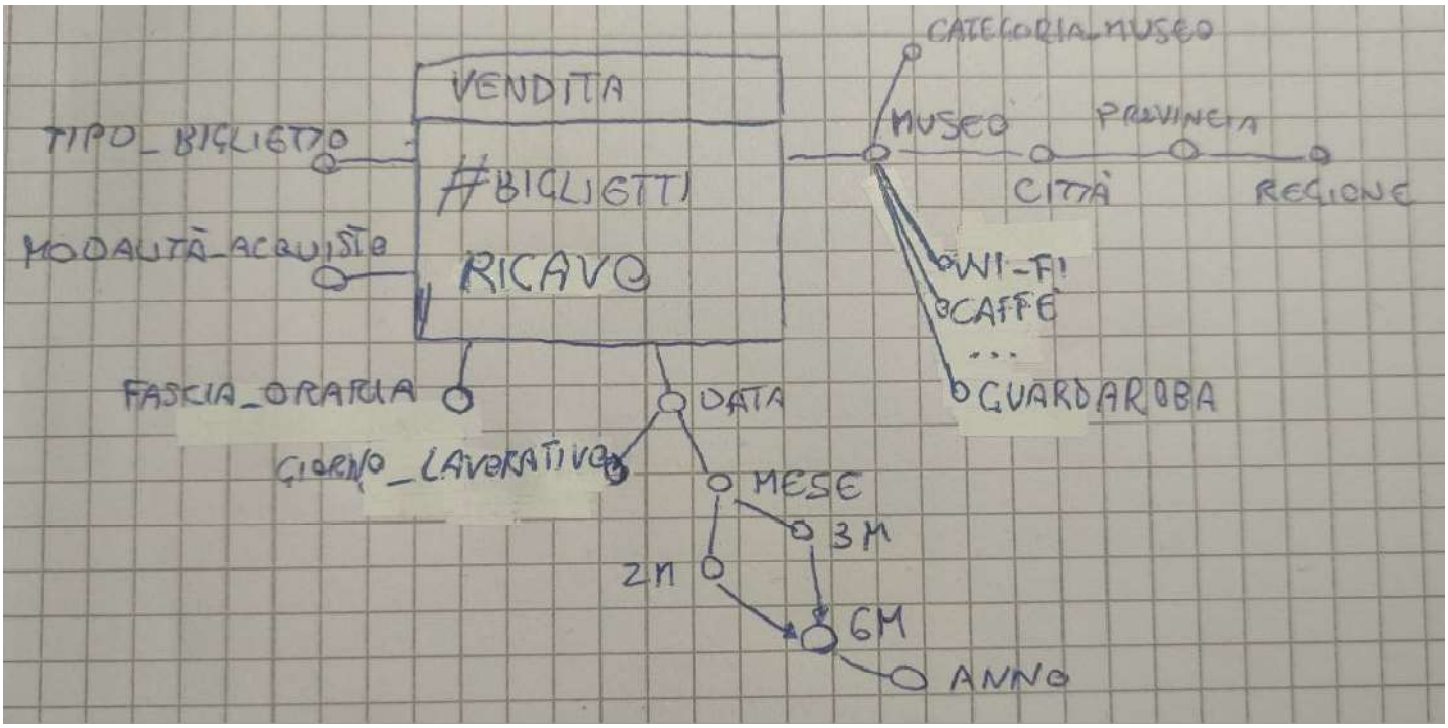
QUADERNO 1 S317641 FEDERICO BUSSOLINO

1.SCHEMA CONCETTUALE



SCHEMA LOGICO

MUSEO(<u>IdMuseo</u>, Cat, Citta, Prov, Reg, Caffe, WiFi, …, Guardaroba)

TEMPO(<u>IdTempo</u>, Data, Lavorativo, Mese, 2M, 3M, 6M, Anno)

JUNK_T(<u>IdJunk</u>, TipoBiglietto, ModAcq, FasciaOraria)

VENDITE(<u>IdMuseo, IdTempo, IdJunk</u>, NBiglietti, Ricavo)

NOTE: ho ritenuto opportuno:

- aggregare tutte le dimensioni degeneri in una junk dimension al fine di non memorizzare dati aggiuntivi nella tabella dei fatti;
- abbreviare i seguenti nomi al fine di rendere più veloce la scrittura di query: Categoria_Museo→Cat, Provincia→Prov, Regione→Reg, Giorno_Lavorativo→Lavorativo, Modalità_Acquisto→ModAcq, #Biglietti→NBiglietti
- considerare i servizi(CAFFE,WIFI,…,GUARDAROBA) e LAVORATIVO attributi booleani, →nel caso dei servizi bisogna quindi essere sicuri che la lista nota a priori non venga cambiata e per quanto riguarda l'attributo LAVORATIVO si assume che non vi sia la futura necessità di calcolare statistiche sul giorno settimanale;

2. QUERY SQL ESTESO

A.

```
SELECT TipoBiglietto, Mese, SUM(Ricavo)/ COUNT(DISTINCT Data) AS EntrGiornaliereMedie
        SUM(SUM(Ricavo)) OVER (PARTITION BY Anno, TipoBiglietto
                                ORDER BY Mese
                                ROWS UNBOUNDED PRECEDING) AS CumAnno,
        100*SUM(NBiglietti)/ SUM(SUM(NBiglietti)) OVER(PARTITION BY Mese) AS PercTipoBiglietto
FROM TEMPO, JUNK_T, VENDITE
WHERE TEMPO.IdTempo=VENDITE.IdTempo AND JUNK_T.IdJunk=VENDITE.IdJunk
GROUP BY TipoBiglietto, Mese, Anno;
```

B.

```
SELECT Museo, TipoBiglietto, SUM(Ricavo)/SUM(NBiglietti) AS RicavoMedioBiglietto,
        100*SUM(Ricavo)/SUM(SUM(Ricavo)) OVER(PARTITION BY Cat) AS PercTipoMuseoOVERCat,
        RANK() OVER(PARTITION BY TipoBiglietto
                        ORDER BY SUM(NBiglietti) DESC)
FROM TEMPO, MUSEO, JUNK_T, VENDITE
WHERE Anno=2021 AND MUSEO.IdMuseo=VENDITE.IdMuseo AND JUNK_T.IdJunk=VENDITE.IdJunk
                AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP BY Museo, TipoBiglietto, Cat;
```

NOTA:

query A: se nessun museo aderente all'Associazione Nazionale Musei Italiani ha venduto biglietti di uno dei tre tipi per almeno una giornata la query è sbagliata (eventualità molto rara se si considerano tutti i musei italiani),

→il numero di giorni del mese, per il mese in cui non sono stati venduti biglietti di quel tipo, sarebbe maggiore di COUNT(DISTINCT Data) del primo denominatore, servirebbe allora una funzione SQL tipo DAY_IN_MONTH(Mese) che ad esempio restituisca 29 o 28 per febbraio(in Mese c'è anche informazione sull'anno) 30 per novembre, 31 per agosto oppure servirebbe una tabella del tipo: (Mese,NGiorniMese)

3. VISTA MATERIALIZZATA

A.* vale la nota sulla query 2.A.

```
SELECT TipoBiglietto, Mese, SUM(Ricavo)/COUNT(DISTINCT Data)
FROM JUNK_T, TEMPO, VENDITE
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP  BY Mese, TipoBiglietto;
```

B.

```
SELECT TipoBiglietto, Mese, SUM(SUM(Ricavo)) OVER (PARTITION BY Anno, TipoBiglietto
                                    ORDER BY Mese
                                    ROWS UNBOUNDED PRECEDING)
FROM JUNK_T, TEMPO, VENDITE
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP  BY Mese, TipoBiglietto, Anno;
```

C.

```
SELECT TipoBiglietto, Mese, SUM(NBiglietti), SUM(Ricavo), SUM(Ricavo)/SUM(NBiglietti)
FROM JUNK_T, TEMPO, VENDITE
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP  BY Mese, TipoBiglietto;
```

D.

```
SELECT TipoBiglietto, Mese, SUM(NBiglietti), SUM(Ricavo), SUM(Ricavo)/SUM(NBiglietti)
FROM JUNK_T, TEMPO, VENDITE
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo AND Anno=2021
GROUP  BY Mese, TipoBiglietto;
```

E.

```
SELECT TipoBiglietto, Mese, 100*SUM(NBiglietti)/ SUM(SUM(Nbiglietti)) OVER
        (PARTITION BY Mese)
FROM JUNK_T, TEMPO, VENDITE
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP  BY Mese, TipoBiglietto;
```

attributi di raggruppamento scelti in modo da ridurre tempo di esecuzione di tutte le query:

A→+ TipoBiglietto, Mese, SUM(Ricavo), Data

B→+ Anno

C→+ SUM(NBigletti)

D→

E→

```
CREATE MATERIALIZED VIEW TIPOBIGLIETTO_MESE
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
ENABLE QUERY REWRITE
AS
        SELECT(TipoBiglietto, Data, Mese, Anno,
                SUM(Ricavo) AS RicavoTOT, SUM(NBigliettI) AS NBigliettiTOT
        FROM VENDITE, TEMPO, JUNK_T
        WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND
                 TEMPO.IdTempo=VENDITE.IdTempo
        GROUP BY TipoBiglietto, Data, Mese, Anno;
```

3.2.

1.

```
CREATE MATERIALIZED VIEW LOG ON VENDITE
WITH SEQUENCE, ROWID, (IdTempo, IdJunk, NBiglietti, Ricavo)
INCLUDING NEW VALUES;
```

2.

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

3.

```
CREATE MATERIALIZED VIEW LOG ON JUNK_T
WITH SQWUENCE, ROWID, (IdJunk, TipoBiglietto)
INCLUDING NEW VALUES;
```

3.3

OPERAZIONI CHE GENERANO AGGIORNAMENTI MV:

INSERT,UPDATE,DELETE ON VENDITE

UPDATE ON TEMPO

UPDATE ON JUNK_T

NOTE:

3.1. FATTORE DI RIDUZIONE (stima di 3000 musei italiani):

JUNK_T → 9 * NO DIM MUSEO →3000 = 27000 → ok

Se aggregassi anche per mese otterrei *30 nel fattore di riduzione ma viene chiesta 1 sola MV per 5 query, COUNT(DISTINCT Data) non è nella query siccome sarebbe complesso aggiornarla.

3.3. UPDATE E DELETE SARANNO ESEGUITE POCO DI FREQUENTE.

4. TEBELLA + TRIGGER

1.

```
CREATE TABLE VM1(
Data DATE CHECK (Data IS NOT NULL)
Mese DATE CHECK (Mese IS NOT NULL)
Anno YEAR CHECK (Anno IS NOT NULL)
TipoBiglietto VARCHAR(20) CHECK (TipoBiglietto IS NOT NULL)
RicavoTOT INTEGER CHECK(RicavoTOT IS NOT NULL)
NBigliettiTOT INTEGER CHECK(NBigliettiTOT IS NOT NULL));
```

2.

```
INSERT INTO VM1(Data,Mese,Anno,TipoBiglietto,RicavoTOT,NBigliettiTOT)
(SELECT(Data, Mese, Anno, TipoBiglietto
        SUM(Ricavo), SUM(NBigliettI)
FROM VENDITE, TEMPO, JUNK_T
WHERE JUNK_T.IdJunk=VENDITE.IdJunk AND TEMPO.IdTempo=VENDITE.IdTempo
GROUP BY TipoBiglietto, Data, Mese, Anno);
```

3.

```
CREATE OR REPLACE TRIGGER AGGVM1
AFTER INSERT ON VENDITE
FOR EACH ROW
DECLARE
        VTipoB VARCHAR (20)
        VData,VMese DATE
        VAnno YEAR
        N INTEGER
        RBV,NBTV INTEGER
BEGIN
        SELECT Data,Mese,Anno INTO (VData,VMese,VAnno)
        FROM TEMPO
        WHERE TEMPO.IdTempo=:NEW.IdTempo
        SELECT TipoBiglietto INTO (VTipoB)
        FROM JUNK_T
        WHERE JUNK_T.IdJunk=:NEW.IdJunk
        SELECT COUNT(*) INTO N
        FROM VENDITE
        WHERE VTipoB=TipoBiglietto AND VData=Data
        IF(N>0) THEN
                UPDATE VM1
                SET RicavoTOT=RicavoTOT+:NEW.Ricavo
                        NBigliettiTOT = NBigliettiTOT+:NEW.NBiglietti
                WHERE VTipoB=TipoBiglietto AND VData=Data
        ELSE
                INSERT INTO VM1(Data,Mese,Anno,TipoBiglietto,RicavoTOT,NBigliettiTOT)
                VALUES (VData,VMese,VAnno,VTipoB,:NEW.Ricavo,:NEW.NBiglietti)
        ENDIF;
END;
```

4.operazioni che attivano il trigger:

INSERT ON VENDITE