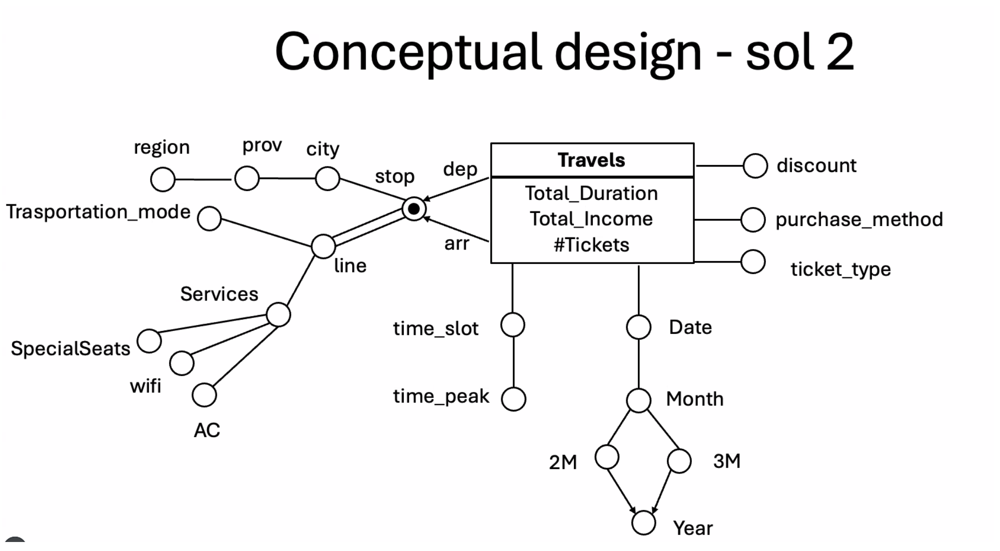
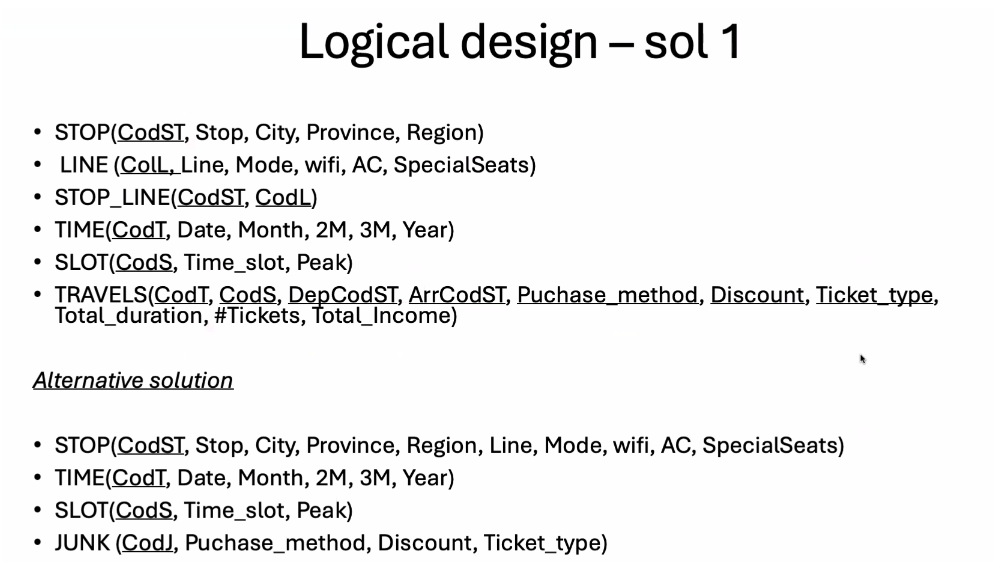
DataBase HW1 Correction

Solution 1:



It differs from mine regarding the travel. Stops are modeled with an apposite node. Double edge between stop and line: each stop can be served by different lines and each line has more stops (many to many relation). Fo Services it's a configuration of three possible values that can cohexist together too.

Translation to Logical Schema:



Logical design – sol 1

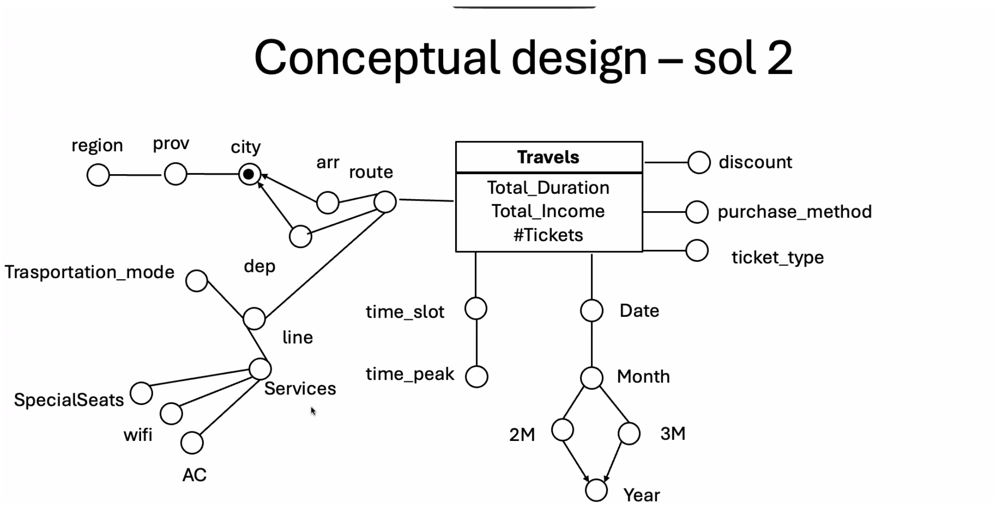
- STOP(<u>CodST</u>, Stop, City, Province, Region)
- LINE (<u>ColL,</u> Line, Mode, wifi, AC, SpecialSeats)
- STOP_LINE(<u>CodST</u>, <u>CodL</u>)
- TIME(<u>CodT</u>, Date, Month, 2M, 3M, Year)
- SLOT(<u>CodS</u>, Time_slot, Peak)
- TRAVELS(<u>CodT</u>, <u>CodS</u>, <u>DepCodST</u>, <u>ArrCodST</u>, <u>Puchase_method</u>, <u>Discount</u>, <u>Ticket_type</u>, Total_duration, #Tickets, Total_Income)

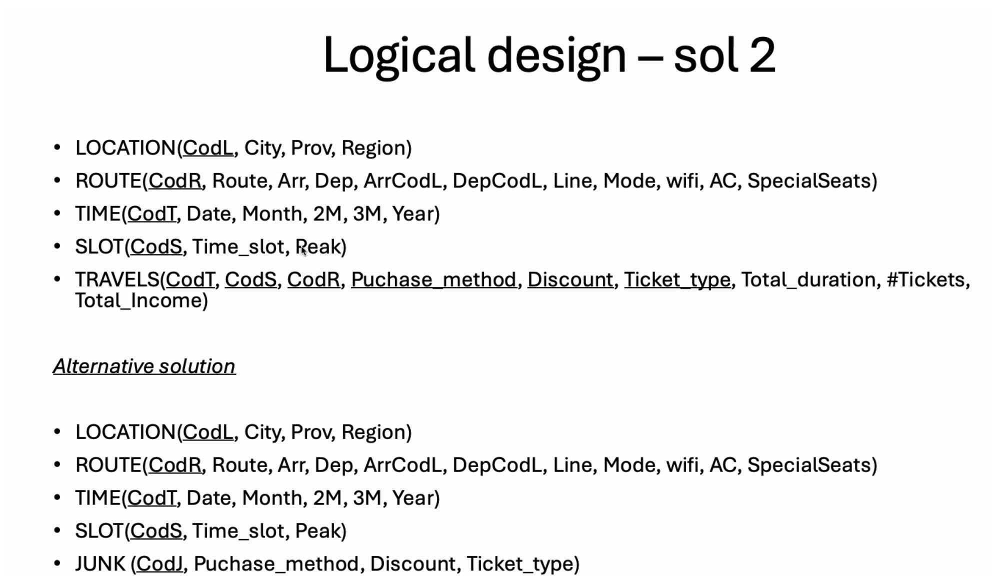
<u>Alternative solution</u>

- STOP(<u>CodST</u>, Stop, City, Province, Region, Line, Mode, wifi, AC, SpecialSeats)
- TIME(<u>CodT</u>, Date, Month, 2M, 3M, Year)
- SLOT(<u>CodS</u>, Time_slot, Peak)
- JUNK (<u>CodJ</u>, Puchase_method, Discount, Ticket_type)

STOP_LINE table links each stop to each line.
The alternative solution uses a JUNK table. In the first case I put all the possible values into the fact table, whereas in the second case I have only codJ inside the fact table.

SECOND SOLUTION:





Logical design – sol 2

- LOCATION(CodL, City, Prov, Region)
- ROUTE(CodR, Route, Arr, Dep, ArrCodL, DepCodL, Line, Mode, wifi, AC, SpecialSeats)
- TIME(CodT, Date, Month, 2M, 3M, Year)
- SLOT(CodS, Time_slot, Peak)
- TRAVELS(CodT, CodS, CodR, Puchase_method, Discount, Ticket_type, Total_duration, #Tickets, Total_Income)

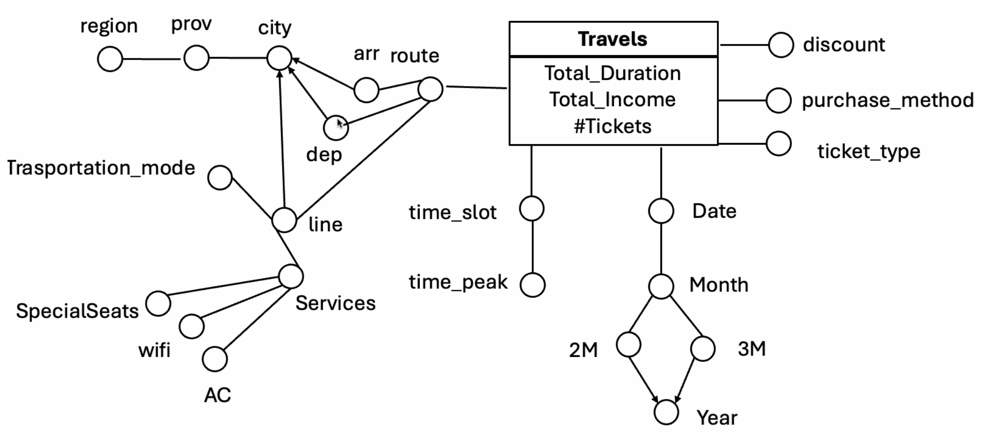
Alternative solution

- LOCATION(CodL, City, Prov, Region)
- ROUTE(CodR, Route, Arr, Dep, ArrCodL, DepCodL, Line, Mode, wifi, AC, SpecialSeats)
- TIME(CodT, Date, Month, 2M, 3M, Year)
- SLOT(CodS, Time_slot, Peak)
- JUNK (CodJ, Puchase_method, Discount, Ticket_type)

Route can be covered by several lines! Route = il tuo percorso. Line= tutta la linea del mezzo di trasporto

THIRD SOLUTION:

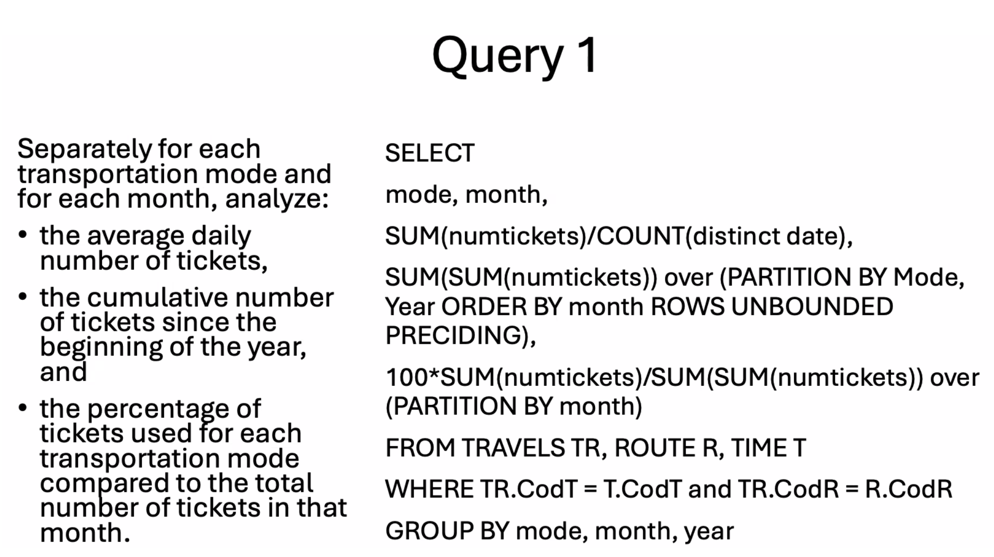
Conceptual design – sol 3



Line belongs only to one city here and it's better.

We'll use solution 3 for the other parts of the homework.

Query 1



Query 1

Separately for each transportation mode and for each month, analyze:

- the average daily number of tickets,
- the cumulative number of tickets since the beginning of the year, and
- the percentage of tickets used for each transportation mode compared to the total number of tickets in that month.

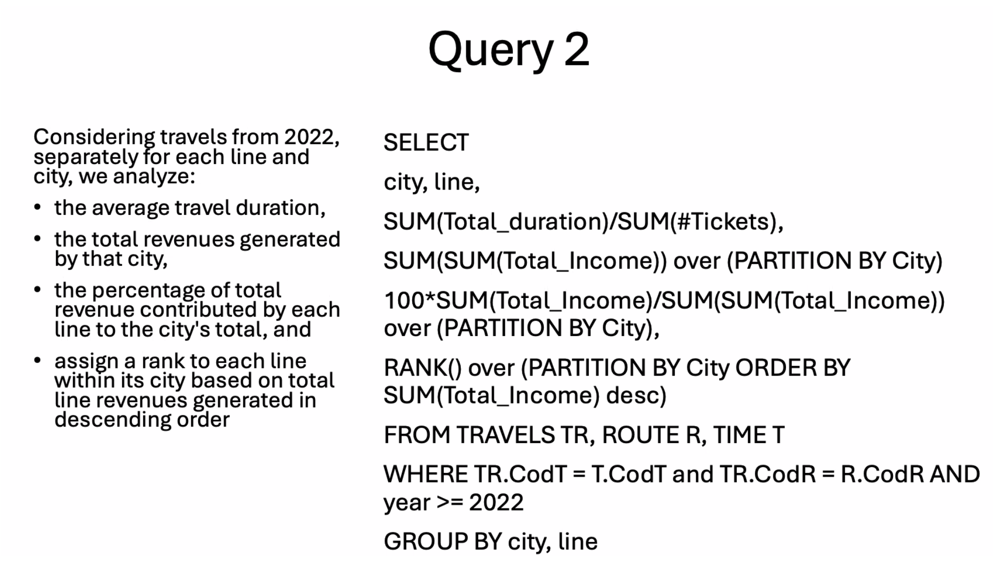
```
SELECT
mode, month,
SUM(numtickets)/COUNT(distinct date),
SUM(SUM(numtickets)) over (PARTITION BY Mode,
Year ORDER BY month ROWS UNBOUNDED
PRECIDING),
100*SUM(numtickets)/SUM(SUM(numtickets)) over
(PARTITION BY month)
FROM TRAVELS TR, ROUTE R, TIME T
WHERE TR.CodT = T.CodT and TR.CodR = R.CodR
GROUP BY mode, month, year
```

Why partitioning also by Mode in the second point???

If I partition by month and mode in the percentage (third point) I'd have just one row per partition.

In the second point the cumulative is asked. In general the measures are related to attributes in Group By. I want the cumulative of number of tickets for each year and for each mode. If you remove Mode from partition I compute for each Year the number of tickets independently from the Mode and I'll lose the info about the Mode.

Query 2



Query 2

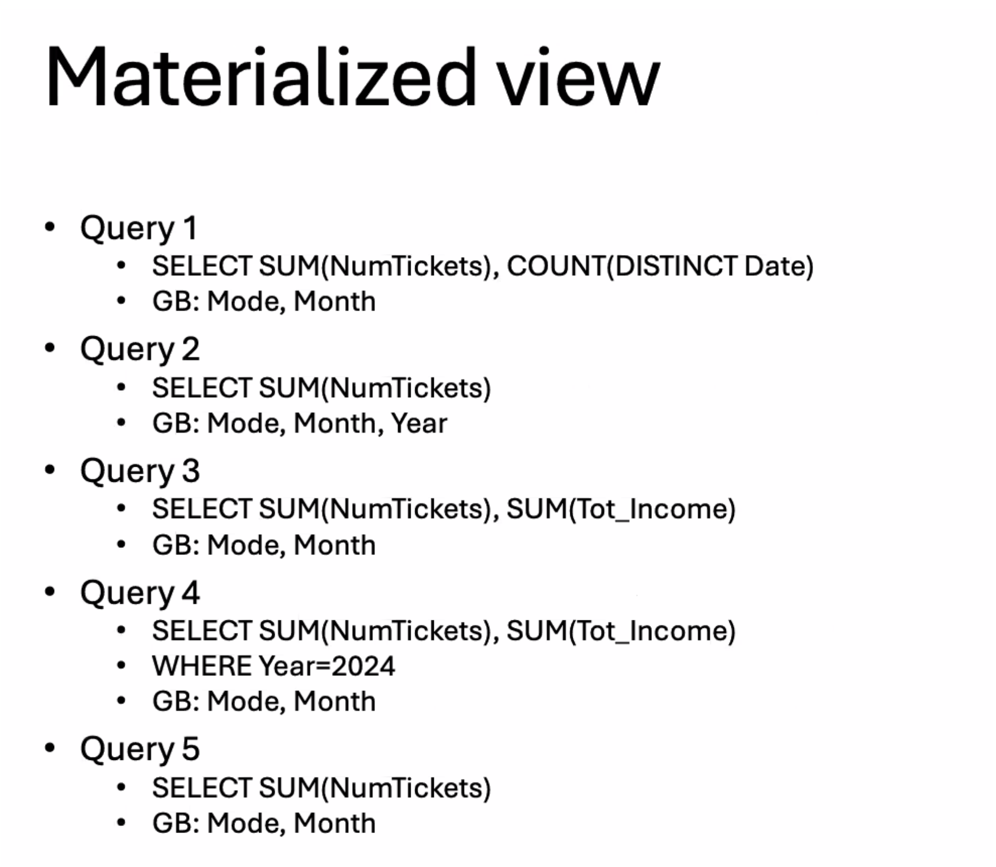
Considering travels from 2022, separately for each line and city, we analyze:

- the average travel duration,
- the total revenues generated by that city,
- the percentage of total revenue contributed by each line to the city's total, and
- assign a rank to each line within its city based on total line revenues generated in descending order

```
SELECT
city, line,
SUM(Total_duration)/SUM(#Tickets),
SUM(SUM(Total_Income)) over (PARTITION BY City)
100*SUM(Total_Income)/SUM(SUM(Total_Income))
over (PARTITION BY City),
RANK() over (PARTITION BY City ORDER BY
SUM(Total_Income) desc)
FROM TRAVELS TR, ROUTE R, TIME T
WHERE TR.CodT = T.CodT and TR.CodR = R.CodR AND
year >= 2022
GROUP BY city, line
```

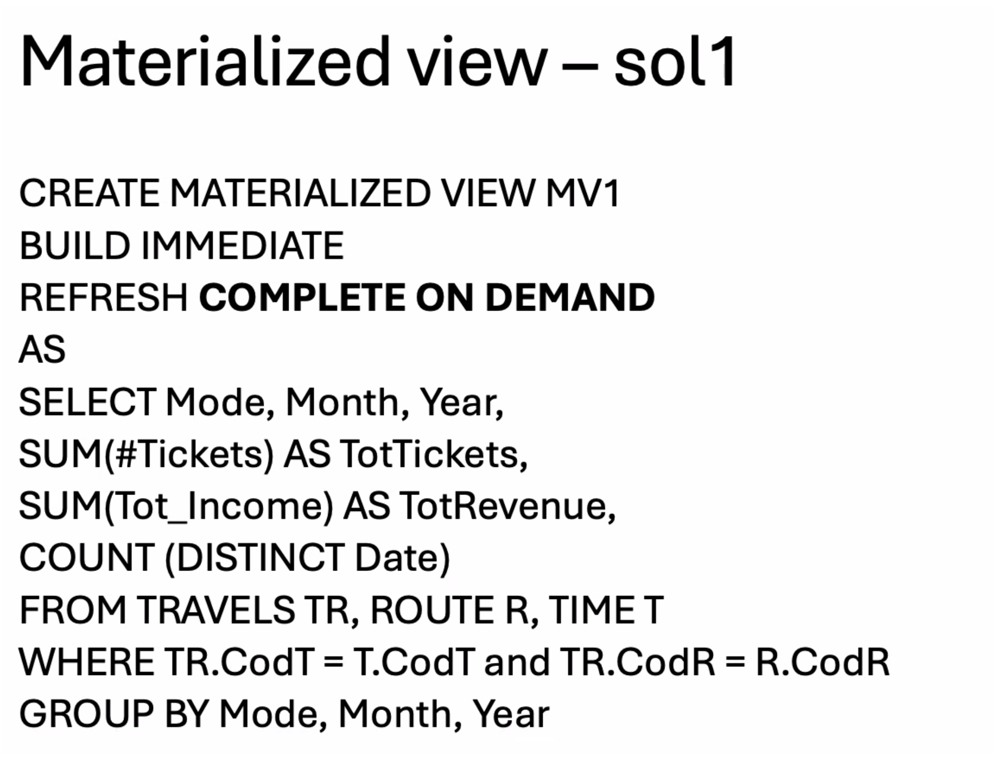
The text of the query is changed and instead of separately for Transportation Mode and City we are required to fecth the data separately for Line and City

Metarialized View



Materialized view

- Query 1
  - SELECT SUM(NumTickets), COUNT(DISTINCT Date)
  - GB: Mode, Month
- Query 2
  - SELECT SUM(NumTickets)
  - GB: Mode, Month, Year
- Query 3
  - SELECT SUM(NumTickets), SUM(Tot_Income)
  - GB: Mode, Month
- Query 4
  - SELECT SUM(NumTickets), SUM(Tot_Income)
  - WHERE Year=2024
  - GB: Mode, Month
- Query 5
  - SELECT SUM(NumTickets)
  - GB: Mode, Month



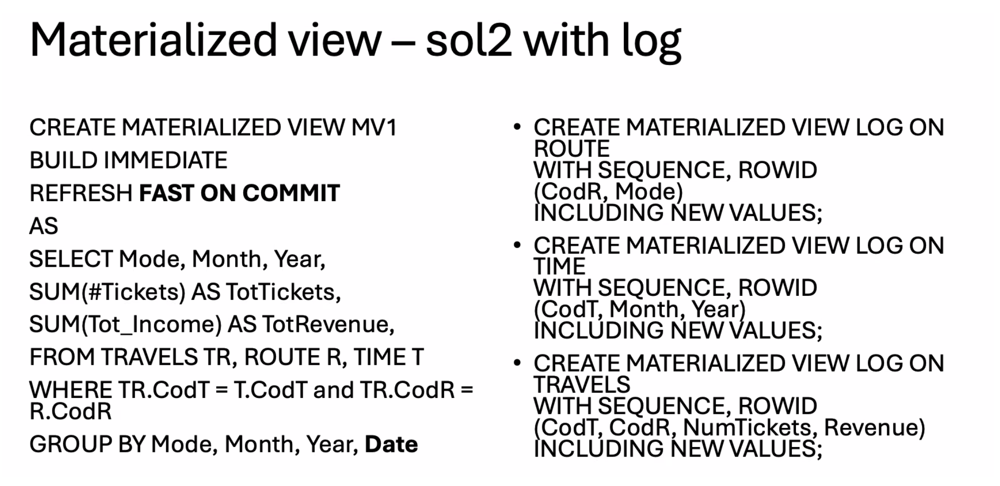
Materialized view – sol1

```
CREATE MATERIALIZED VIEW MV1
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
AS
SELECT Mode, Month, Year,
SUM(#Tickets) AS TotTickets,
SUM(Tot_Income) AS TotRevenue,
COUNT (DISTINCT Date)
FROM TRAVELS TR, ROUTE R, TIME T
WHERE TR.CodT = T.CodT and TR.CodR = R.CodR
GROUP BY Mode, Month, Year
```

REFRESH COMPLETE ON DEMAND is better than REFRESH FAST ON COMMIT because we have many many data in each trasaction. This way the mw is not updated after each commit but just after the user's explicit request.

SOLUTION 2 WITH REFRESH FAST OM COMMIT:

I cannot put the Count(Distinct Date) as measure but I have Date inside the GB. That way I can proceed counting. **I don't put the counting of Dates because it will slow down the processes since it would have to be recomputed from the ground up after every commit!**



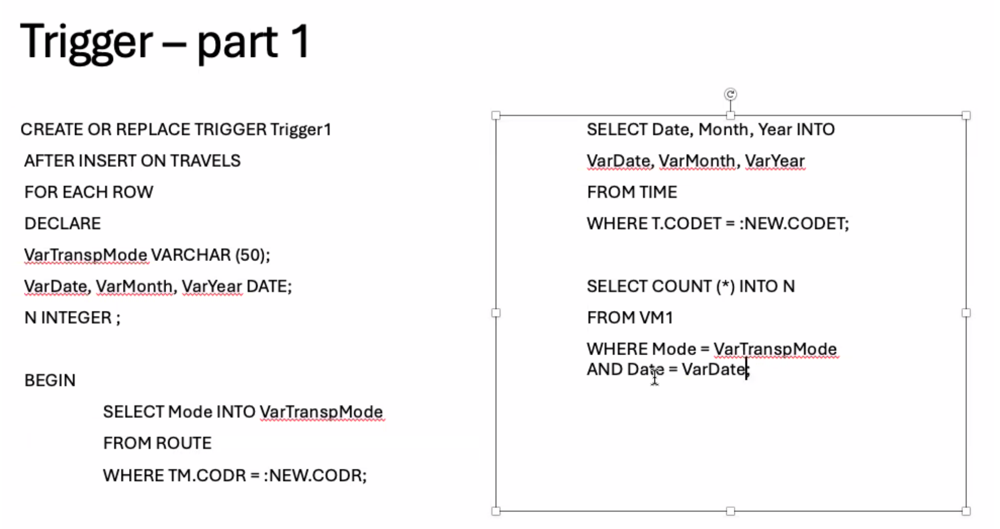
Materialized view – sol2 with log

```
CREATE MATERIALIZED VIEW MV1
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
AS
SELECT Mode, Month, Year,
SUM(#Tickets) AS TotTickets,
SUM(Tot_Income) AS TotRevenue,
FROM TRAVELS TR, ROUTE R, TIME T
WHERE TR.CodT = T.CodT and TR.CodR =
R.CodR
GROUP BY Mode, Month, Year, Date
```

- CREATE MATERIALIZED VIEW LOG ON ROUTE
  WITH SEQUENCE, ROWID
  (CodR, Mode)
  INCLUDING NEW VALUES;
- CREATE MATERIALIZED VIEW LOG ON TIME
  WITH SEQUENCE, ROWID
  (CodT, Month, Year)
  INCLUDING NEW VALUES;
- CREATE MATERIALIZED VIEW LOG ON TRAVELS
  WITH SEQUENCE, ROWID
  (CodT, CodR, NumTickets, Revenue)
  INCLUDING NEW VALUES;

<u>Trigger</u>

This version is related to the second version of the mv with REFRESH FAST ON COMMIT and without computing the DISTINCT(COUNT Date)



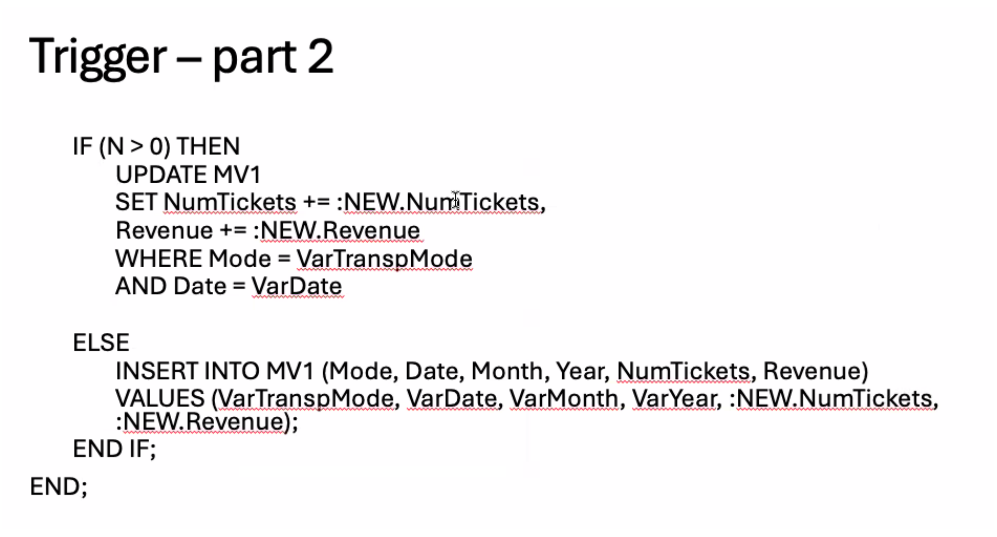
Trigger – part 1

```
CREATE OR REPLACE TRIGGER Trigger1
 AFTER INSERT ON TRAVELS
 FOR EACH ROW
 DECLARE
 VarTranspMode VARCHAR (50);
 VarDate, VarMonth, VarYear DATE;
 N INTEGER ;

 BEGIN
           SELECT Mode INTO VarTranspMode
           FROM ROUTE
           WHERE TM.CODR = :NEW.CODR;
```

```
SELECT Date, Month, Year INTO
VarDate, VarMonth, VarYear
FROM TIME
WHERE T.CODET = :NEW.CODET;

SELECT COUNT (*) INTO N
FROM VM1
WHERE Mode = VarTranspMode
AND Date = VarDate
```



Trigger – part 2

```
    IF (N > 0) THEN
        UPDATE MV1
        SET NumTickets += :NEW.NumTickets,
        Revenue += :NEW.Revenue
        WHERE Mode = VarTranspMode
        AND Date = VarDate

    ELSE
        INSERT INTO MV1 (Mode, Date, Month, Year, NumTickets, Revenue)
        VALUES (VarTranspMode, VarDate, VarMonth, VarYear, :NEW.NumTickets,
        :NEW.Revenue);
    END IF;
END;
```