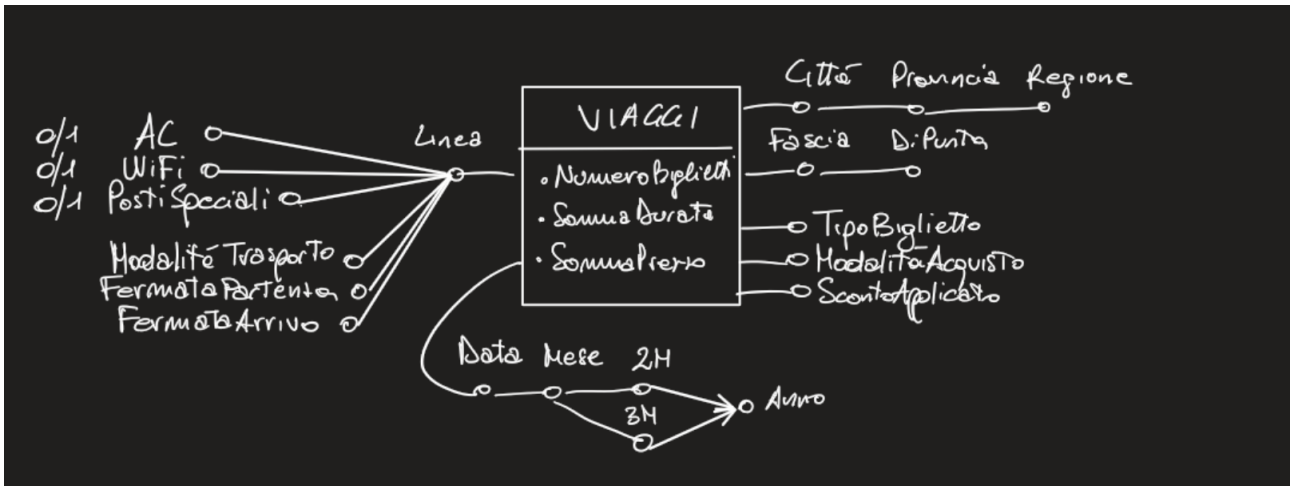
Assunzioni effettuate

Nel progetto dello schema concettuale, ho fatto le seguenti assunzioni:

- Il codice della modalità di trasporto non identifica un singolo mezzo, ma una categoria di mezzi (non il treno, ma la classe dei treni ad esempio) quindi le modalità di trasporto sono un concetto più generare della singola linea
- Una stessa linea può essere operata da più mezzi di trasporto differenti. Questo lo deduco dal passo nel testo in cui si dice “vengono registrati trasferimenti tra linee o modalità diverse”. L’or esclusivo mi fa pensare che ci possano essere trasferimenti tra modalità diverse fermo restando la linea.
- Ogni biglietto è specifico di una singola linea.
- Tutti i biglietti sono di tipo corsa singola.
- Con il termine “percorso” nella richiesta 2.b in realtà ci si riferisca al termine “linea” usato nelle specifiche iniziali.
- Una persona possa viaggiare su una stessa linea anche solo parzialmente, di conseguenza una metrica sarà la somma delle durate.
- Ho assunto infine che le specifiche di analisi debbano essere necessariamente incluse all’interno del progetto, anche se non usate nelle query. Di conseguenza, non ho effettuato pruning su di queste.

Progettazione concettuale (1)



Progettazione Logica

Fatti

- VIAGGI(**IdViaggi**, IdPercorso, IdTempo, IdIntervallo, IdJunk, NumeroBiglietti, SommaDurata, SommaPrezzo)

Dimensioni

- PERCORSO(**IdPercorso**, Linea, AC, WiFi, PostiSpeciali, ModalitàTrasporto, FermataPartenza, FermataArrivo)
- LUOGO(**IdLuogo**, Città, Provincia, Regione)
- TEMPO(**IdTempo**, Data, Mese, 2M, 3M, Anno)
- INTERVALLO(**IdIntervallo**, Fascia, DiPunta)

- JUNK(**TipoBiglietto**, ModalitàAcquisto, ScontoApplicato)

Progettazione Fisica

```
CREATE TABLE PERCORSO (
    IdPercorso INTEGER NOT NULL PRIMARY KEY,        -- Chiave primaria
    Linea CHAR(50),                    -- Nome o numero della linea
    AC INTEGER,                             -- Presenza di aria condizionata (1/0)
    Wifi INTEGER,                           -- Presenza di Wi-Fi (1/0)
    PostiSpeciali INTEGER,                  -- Presenza di posti speciali (1/0)
    ModalitaTrasporto CHAR(50),        -- Modalità di trasporto (es. autobus, treno)
    FermataPartenza CHAR(100),         -- Fermata di partenza
    FermataArrivo CHAR(100)           -- Fermata di arrivo
);
CREATE TABLE TEMPO (
    IdTempo INTEGER NOT NULL PRIMARY KEY,             -- Identificativo temporale
    Giorno DATE,                                     -- Data specifica
    MeseNome CHAR(10),                            -- Nome del mese
    Mese CHAR(10),                                -- Mese
    Bimestre CHAR(10),                            -- Intervallo di 2 mesi
    Trimestre CHAR(10),                           -- Intervallo di 3 mesi
    Anno INTEGER                                       -- Anno
);
CREATE TABLE LUOGO (
    IdLuogo INTEGER NOT NULL PRIMARY KEY,              -- Identificativo del luogo
    Citta CHAR(50),                    -- Nome della città
    Provincia CHAR(50),                -- Nome della provincia
    Regione CHAR(50)                   -- Nome della regione
);
CREATE TABLE INTERVALLO (
    IdIntervallo INTEGER NOT NULL PRIMARY KEY,         -- Identificativo
dell'intervallo
    Fascia CHAR(50),                   -- Fascia oraria
    DiPunta INTEGER                       -- Se l'intervallo è di punta (TRUE/FALSE)
);
CREATE TABLE JUNK (
    IdJunk  INTEGER NOT NULL PRIMARY KEY,
    TipoBiglietto CHAR(50),           -- Tipo di biglietto
    ModalitaAcquisto CHAR(50),         -- Modalità di acquisto (es. online, in loco)
    ScontoApplicato CHAR(50)           -- Tipo di sconto applicata
);
CREATE TABLE VIAGGI (
    IdViaggi INTEGER NOT NULL,             -- Identificativo del viaggio
    IdPercorso INTEGER NOT NULL,                    -- Chiave esterna per la
tabella PERCORSO
    IdLuogo INTEGER NOT NULL,                       -- Chiave esterna per la
tabella LUOGO
```

```
    IdTempo INTEGER NOT NULL,                       -- Chiave esterna per la
tabella TEMPO
    IdIntervallo INTEGER NOT NULL,                  -- Chiave esterna per la
tabella INTERVALLO
    IdJunk INTEGER NOT NULL,                        -- Chiave esterna per la
tabella JUNK
    NumeroBiglietti INTEGER,                -- Numero totale di biglietti
    SommaDurata FLOAT,                      -- Somma delle durate dei viaggi
    SommaPrezzo FLOAT,                      -- Somma dei prezzi dei biglietti
    PRIMARY KEY (IdViaggi, IdPercorso, IdTempo, IdLuogo, IdIntervallo, IdJunk), --
Chiave primaria
    FOREIGN KEY (IdPercorso) REFERENCES PERCORSO(IdPercorso),
    FOREIGN KEY (IdLuogo) REFERENCES LUOGO(IdLuogo),
    FOREIGN KEY (IdTempo) REFERENCES TEMPO(IdTempo),
    FOREIGN KEY (IdIntervallo) REFERENCES INTERVALLO(IdIntervallo),
    FOREIGN KEY (IdJunk) REFERENCES JUNK(IdJunk)
);
```

Scrittura delle queries richieste (2)

```
-- QUERY 2.A

SELECT P.MODALITATRASPORTO, T.MESE,
    SUM(NUMEROBIGLIETTI) AS BIGLIETTIVENDUTI,
    SUM(SUM(NUMEROBIGLIETTI)) OVER (PARTITION BY MODALITATRASPORTO ORDER BY MESE
ROWS UNBOUNDED PRECEDING) AS SOMMACUMULATIVA,
    ROUND(SUM(NUMEROBIGLIETTI)*100.0/SUM(SUM(NUMEROBIGLIETTI)) OVER (PARTITION BY
MESE),2) AS PERCENTUALEMESE
FROM VIAGGI V
JOIN PERCORSO P ON P.IDPERCORSO=V.IDPERCORSO
JOIN TEMPO T ON T.IDTEMPO=V.IDTEMPO
GROUP BY P.MODALITATRASPORTO, T.MESE
ORDER BY MESE, MODALITATRASPORTO;

-- QUERY 2.B

SELECT  CITTA, MODALITATRASPORTO, LINEA,
    ROUND(SUM(SOMMADURATA)/SUM(NUMEROBIGLIETTI), 2) AS DURATAMEDIA,
    SUM(SUM(SOMMAPREZZO)) OVER (PARTITION BY CITTA) AS RICAVITOTALI,
    ROUND(
        SUM(SUM(SOMMAPREZZO)) OVER (PARTITION BY CITTA,
MODALITATRASPORTO,LINEA)*100.0/
        SUM(SUM(SOMMAPREZZO)) OVER (PARTITION BY CITTA,MODALITATRASPORTO),2)
    AS PERCRICAVILINEE,
    RANK() OVER (PARTITION BY CITTA, MODALITATRASPORTO ORDER BY SUM(SOMMAPREZZO)
DESC) AS CLASSIFICALINEE
FROM VIAGGI V
JOIN TEMPO T ON T.IDTEMPO=V.IDTEMPO
```

```
JOIN PERCORSO P ON P.IDPERCORSO=V.IDPERCORSO
JOIN LUOGO L ON L.IDLUOGO=V.IDLUOGO
WHERE T.ANNO>=2022
GROUP BY P.MODALITATRASPORTO, L.CITTA, P.LINEA
ORDER BY L.CITTA, P.MODALITATRASPORTO, PERCRICAVILINEE DESC
;
```

Valutazione per la vista materializzata (3)

Q	FROM	GROUP BY	WHERE	SELECT
1	V, P, T	Modalitatrasporto, Mese		SUM(NumeroBiglietti)/30
2	V, P, T	Modalitatrasporto, Mese		SUM(SUM(NumeroBiglietti))) OVER (PARTITION BY ModalitaTrasporto ORDER BY Mese ROW UNBOUNDED PRECEDING)
3	V, P, T	Modalitatrasporto, Mese		SUM(NumeroBiglietti) , SUM(SommaPrezzo), SUM(SUM(SOMMAPREZZO)) OVER () / COUNT(*) OVER ()
4	V, P, T	Modalitatrasporto, Mese	Anno=2024	SUM(NumeroBiglietti) , SUM(SommaPrezzo), SUM(SUM(SOMMAPREZZO)) OVER () / COUNT(*) OVER ()
5	V, P, T	Modalitatrasporto, Mese		SUM(NUMEROBIGLIETTI)*100.0 / SUM(SUM(NUMEROBIGLIETTI)) OVER (PARTITION BY MESE)

P=Percorso, T=Tempo, J=Junk, I=intervallo, L=luogo, V=Viaggi

C'è una sola query che impone un vincolo nella where, di conseguenza nella vista materializzata questo vincolo sarà trascurato.
L'insieme minimale di attributi della vista materializzata che ne costituisce la chiave primaria è ModalitaTrasporto, Mese. Includerà anche l'anno per poter essere utilizzata dalla query 4 e imporre il vincolo.
Per quanto riguarda le funzioni di aggregazione, si implementeranno quelle relative alla query 1,2,3,5 ad eccezione della funzione di aggregazione che calcola la media rispetto al totale dei ricavi separatamente per mese e mezzo, dal momento che introduce una colonna con lo stesso valore per ogni riga e si può ottenere facilmente e soprattutto efficientemente da una vista che fornisce per mese e mezzo separatamente la rendita.
Inoltre, dal momento che si richiede di utilizzare un fast refresh, questo impone delle limitazioni per quanto riguarda l'uso delle finestre nelle viste materializzate. Pertanto, la vista materializzata creerà solamente le aggregazioni SUM(Numerobiglietti) e SUM(Sommaprezzo).

E' utile, e soprattutto necessario nel caso del fast refresh, creare i log per ognuna delle tabelle coinvolte nella vista materializzata, che sono nel mio vaso viaggi, percorso e tempo. Il filtro sugli attributi è stato impostato per includere tutti e soli gli attributi considerati nella select e le chiavi primarie delle tabelle (implicitamente inclusi attraverso la direttiva PRIMARY KEY nella definizione dei log).

Scrittura vista materializzata

```
-- LOG PER LA VISTA MATERIALIZZATA
CREATE MATERIALIZED VIEW LOG ON VIAGGI
  WITH PRIMARY KEY, ROWID, SEQUENCE (SOMMAPREZZO, NUMEROBIGLIETTI)
```

```
  INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON TEMPO
  WITH PRIMARY KEY, ROWID, SEQUENCE (MESE, ANNO)
  INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON PERCORSO
  WITH PRIMARY KEY, ROWID, SEQUENCE(MODALITATRASPORTO)
  INCLUDING NEW VALUES;

-- VISTA MATERIALIZZATA

CREATE MATERIALIZED VIEW VM
BUILD IMMEDIATE
    REFRESH FAST ON COMMIT
    ENABLE QUERY REWRITE
AS
    SELECT P.MODALITATRASPORTO, T.MESE, T.ANNO,
        SUM(V.NumeroBiglietti),
        SUM(V.SommaPrezzo)
    FROM VIAGGI V, PERCORSO P, TEMPO T
    WHERE V.IDPERCORSO = P.IDPERCORSO AND V.IDTEMPO = T.IDTEMPO
    GROUP BY P.MODALITATRASPORTO, T.MESE, T.ANNO;

SELECT * FROM VM;
```

Viste materializzate con trigger (4)

```
-- creazione della vista come tabella
CREATE TABLE VM1 (
    MODALITATRASPORTO CHAR(50) NULL,
    MESE CHAR(10) NULL,
    ANNO NUMBER(22, 0) NULL,
    NUMEROBIGLIETTI_SUM NUMBER(22, 0) NULL,
    SOMMAPREZZO_SUM NUMBER(22, 0) NULL
);

--istruzione per forzare un refresh complete della tabella
INSERT INTO VM1 (MODALITATRASPORTO, MESE, ANNO, NUMEROBIGLIETTI_SUM,
SOMMAPREZZO_SUM)
SELECT P.MODALITATRASPORTO, T.MESE, T.ANNO, SUM(V.NUMEROBIGLIETTI),
SUM(V.SOMMAPREZZO)
    FROM VIAGGI V
    JOIN PERCORSO P ON P.IDPERCORSO = V.IDPERCORSO
    JOIN TEMPO T ON T.IDTEMPO = V.IDTEMPO
    GROUP BY P.MODALITATRASPORTO, T.MESE, T.ANNO;

--definizione del trigger
CREATE TRIGGER UpdateAfterInsertionVM1
```

```
AFTER INSERT ON VIAGGI
FOR EACH ROW
DECLARE
    needUpdate NUMBER;
    meseNew VARCHAR2(10);
    annoNew VARCHAR2(10);
    modalitatrasportoNew VARCHAR2(50);
BEGIN
    SELECT T.MESE, T.ANNO
    INTO meseNew, annoNew
    FROM TEMPO T
    WHERE T.IDTEMPO = :NEW.IDTEMPO;

    SELECT P.MODALITATRASPORTO
    INTO modalitatrasportoNew
    FROM PERCORSO P
    WHERE P.IDPERCORSO = :NEW.IDPERCORSO;

    SELECT COUNT(*) INTO needUpdate
    FROM VM1
    WHERE VM1.MESE = meseNew
      AND VM1.MODALITATRASPORTO = modalitatrasportoNew;

    -- Se non esiste, inserisci una nuova riga
    IF needUpdate = 0 THEN
        INSERT INTO VM1 (MODALITATRASPORTO, MESE, ANNO, NUMEROBIGLIETTI_SUM,
SOMMAPREZZO_SUM)
        VALUES (modalitatrasportoNew, meseNew, annoNew, :NEW.NUMEROBIGLIETTI,
:NEW.SOMMAPREZZO);
    ELSE
        -- Se esiste, aggiorna i valori esistenti
        UPDATE VM1
        SET NUMEROBIGLIETTI_SUM = NUMEROBIGLIETTI_SUM + :NEW.NUMEROBIGLIETTI,
            SOMMAPREZZO_SUM = SOMMAPREZZO_SUM + :NEW.SOMMAPREZZO
        WHERE MESE = meseNew
          AND MODALITATRASPORTO = modalitatrasportoNew;
    END IF;
END;
```

Il trigger si attiva solamente con una operazione di INSERT che rispetta i vincoli sulla tabella dei fatti VIAGGI. Inserzioni sulle altre tabelle coinvolte come TEMPO e PERCORSO non attivano il trigger né tantomeno ciò è necessario. Grazie alla clausola AFTER, una inserzione in VIAGGI deve rispettare i vincoli di integrità che includono le referenze alle chiavi esterne di tempo e percorso.