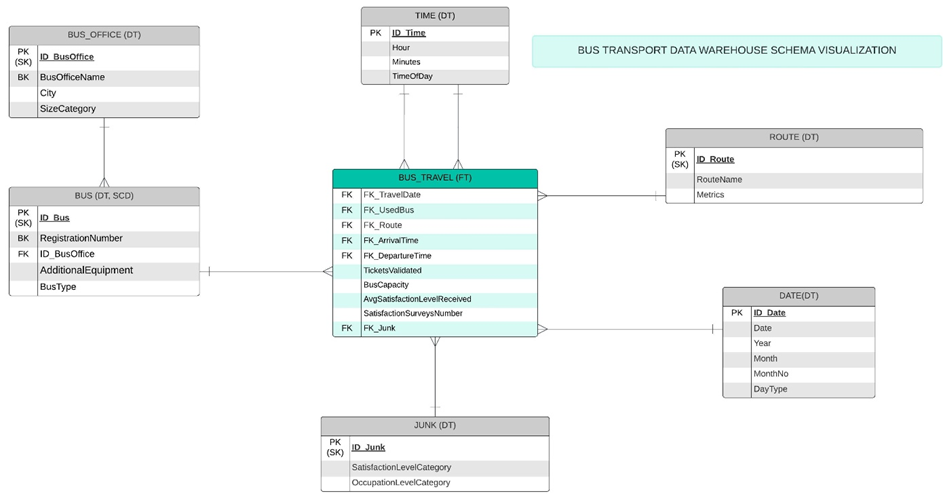
**Bus transport Warehouse Design in Hive**

Scenarios and explanation of the decisions made



**Competency questions and scenarios**

1. **What are the weekday routes with the highest number of tickets validated?(comparing two months to each other)**

**Scenario:** A transport manager is analyzing weekday bus operations to identify the most profitable or popular routes, in order to allocate more buses for these routes.

**Partitioning** by day\_type (weekday) and month\_no allows the query to efficiently filter for weekdays in given months.

**Bucketing** on route\_id helps speed up the join between travel and route tables.

SELECT r.route\_id, SUM(CASE WHEN d.month\_no = 4 THEN t.tickets\_validated ELSE 0 END) AS total\_tickets\_april, SUM(CASE WHEN d.month\_no = 5 THEN t.tickets\_validated ELSE 0 END) AS total\_tickets\_may, (SUM(CASE WHEN d.month\_no = 5 THEN t.tickets\_validated ELSE 0 END)-SUM(CASE WHEN d.month\_no = 4 THEN t.tickets\_validated ELSE 0 END)) AS tickets\_difference\_to\_prev\_month

FROM travel t

JOIN route r ON t.route\_id = r.route\_id

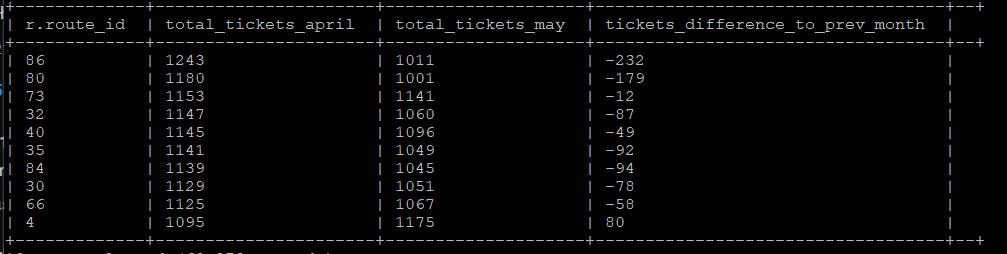
JOIN date\_dim d ON t.travel\_date = d.date\_id

WHERE d.month\_no IN (4, 5) AND d.day\_type= 'weekday'

GROUP BY r.route\_id

ORDER BY total\_tickets\_april DESC

LIMIT 10;



(case clause used in query: <https://stackoverflow.com/questions/31489073/hive-else-in-sum> )

1. **Which bus type has the highest average satisfaction level?**

**Scenario:** The management team wants to understand which type of buses provide the best customer experience to decide on future bus purchases.

**Partitioning** by bus\_type in the bus table allows efficient retrieval of bus types (on bus\_type we use static partitioning which ensures that each bus type has its own dedicated partition)

**Bucketing** by bus\_id helps speed up joins between the travel and bus tables.

SELECT b.bus\_type, AVG(t.avg\_satisfaction\_level\_received) AS avg\_satisfaction

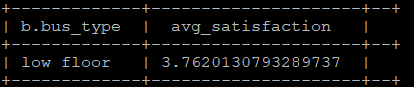
FROM travel t

JOIN bus b ON t.bus\_id = b.bus\_id

GROUP BY b.bus\_type

ORDER BY avg\_satisfaction DESC

LIMIT 1;



1. **What is the number of validated tickets for each bus comparing across weekdays and weekends in May?**

**Scenario:** A data analyst is looking at bus usage in May to see which buses are used more on weekdays than at weekends to spot which buses have higher demand on weekdays versus weekends.

**Partitioning** on by month\_no and day\_type for faster access only the relevant data for May and weekdays/weekends.

**Bucketing** on bus\_id in the travel table helps improve the efficiency of operations involving bus\_id.

SELECT t.bus\_id, d.day\_type, SUM(t.tickets\_validated) AS total\_validated\_tickets

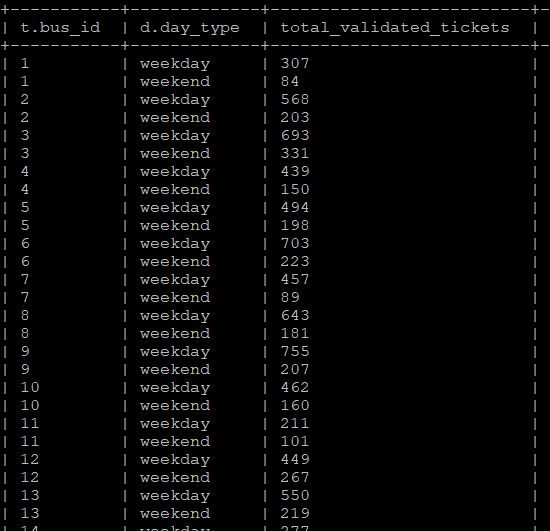
FROM travel t

JOIN date\_dim d ON t.travel\_date = d.date\_id

WHERE d.month\_no = 5 AND d.day\_type IN ('weekday', 'weekend')

GROUP BY t.bus\_id, d.day\_type

ORDER BY t.bus\_id, d.day\_type;



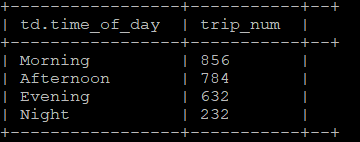
1. **What was the busiest time of the day for bus routes on weekends in April?**

**Scenario:** The operations team is making changes to the weekend bus schedule in April to make better use of resources and reduce crowdes.

**Partitioning** by day\_type (weekend) allows fast filtering for weekend records. Partitioning by month\_no helps quickly focus on April data.

**Bucketing** on route\_id helps optimize joins between travel and route tables.

SELECT td.time\_of\_day, COUNT(\*) AS trip\_num   
FROM travel t   
JOIN date\_dim d ON t.travel\_date = d.date\_id   
JOIN time\_dim td ON t.departure\_time = td.time\_id   
WHERE month\_no = 4 AND d.day\_type = 'weekend'   
GROUP BY td.time\_of\_day   
ORDER BY trip\_num DESC;



1. **How many satisfaction surveys were received for a given route comparing two months (April and May)?**

**Scenario:** The quality assurance team wants to check customer engagement by comparing the number of satisfaction surveys collected for a routes in April and May.

**Partitioning** by month\_no allows the query to quickly filter records for April and May.

**Bucketing** on route\_id helps optimize joins with the route table.

SELECT r.route\_name, SUM(CASE WHEN d.month\_no = 4 THEN t.satisfaction\_surveys\_number ELSE 0 END) as surveys\_april, SUM(CASE WHEN d.month\_no = 5 THEN t.satisfaction\_surveys\_number ELSE 0 END) as surveys\_may

FROM travel t

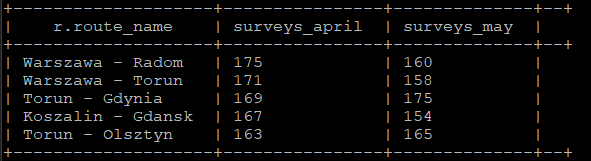
JOIN route r ON t.route\_id = r.route\_id

JOIN date\_dim d ON t.travel\_date = d.date\_id

GROUP BY r.route\_name

ORDER BY surveys\_april DESC

LIMIT 5;



1. **How satisfaction level of bus users changed over time?**

**Scenario**: The quality assurance team wants to see how (for example) recent changes in service affect customer satisfaction.

**Partitioning** by month\_no helps quickly retrieve data for specific months.

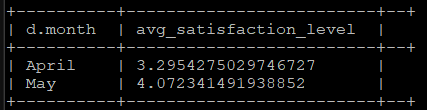
SELECT d.month, AVG(t.avg\_satisfaction\_level\_received) AS avg\_satisfaction\_level

FROM travel t

JOIN date\_dim d ON t.travel\_date = d.date\_id

GROUP BY d.month

ORDER BY d.month;



1. **How does the satisfaction level vary based on occupation level categories?**

**Scenario:** A quality assurance team evaluates how passenger satisfaction levels change based on bus crowd levels to determine whether overcrowding impacts user satisfaction and identify areas for capacity improvement.

The junk table, which contains occupation categories, is small, so partitioning or bucketing isn’t critical here.

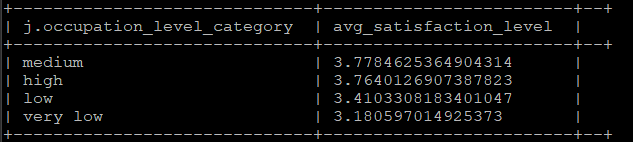
SELECT j.occupation\_level\_category, AVG(avg\_satisfaction\_level\_received) AS avg\_satisfaction\_level

FROM travel t

JOIN junk j ON t.junk\_id=j.junk\_id

GROUP BY j.occupation\_level\_category

ORDER BY avg\_satisfaction\_level DESC;



1. **What is the breakdown of travel counts by bus type that departures in the morning?**

(how many buses departured in the morning until now and what bus type was it)

**Scenario:** The transport planner analyses the number of buses of each type that departures in the morning to identify patterns of use and to ensure a balanced use of the different bus types.

**Partitioning** by bus\_type allows efficient filtering of travel records by bus type, partitioning by time\_of\_day (morning) filters morning travel data.

**Bucketing** by bus\_id can help improve performance when joining the travel table with the bus table.

SELECT b.bus\_type, COUNT(\*) travels\_num

FROM travel t

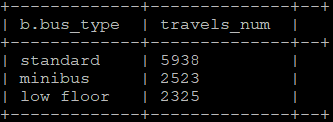
JOIN bus b ON t.bus\_id=b.bus\_id

JOIN time\_dim tdim ON t.departure\_time=tdim.time\_id

WHERE tdim.time\_of\_day = 'Morning'

GROUP BY b.bus\_type, tdim.time\_of\_day

ORDER BY travels\_num DESC;



1. **Which are the routes with the highest distance value?**

**Scenario:** Identifying long-distance routes for subsequent assessment of their viability, fuel consumption and resource requirements.

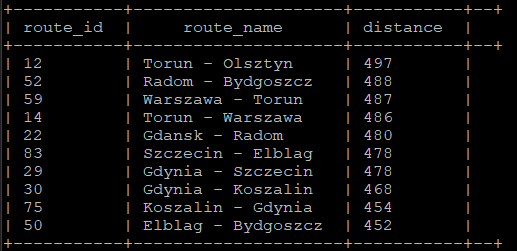
**Mapping** the distance\_km field inside metrics['distance\_km']. Because bucketing by route\_id ensures that the data related to each route is grouped together, querying the metrics['distance\_km'] map field becomes more efficient.

SELECT route\_id, route\_name, metrics['distance\_km'] AS distance

FROM route

ORDER BY distance DESC

LIMIT 10;



1. **What is the satisfaction level on weekend routes that have the highest avg duration? (in the selected month - April)**

**Scenario:** A quality control team looks at how customers feel about long weekend trips.

**Partitioning** by day\_type (weekend) and month\_no (April) allows for efficient filtering of month and day type.

**Bucketing** on route\_id speeds up the join with the route table.

**Mapping** the duration\_min in metrics for each route, and bucketing by route\_id helps optimize the retrieval of metrics['duration\_min']

SELECT t.route\_id, r.route\_name, AVG(r.metrics['duration\_min']) as avg\_duration,

AVG(t.avg\_satisfaction\_level\_received) AS avg\_satisfaction

FROM travel t

JOIN route r ON t.route\_id = r.route\_id

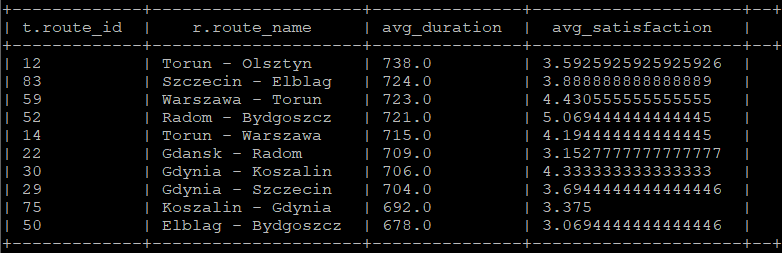
JOIN date\_dim d ON t.travel\_date = d.date\_id

WHERE d.month\_no = 4 AND d.day\_type = 'weekend'

GROUP BY t.route\_id, r.route\_name

ORDER BY avg\_duration DESC

LIMIT 10;



**1. Partitioning:**

Used for tables where queries often filter by specific columns like month\_no, day\_type, time\_of\_day, travel\_date, bus\_type. It limits the number of nodes that needs to be scanned when filtering some columns.

Partitions in tables are comparatively equal size (travel\_date in travel table as the schedule of buses does not vary significantly on different days) or the values possible for each field is limited (like month\_no, day\_type, time\_of\_day and bus\_type).

Static partitioning is used for bus\_type, time\_of\_day where we manually decide on number of partitions and to which partition data will be loaded, we need to be sure that we put data into right partition e.g. loading buses of only one type at once!

For dynamic paritioning (month\_no, day\_type, travel\_date) - partitions are created automatically depending on data we are inserting into, what for travel\_date field adding records while manually giving the date may be more complex process, probably this table will grow large in time.

**2. Bucketing:**

Bucket by route\_id, bus\_is travel table to improve performance of aggregations and joins. Partitioning here might lead to some larger and some smaller partitions as we assume that some routes have more travels – we use buckets to decompose dataset based on value from hash function. As the number of travels is around 20k (data of travels is from April) we can also imply that the number of records in the datawarehouse will arise at most to 240k – when storing data for one year, and we have 90 unique routes we decided on splitting data into 50 buckets.

Bucketing also on route\_id in routes table improves joins and searching on this column, we have 90 routes so splittinig into 10 buckets is reasonable (on average 9 routes per bucket). In queries the route\_id field (what is unique attribute) will be used often in queries (e.g. searching for specific routes usually combined with travel table) so map-side joins will work faster on bucketing tables.

Bucketing on bus\_id into 10 buckets and date\_id into 30 buckets – for joins with travel table.

No partitioning/bucketing on junk table as it has only 12 records.

**3. Data Format:**

ORC – used for routes and buses as it support complex datatypes (in routes table map, in bus array) as well as queries that are using aggregations (travel is join-heavy with other tables).

PARQUET – used for date, time, junk tables, frequently queried for filtering/grouping (day\_type, time\_of\_day). Those tables are realatively small tables. Parquet avoids reading unnecessary columns during query execution (columnar storage).

TEXTFILE – used for bus\_office table; as it may be easier to use for external systems. (reading data in Excel, Python or R) No partitioning here due to relatively small table.