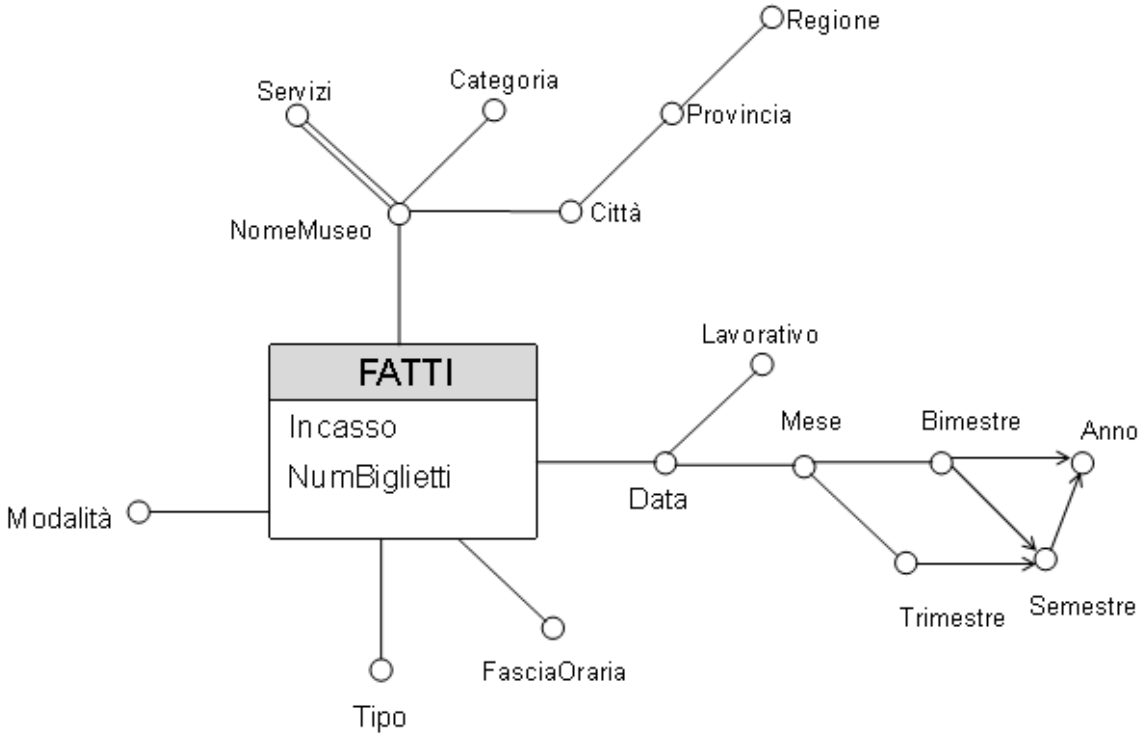
QUADERNO #1 *Data warehouse e Viste materializzate*

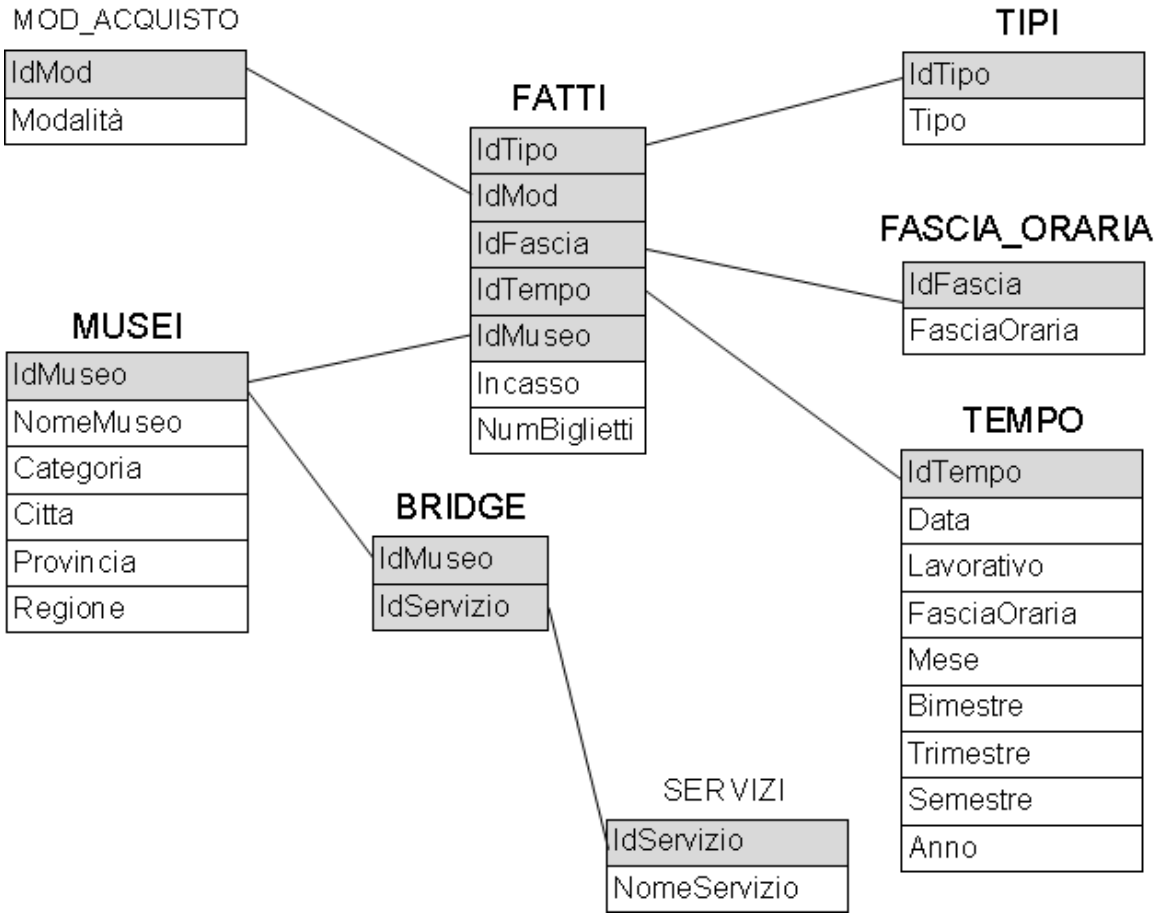
Carlo MIGLIACCIO matricola: 272489

1. Progettazione Data warehouse

a. *Progettazione Concettuale* ***(Dimensional Fact Model)***



b. *Progettazione logica* ***(Schema a Stella)***



2. Query in 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, Anno,
          --La granularità della tupla non è uguale a quella richiesta
          --dalla media da calcolare. Applico la definizione
       (SUM(Incasso)/COUNT(DISTINCT TE.Data)) AS Incasso_AVG,
       SUM(SUM(Incasso)) OVER ( PARTITION BY Tipo, Anno
                                ORDER BY TO_DATE(Mese, "MM-YYYY")
                                ROWS UNBOUNDED PRECEDING  ) AS Incasso_CUM,
        100*(SUM(NumBiglietti) /
        SUM((SUM(NumBiglietti)) OVER ( PARTITION BY Mese ) ) AS Perc_Tipo
FROM Fatti F, Tipo T, Tempo TE
WHERE F.IdTipo=T.Id_Tipo AND
      F.IdTempo=TE.IdTempo
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, Categoria, Tipo,
        (SUM(Incasso)/SUM(NumBiglietti)) AS Prezzo_Medio,
       (100*SUM(Incasso)/
          ( SUM(SUM(Incasso)) OVER
               ( PARTITION BY Categoria, Tipo) )
          ) AS Perc_Ricavo,
        DENSE_RANK() OVER (   PARTITION BY Tipo
                              ORDER BY SUM(NumBiglietti) DESC
                           ) AS Ranking_Museo
FROM Fatti F, Museo M, Tempo T, Tipo TI
WHERE F.IdMuseo=M.IdMuseo AND
      T.IdTempo=F.IdTempo AND
      TI.IdTipo=F.IdTipo AND
      Anno='2021'
GROUP BY NomeMuseo, Categoria, Tipo
```

3. Viste materializzate e log

Di seguito si riporta una tabella in cui per ogni query del *carico di lavoro* si individuano: gli attributi di selezione, gli attributi di `GROUP BY` e le funzioni aggregate d'interesse.

	SELEZIONE	**GROUP BY**	**AGGREGATI**
Query a		Tipo, Semestre	**SUM**(Incasso)
Query b		Tipo, Mese, Anno	**SUM**(Incasso)
Query c	Modalità='online'	Tipo, Mese	**SUM**(NumBiglietti)
Query d	Anno='2021'	Tipo, Mese	**SUM**(Incasso)
Query e		Tipo, Mese	**SUM**(NumBiglietti)

La *vista materializzata* dovrà avere quindi la seguente definizione:

```
CREATE MATERIALIZED VIEW VM1
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
ENABLE QUERY REWRITE
AS ( SELECT Tipo, Modalita, Mese, Semestre,Anno,
                  SUM(Incasso) AS Incasso,
                  SUM(NumBiglietti) AS NumBiglietti
          FROM Fatti F, Tempo T, Tipo TI, Mod_Acquisto M
          WHERE F.IdTempo=T.IdTempo AND
                 F.IdTipo=TI.IdTipo AND
                 M.IdMod=F.IdMod
          GROUP BY Tipo, Modalita, Mese, Semestre,Anno
)
```

Identificatore vista (chiave primaria): **(Tipo, Modalità, Mese)**

Nello schema della vista ci sono le seguenti dipendenze funzionali: Mese → Semestre e Mese → Anno, ragione per cui Semestre e Anno non faranno parte della chiave.

Per usare il **FAST REFRESH** nella vista, bisogna creare dei **log** per tutte le tabelle i cui attributi sono coinvolti nella definizione della vista stessa. La creazione dei log deve **precedere** la creazione della vista. Di seguito se ne riporta la definizione:

```
CREATE MATERIALIZED VIEW LOG ON TIPO
WITH ROWID, SEQUENCE
(IdTipo, Tipo)
INCLUDING NEW VALUES;
```

```
CREATE MATERIALIZED VIEW LOG ON Mod_Acquisto
WITH ROWID, SEQUENCE
(IdMod, Modalita)
INCLUDING NEW VALUES;
```

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

```
CREATE MATERIALIZED VIEW LOG ON Fatti
WITH ROWID, SEQUECE
(IdTipo, IdMod, IdTempo, IdMuseo, Incasso, NumBiglietti)
INCLUDING NEW VALUES;
```

Le seguenti **operazioni sulla base dati** portano ad aggiornamenti della vista definita:

UPDATE	*di*	TIPI TEMPO FATTI MOD_ACQUISTO
INSERT	*in*	FATTI
DELETE	*da*	FATTI

4. Aggiornamento e gestione viste tramite Trigger

Definizione della vista (ipotizzando che il sistema non abbia il comando `CREATE MATERIALIZED VIEW`):

```
CREATE TABLE VM1 (
     Tipo           VARCHAR(30)    NOT NULL,
     Modalita       VARCHAR(30)    NOT NULL,
     Mese           VARCHAR(30)    NOT NULL,
     Semestre       VARCHAR(30)    NOT NULL,
     Anno           INTEGER             NOT NULL,
     Incasso             FLOAT               NOT NULL,
     NumBiglietti   INTEGER        NOT NULL,
     PRIMARY KEY(Tipo, Modalita, Mese)
);
```

Per popolare la vista possiamo sfruttare insieme alla `INSERT INTO` il comando DML usato in precedenza:

```
INSERT INTO VM1 (Tipo, Modalita, Mese, Semestre,
     Anno, Incasso, NumBiglietti)
(    SELECT Tipo, Modalita, Mese, Semestre,Anno,
                SUM(Incasso) AS Incasso,
                SUM(NumBiglietti) AS NumBiglietti
     FROM Fatti F, Tempo T, Tipo TI, Mod_Acquisto M
     WHERE F.IdTempo=T.IdTempo AND
               F.IdTipo=TI.IdTipo AND
               M.IdMod=F.IdMod
     GROUP BY Tipo, Modalita, Mese, Semestre,Anno
);
```

Il **trigger** seguente è stato realizzato per propagare anche nella vista materializzata le modifiche in inserimento fatte sulla tabella dei FATTI:

```
CREATE OR REPLACE TRIGGER Refresh_VM1
AFTER INSERT ON FATTI
FOR EACH ROW
DECLARE
VarTipo        varchar(30);
VarMod         varchar(30);
VarMese        varchar(30);
VarSemestre    varchar(30);
VarAnno        number;
N              number;
BEGIN
--recupero dalle tabelle dimensionali
--le informazioni che mi servono
SELECT Tipo into VarTipo
FROM Tipo
WHERE IdTipo=:NEW.IdTipo;

SELECT Modalita into VarMod
FROM Mod_Acquisto
WHERE IdMod=:NEW.IdMod;

SELECT Mese into VarMese
FROM TEMPO
WHERE IdTempo=:NEW.IdTempo;
```

```
SELECT Semestre into VarSemestre
FROM TEMPO
WHERE IdTempo=:NEW.IdTempo;

SELECT Anno into VarAnno
FROM TEMPO
WHERE IdTempo=:NEW.IdTempo;

--controllo (tramite la chiave) se nella vista
--ho la tupla (Tipo, Modalita, Mese)
SELECT COUNT(*) into N
FROM VM1
WHERE Tipo=VarTipo AND Modalita=VarMod AND
       Mese=VarMese;

IF(N>0) THEN
     --devo solo fare l'update
     UPDATE VM1
     SET Incasso=Incasso+:NEW.Incasso,
          NumBiglietti=NumBiglietti+:NEW.NumBiglietti
     WHERE Tipo=VarTipo AND
            Modalita=VarMod AND
            Mese= VarMese;
ELSE
     --devo inserire per la prima volta il record
     INSERT INTO VM1 (Tipo, Modalita, Mese, Semestre, Anno,
                         Incasso, NumBiglietti)
     VALUES(VarTipo, VarModalita, VarMese, VarSemestre, VarAnno,
             :NEW.Incasso, :NEW.NumBiglietti);
END IF;

END
```

Il trigger, come da definizione, viene attivato dopo (`AFTER`) l'inserimento (`INSERT`) di una tupla nella tabella dei fatti (`ON FATTI`) e viene eseguito per ogni riga che lo innesca (`FOR EACH ROW`).