Cazzola Michele s323270

QUADERNO 1

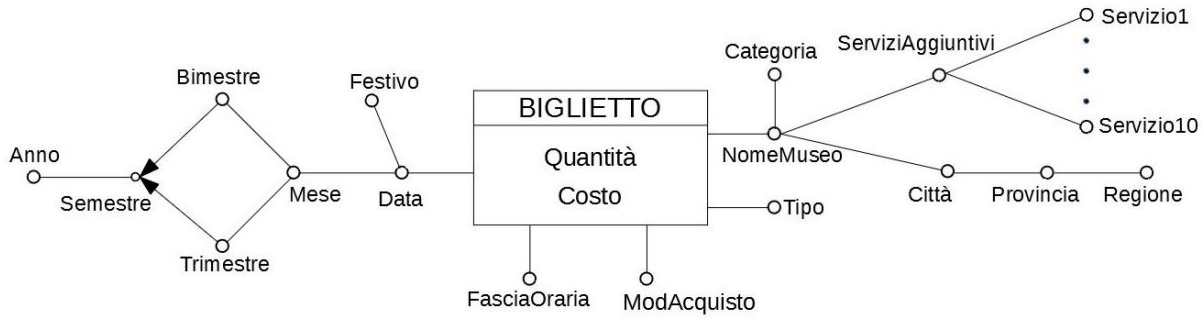
Data Science e Tecnologie per le Basi di Dati

2023-2024

Esercizio 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).

Schema concettuale



Note:

- la tabella di fatto BIGLIETTO rappresenta un biglietto di una tipologia, venduto per una determinata data e una specifica fascia oraria, per un dato museo e acquistato in una delle modalità previste: la misura *Quantità* indica il numero di biglietti venduti, date le caratteristiche sopra specificate, mentre la misura *Costo* indica il costo unitario del biglietto; per ottenere il ricavo nelle interrogazioni, si effettua il prodotto tra le due misure;
- la *Categoria* è funzionalmente dipendente da *NomeMuseo*, in quanto ogni museo appartiene ad una ed una sola categoria;
- tra i servizi aggiuntivi, sono presenti “visite guidate”, “audio guide”, “guardaroba”, “caffè” e “Wi-Fi”, come specificato nel testo.

Schema logico

TEMPO (CodT, Data, Festivo, Mese, Bimestre, Trimestre, Semestre, Anno)

MUSEO (CodM, NomeMuseo, Categoria, Città, Provincia, Regione, CodS, Servizio1, …, Servizio10)

BIGLIETTO (CodM, CodT, Tipo, ModAcquisto, FasciaOraria, Quantità, Costo)

Nota:

- è stata effettuata un’operazione di *push down* delle dimensioni degeneri FasciaOraria, ModAcquisto e Tipo, all’interno della tabella di fatto BIGLIETTO: ciò è possibile in quanto ogni biglietto, rappresentato da un singolo record nella tabella di fatto, ha validità per una sola fascia oraria, è stato acquistato solo secondo una modalità ed appartiene ad una sola tipologia.

Esercizio 2

Scrivere le seguenti query utilizzando 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

```
SELECT  Tipo, Mese,
        SUM(Quantità * Costo) / COUNT(DISTINCT Data) as EntrateMedieGiornaliere,
        SUM(SUM(Quantità * Costo)) OVER (   PARTITION BY Tipo, Anno
                                            ORDER BY Mese
                                            ROWS UNBOUNDED PRECEDING
                                        ) as EntrateCumulativeAnnuali,
        100 * SUM(Quantità) / SUM(SUM(Quantità))
            OVER (PARTITION BY Mese) as PercTipoBigliettiSuTotaleMese
FROM BIGLIETTO B, TEMPO T
WHERE B.CodT = T.CodT
GROUP BY Tipo, Mese, Anno
```

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.

```
SELECT  NomeMuseo, Tipo,
        SUM(Quantità * Costo) / SUM(Quantità) as RicavoMedioPerBiglietto,
        100 * SUM(Quantità * Costo) / SUM(SUM(Quantità * Costo))
            OVER (PARTITION BY Categoria, Tipo) as PercRicavoSuTotalePerCategoriaETipo,
        RANK() OVER (   PARTITION BY Tipo
                        ORDER BY SUM(Quantità) DESC
                    ) as RankNumBigliettiPerTipo
FROM MUSEO M, BIGLIETTO B, TEMPO T
WHERE   M.CodM = B.CodM AND
        B.CodT = T.CodT AND
        T.Anno = 2021
GROUP BY NomeMuseo, Categoria, Tipo
```

Esercizio 3

Considerare le seguenti query di interesse:

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

  Group by: Tipo, Semestre

  Selezione: /

  Misure: SUM (Quantità * Costo), COUNT (DISTINCT Mese)

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

  Group by: Tipo, Mese, Anno

  Selezione: /

  Misure: SUM (Quantità * Costo)

- 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.

  Group by: Tipo, Mese

  Selezione: ModAcquisto

  Misure: SUM (Quantità * Costo), SUM (Quantità)

- 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.

  Group by: Tipo, Mese

  Selezione: Anno

  Misure: SUM (Quantità * Costo), SUM (Quantità)

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

  Group by: Tipo, Mese

  Selezione: /

  Misure: SUM (Quantità)

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

```
CREATE MATERIALIZED VIEW VM_UpdateTempoBiglietti
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
AS
    SELECT  Tipo, Mese, Semestre, Anno, ModAcquisto,
            SUM(Quantità) as NumBigliettiPerTipoMeseModalità,
            SUM(Quantità * Costo) as EntrataPerTipoMeseModalità
    FROM BIGLIETTO B, TEMPO T
    WHERE B.CodT = T.CodT
    GROUP BY Tipo, Mese, Semestre, Anno, ModAcquisto;
```

Nota: non si inserisce la misura COUNT (DISTINCT Mese), necessaria al corretto svolgimento della prima delle query elencate sopra, in quanto è ricavabile direttamente dall'attributo Mese.

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 ROWID, SEQUENCE(Tipo, CodT, ModAcquisto, Quantità, Costo)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON TEMPO
WITH ROWID, SEQUENCE(CodT, Mese, Semestre, Anno)
INCLUDING NEW VALUES;
```

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

Un aggiornamento della vista *VM_UpdateTempoBiglietti* è causato da operazioni di INSERT, UPDATE e DELETE su una delle tabelle TEMPO o BIGLIETTO.

Esercizio 4

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 VM_UpdateTempoBiglietti(
    Tipo varchar(30) NOT NULL,
    Mese date NOT NULL,
    Semestre date NOT NULL,
    Anno integer NOT NULL,
    ModAcquisto varchar(50) NOT NULL,
    NumBigliettiPerTipoMeseModalità integer NOT NULL,
    EntrataPerTipoMeseModalità number NOT NULL,
    PRIMARY KEY(Tipo, Mese, ModAcquisto)
);
```

4.2) Popolare opportunamente la tabella creata con il seguente comando:

INSERT INTO VM1 (...)

(SELECT ...

...)

```
INSERT INTO VM_UpdateTempoBiglietti(Tipo, Mese, Semestre, Anno,
    ModAcquisto, NumBigliettiPerTipoMeseModalità, EntrataPerTipoMeseModalità)
SELECT  Tipo, Mese, Semestre, Anno, ModAcquisto,
            SUM(Quantità) as NumBigliettiPerTipoMeseModalità,
            SUM(Quantità * Costo) as EntrataPerTipoMeseModalità
FROM BIGLIETTO B, TEMPO T
WHERE B.CodT = T.CodT
GROUP BY Tipo, Mese, Semestre, Anno, ModAcquisto;
```

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 OR REPLACE TRIGGER UpdateBiglietto
AFTER INSERT ON BIGLIETTO
FOR EACH ROW
DECLARE
    N number;
    M date;
    S date;
    A integer;
BEGIN

    -- Get informazioni su mese
    select Mese into M
    from TEMPO
    where CodT = :NEW.CodT;

    -- Check esistenza record in vista
    select count(*) into N
    from VM_UpdateTempoBiglietti
    where   Tipo = :NEW.Tipo AND
            Mese = M AND
            ModAcquisto = :NEW.ModAcquisto;
```

```
    if(N > 0) then
        -- Record presente: aggiornamento vista
        update VM_UpdateTempoBiglietti
        set NumBigliettiPerTipoMeseModalità = NumBigliettiPerTipoMeseModalità + :NEW.Quantità,
            EntrataPerTipoMeseModalità = EntrataPerTipoMeseModalità + :NEW.Quantità * :NEW.Costo
        where   Tipo = :NEW.Tipo AND
                Mese = M AND
                ModAcquisto = :NEW.ModAcquisto;

    else
        -- Record assente
        -- Ricavo semestre ed anno corrispondenti al record inserito
        select Semestre, Anno into (S, A)
        from TEMPO
        where CodT = :NEW.CodT;

        -- Inserimento record corrispondente in vista
        insert into VM_UpdateTempoBiglietti(Tipo, Mese, Semestre, Anno,
                ModAcquisto, NumBigliettiPerTipoMeseModalità, EntrataPerTipoMeseModalità)
            value(:NEW.Tipo, M, S, A, :NEW.ModAcquisto, :NEW.Quantità, :NEW.Quantità * :NEW.Costo);

    end if;

END;
```

Nota:

- M ed S, che rappresentano rispettivamente Mese e Semestre della vista, sono considerati come date in quanto contengono anche l'informazione sugli elementi superiori della gerarchia temporale;
- Anno è considerato come numero intero.

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

L'operazione che attiva il trigger creato al punto 4.3 è la INSERT effettuata sulla tabella di fatto BIGLIETTO.