ)
```

4.3 TRIGGER

```
CREATE OR REPLACE TRIGGER REFRESH-VIEW
AFTER INSERT ON BIGLIETTI
FOR EACH ROW

DECLARE
        VARMESE VARCHAR(20) ,
        VARSEM VARCHAR(20) ,
        VARANNO VARCHAR(20) ,
        VAR_TIPO, VARCHAR(20) ,
        VAR_MOD VARCHAR(20) ,
        N INT;

BEGIN
        SELECT MESE, SEMESTRE, ANNO
        INTO VARMESE,VARSEM,VARANNO
        FROM TEMPO
        WHERE ID_TEMPO=NEW.ID_TEMPO
```

```
SELCT TIPO_BIGLIETTO, MOD_ACQ
INTO VAR_TIPO,VAR_MOD
FROM INFO_BIGLIETTO
WHERE ID_INFO=NEW.ID_INFO


SELECT COUNT(*) INTO N
FORM VM1
WHERE MESE=VARMESE AND TIPO_BIGLIETTO= VAR_TIPO AND MOD_ACQ=VAR_MOD

IF (N>0) THEN
        UPDATE VM1
        SET TOT_BIGLIETTI = TOT_BIGLIETTI +:NEW.NUM_BIGLIETTI,
        TOT_RICAVO = TOT_RICAVO +:NEW.RICAVO
        WHERE MESE=VARMESE AND TIPO_BIGLIETTO= VAR_TIPO AND MOD_ACQ=VAR_MOD

ELSE
        INSERT INTO  VM1 (MESE,SEMESTRE, ANNO,TIPO_BIGLIETTO, MOD_ACQUI, TOT_BIGLIETTI,
        TOT_RICAVO)
        VALUES(  MESE,SEMESTRE, ANNO,TIPO_BIGLIETTO, MOD_ACQ,:NEW.NUM_BIGLIETTI,
        :NEW.RICAVO )

END IF

END
```

4.4

L’operazione sulla base dati che causano l’attivazione del trigger è l’istruzione di INSERT