**TOURISM KNOWLEDGE REPRESENTATION**

by

KAVYA PALIWAL - 20BRS1111

A project report submitted to

**Dr. RajaRajeshwari**

**SCHOOL OF Computer Science And Enginnering**

in partial fulfilment of the requirements for the course of

**CSE3056 – Knowledge Representation And Reasoning**

in

**B. Tech. COMPUTER SCIENCE ENGINEERING**

****

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**APRIL 2023**

**ABSTRACT**

Tourism knowledge representation refers to the process of capturing and organizing information about tourist destinations, attractions, and experiences in a way that can be easily understood and used by both tourists and tourism professionals. This involves identifying and categorizing different types of knowledge, such as historical and cultural information, geographical data, and tourist preferences and interests.

In order to represent tourism knowledge, various techniques and methods can be used, such as semantic web technologies, geographic information systems (GIS), and artificial intelligence (AI) algorithms. These approaches can help to identify and extract relevant information, organize it in a structured way, and provide personalized recommendations to tourists based on their interests and preferences.

Overall, tourism knowledge representation plays a critical role in enhancing the tourist experience and supporting sustainable tourism development. By providing accurate and comprehensive information to tourists, it can help to increase their satisfaction and reduce the negative impacts of tourism on the environment and local communities. At the same time, it can also benefit tourism professionals by improving their ability to manage tourism resources and provide high-quality services to visitors.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. RajaRajeshwari,** for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We express our thanks to our Head of the Department **Dr. Rekha D** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends, team-mates for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**Kavya**

**DESCRIPTION OF THE SYSTEM**

A tourism knowledge representation system is a system that represents and organizes tourism-related knowledge in a structured and machine-readable format. The system aims to capture the relationships and connections between different entities, such as destinations, accommodations, restaurants, activities, and transportation options.

The system typically consists of a knowledge graph, which is a graphical representation of the relationships between entities, attributes, and relationships. The entities are represented as nodes, and the relationships between them are represented as edges. The attributes of the entities are also represented as nodes, and they are connected to the corresponding entity nodes through edges.

The tourism knowledge representation system can be used for a variety of applications, such as trip planning, personalized recommendations, and tourism management. The system can also incorporate user-generated content, such as reviews and ratings, to provide more accurate and relevant information to users.

The tourism knowledge representation system may also use natural language processing techniques to extract information from unstructured text data, such as online travel blogs, reviews, and social media posts. This information can then be added to the knowledge graph to enhance its accuracy and relevance.

Overall, a tourism knowledge representation system is a valuable tool for tourism professionals, travelers, and other stakeholders in the tourism industry. By capturing and organizing tourism-related knowledge in a structured format, the system can provide insights and recommendations that can improve the tourism experience for everyone involved.

**SCENARIOS:**

1. **Geographic Information System (GIS) mapping:** Touristknowledge representation could involve the use of GIS mapping software to create digital maps that highlight tourist attractions, points of interest, and landmarks in a particular region. This kind of representation could include interactive features, such as clickable icons or pop-up windows that provide information about each location, its history, and its significance.
   1. **Environemt**

This scenario could be used in a variety of tourist environments, such as city tours, hiking trails, or theme parks. For example, a city tour could use a GIS map to highlight popular landmarks and provide information about each one, while a hiking trail map could show the trail route and points of interest along the way. Similarly, a theme park could use GIS mapping to help visitors navigate the park and find their favorite rides and attractions.

* 1. **Different States**
* The location of the tourist
* The location of nearby tourist attractions
* The tourist's preferences and interests
* The time of day or day of the week
* The weather conditions
  1. **Input Values**
* The tourist's current location (e.g. GPS data)
* The tourist's destination or desired attraction
* The tourist's interests and preferences (e.g. cultural, historical, nature, food)
* The tourist's transportation mode (e.g. walking, biking, public transportation, driving)
* The tourist's time constraints (e.g. limited time, specific hours of operation)
  1. **Action Parameters**
* Recommended tourist attractions based on the tourist's location and preferences
* The shortest or most scenic route to a desired attraction
* Real-time traffic information and alternate routes
* Navigation instructions and turn-by-turn directions
* Information about nearby restaurants, cafes, and other amenities

1. **Augmented Reality (AR):** Tourist knowledge representation could involve the use of augmented reality (AR) applications that superimpose digital information over real-world locations. For example, a tourist could use an AR app on their smartphone to view a famous landmark and see information about its history, construction, or cultural significance. The app could also provide suggestions for nearby restaurants, hotels, or other attractions.
   1. **Environemt**

This scenario could be used in a variety of tourist environments, such as museums, historical sites, or natural parks. For example, a museum could use an AR app to provide visitors with additional information about the exhibits, while a historical site could use AR to recreate historical events or show how the site looked in the past. Similarly, a natural park could use AR to highlight flora and fauna, or to provide guided tours of the park's trails and features.

* 1. **Different States**
* The tourist's location
* The location of nearby tourist attractions
* The tourist's preferences and interests
* The direction and orientation of the tourist's device
* The sensor data from the tourist's device (e.g. camera, accelerometer)
  1. **Input Values**
* The tourist's location (e.g. GPS data)
* The tourist's preferences and interests (e.g. cultural, historical, nature, food)
* The tourist's device type and capabilities (e.g. camera, sensors)
* The tourist's input and feedback (e.g. clicking on an attraction, providing a rating)
  1. **Action Parameters**
* Real-time information and details about nearby tourist attractions
* Interactive 3D models or virtual reconstructions of historical or cultural sites
* Suggestions for related attractions or experiences based on the tourist's preferences and feedback
* Integration with social media for sharing experiences and recommendations
* Personalized recommendations and notifications based on the tourist's previous interactions and behavior

1. **Narrative-based storytelling:** Tourist knowledge representation could involve the use of storytelling techniques to convey information about a particular location or region. This could take the form of audio tours, podcasts, or interactive exhibits that use multimedia content to tell the story of a place. By weaving together historical, cultural, and environmental information, these narratives can create a deeper understanding of a location and its significance.
   1. **Environemt**

This scenario could be used in a variety of tourist environments, such as museums, historical sites, or cultural centers. For example, a museum exhibit could use multimedia content and narration to tell the story of a particular artifact or era in history, while a historical site could use audio tours to guide visitors through the site and provide context for its significance. Similarly, a cultural center could use interactive exhibits and storytelling to share information about local customs, traditions, and beliefs.

* 1. **Different States**
* The tourist's location
* The location of nearby tourist attractions
* The historical and cultural significance of the location
* The tourist's level of familiarity with the location
* The availability of multimedia content (e.g. audio guides, interactive exhibits)
  1. **Input Values**
* The tourist's location (e.g. GPS data)
* The tourist's level of familiarity with the location and its history
* The tourist's preferences and interests (e.g. cultural, historical, nature, food)
* The tourist's language and cultural background
* The tourist's mode of interaction (e.g. listening to an audio guide, watching a video, interacting with an exhibit)
  1. **Action Parameters**
* Audio and visual content that explains the historical or cultural significance of a location or attraction
* Interactive exhibits or multimedia content that engage the tourist and provide a more immersive experience
* Gamification elements that encourage exploration and learning
* Personalized recommendations and notifications based on the tourist's previous interactions and behavior
* Integration with social media for sharing experiences and recommendations

**SCENARIOS:**

1. **Planning the route and itinerary:** The first step in booking and executing a road trip independently is to plan your route and itinerary. This involves researching the destinations you want to visit, the best time to go, and the driving distance between each location. You'll need to decide on the duration of your trip, the number of days you'll spend in each destination, and the activities you want to do. You can use online maps, travel guides, and blogs to help you plan your trip.

* 1. FOL
     1. If a destination is on the route, then it must be visited: ∀d,r (onRoute(d,r) → visit(d))
     2. A route must start and end at the same destination:

∀d,r (start(r)=d ∧ end(r)=d)

* + 1. If an attraction is at a destination, then it must be visited when the destination is visited:

∀d,a (at(d,a) → visit(d) ∧ visit(a)) 4. A destination can only be visited once:

∀d (visit(d) → ¬∃r1,r2 (onRoute(d,r1) ∧ onRoute(d,r2) ∧ r1≠r2))

* + 1. The total distance of the route must not exceed a certain limit: ∑r distance(r) ≤ max\_distance
    2. A minimum amount of time must be spent at each destination:

∀d (visit(d) → timeSpent(d) ≥ min\_time)

* + 1. If an attraction is on the route, then it must be visited:

∀a,r (onRoute(a,r) → visit(a))

* 1. Prolog
     + route(X,Y) :- connection(X,Y).

This statement defines a route between two places X and Y if there is a direct connection between them.

* + - route(X,Y) :- connection(X,Z), route(Z,Y).

This statement defines a route between two places X and Y if there is a direct connection between X and Z and there is a route between Z and Y.

* + - itinerary(X,Y) :- route(X,Y).

This statement defines an itinerary between two places X and Y if there is a direct route between them.

* + - itinerary(X,Y) :- route(X,Z), itinerary(Z,Y).

This statement defines an itinerary between two places X and Y if there is a route between X and Z and an itinerary between Z and Y.

* + - distance(X,Y,D) :- connection(X,Y,D).

This statement defines the distance between two places X and Y if there is a direct connection between them with a distance of D.

* + - distance(X,Y,D) :- connection(X,Z,D1), distance(Z,Y,D2), D is D1 + D2. This statement defines the distance between two places X and Y if there is a connection between X and Z with a distance of D1 and a distance between Z and Y with a distance of D2, and the total distance between X and Y is D1 + D2.

* + - cost(X,Y,C) :- connection(X,Y,C,\_).

This statement defines the cost of traveling between two places X and Y if there is a direct connection between them with a cost of C.

* + - cost(X,Y,C) :- connection(X,Z,C1,\_), cost(Z,Y,C2), C is C1 + C2.

This statement defines the cost of traveling between two places X and Y if there is a connection between X and Z with a cost of C1 and a cost of traveling between Z and Y with a cost of C2, and the total cost of traveling between X and Y is C1 + C2.

* 1. Knowledge Graph using Neo4j

|  |
| --- |
| 2. from neo4j import GraphDatabase, basic\_auth from graphviz import Digraph # Define Neo4j credentials uri = "bolt://localhost:7687" user = "neo4j"  password = "123456kav"    # Connect to Neo4j  driver = GraphDatabase.driver(uri, auth=basic\_auth(user, password))  # Define a function to create the graph def create\_graph(tx):  # Define the domain and predicates  domains = ["Destinations", "Attractions", "Routes"]  predicates = ["connects", "distance", "time", "has\_attraction"]  # Create the domain and predicates nodes for domain in domains:  tx.run("MERGE (:Domain {name: $name})", name=domain) for predicate in predicates: |

|  |
| --- |
| tx.run("MERGE (:Predicate {name: $name})", name=predicate)  # Define the facts about the domain facts = [  {"source": "Delhi", "dest": "Manali", "distance": 1055,  "time": 25},  {"source": "Manali", "dest": "Leh", "distance": 474, "time":  10}, {"source": "Leh", "dest": "Srinagar", "distance": 1030, "time": 23},  {"source": "Srinagar", "dest": "Jammu", "distance": 268,  "time": 7},  {"source": "Jammu", "dest": "Delhi", "distance": 586, "time":  13},  {"source": "Manali", "attraction": "Rohtang Pass"},  {"source": "Manali", "attraction": "Solang Valley"},  {"source": "Leh", "attraction": "Pangong Tso Lake"},  {"source": "Srinagar", "attraction": "Dal Lake"} ]    # Create the nodes and relationships for the facts for fact in facts: if "dest" in fact:  tx.run(  "MERGE (d:Destination {name: $name})"  "MERGE (p:Predicate {name: 'connects'})"  "MERGE (s:Destination {name: $source})"  "MERGE (s)-[:connects {distance: $distance, time:  $time}]->(d)"  , name=fact["dest"], source=fact["source"], distance=fact["distance"], time=fact["time"]  ) elif "attraction" in fact:  tx.run(  "MERGE (a:Attraction {name: $name})"  "MERGE (p:Predicate {name: 'has\_attraction'})"  "MERGE (d:Destination {name: $source})"  "MERGE (d)-[:has\_attraction]->(a)"  , name=fact["attraction"], source=fact["source"]  )    # Define a function to query the graph for the shortest path def shortest\_path(tx, source, destination):  result = tx.run(  "MATCH (s:Destination {name: $source})-[:connects\*]-  >(d:Destination {name: $destination}), "  "p = shortestPath((s)-[:connects\*]->(d)) "  "WITH REDUCE(dist = 0, r IN relationships(p) | dist +  r.distance) AS distance, "  "REDUCE(time = 0, r IN relationships(p) | time + r.time) AS time, "  "[n IN nodes(p) | n.name] AS path "  "RETURN path, distance, time "  "ORDER BY distance ASC LIMIT 1"  , source=source, destination=destination  )  return result.single()    # Connect to Neo4j and create the graph with driver.session() as session:  session.execute\_write(create\_graph) |

|  |
| --- |
| # Query the graph for the shortest path with driver.session() as session:  result = session.execute\_read(shortest\_path, "Delhi", "Leh") print(f"The shortest path from Delhi to Leh is:  {result['path']}")  print(f"The distance is: {result['distance']} km") print(f"The travel time is: {result['time']} hours")  # Define a function to visualize the graph using Graphviz def visualize\_graph(tx):  result = tx.run(  "MATCH (n)-[r]->(m) RETURN n.name AS source, r.name AS predicate, m.name AS destination"  ) graph = Digraph() for record in result: graph.edge(record["source"], record["destination"], label=record["predicate"]) return graph    # Visualize the graph using Graphviz def visualize\_graph(tx, filename):  dot = Digraph(comment='Destinations and Attractions')  result = tx.run("MATCH (d:Destination) RETURN d.name") for record in result:  dot.node(record["d.name"], record["d.name"])  result = tx.run("MATCH (a:Attraction) RETURN a.name, a.type") for record in result: if record["a.type"] == "sightseeing":  dot.node(record["a.name"], record["a.name"], shape='box') else:  dot.node(record["a.name"], record["a.name"])    result = tx.run("MATCH (:Destination)-[r:ROUTE]->(:Destination) RETURN r.distance") for record in result:  dot.attr('edge', {'label': str(record["r.distance"])}) dot.edge(record.start\_node["name"], record.end\_node["name"])  result = tx.run("MATCH (:Destination)-[r:ATTRACTION]-  >(:Attraction) RETURN r") for record in result:  dot.edge(record.start\_node["name"], record.end\_node["name"], label=record["r"])  dot.render(filename, view=True)  with driver.session() as session:  session.execute\_read(visualize\_graph, "destinations.gv")  # Add dummy nodes and relationships with driver.session() as session: session.run("CREATE (:Dummy)")  session.run("MATCH (n:Destination) WHERE NOT ()-[:ROUTE]->(n) CREATE (d:Dummy)-[:ROUTE {distance: 0}]->(n)")  session.run("MATCH (n:Attraction) WHERE NOT ()-[:ATTRACTION]->(n) CREATE (d:Dummy)-[:ATTRACTION {type: 'dummy'}]->(n)")  session.run("MATCH (n:Attraction) WHERE NOT (n)-[:ATTRACTION]->() |

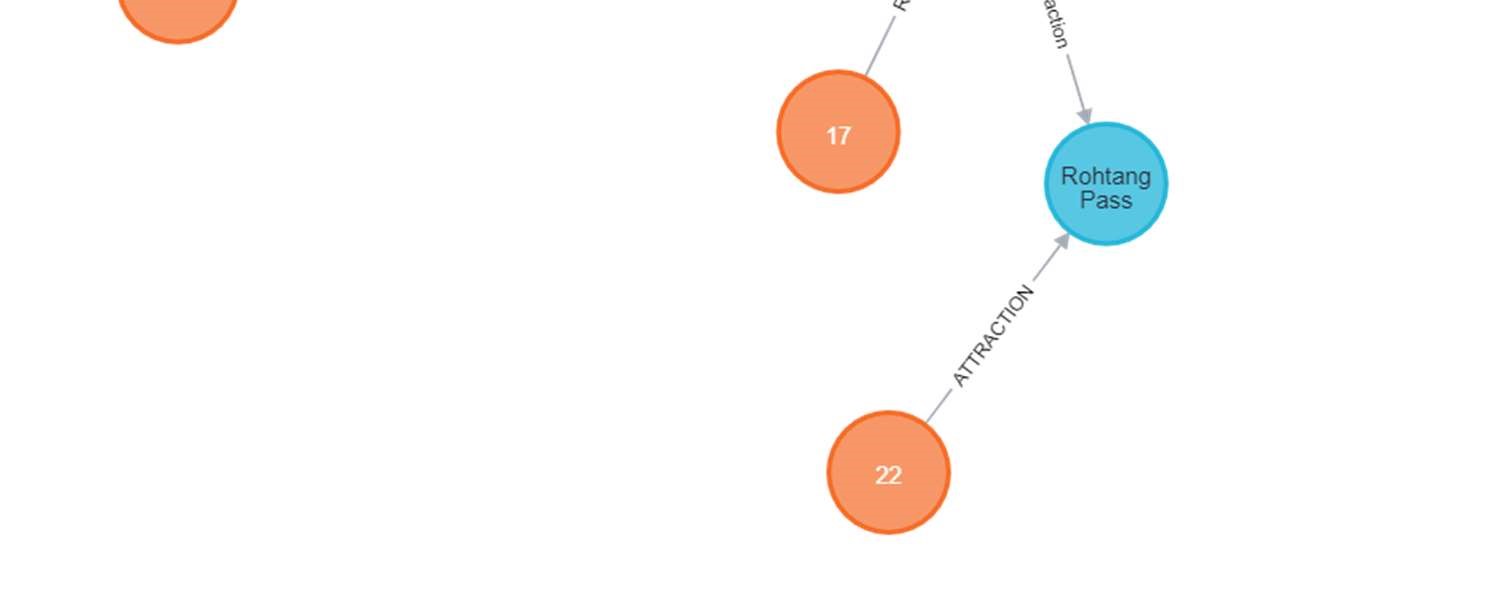
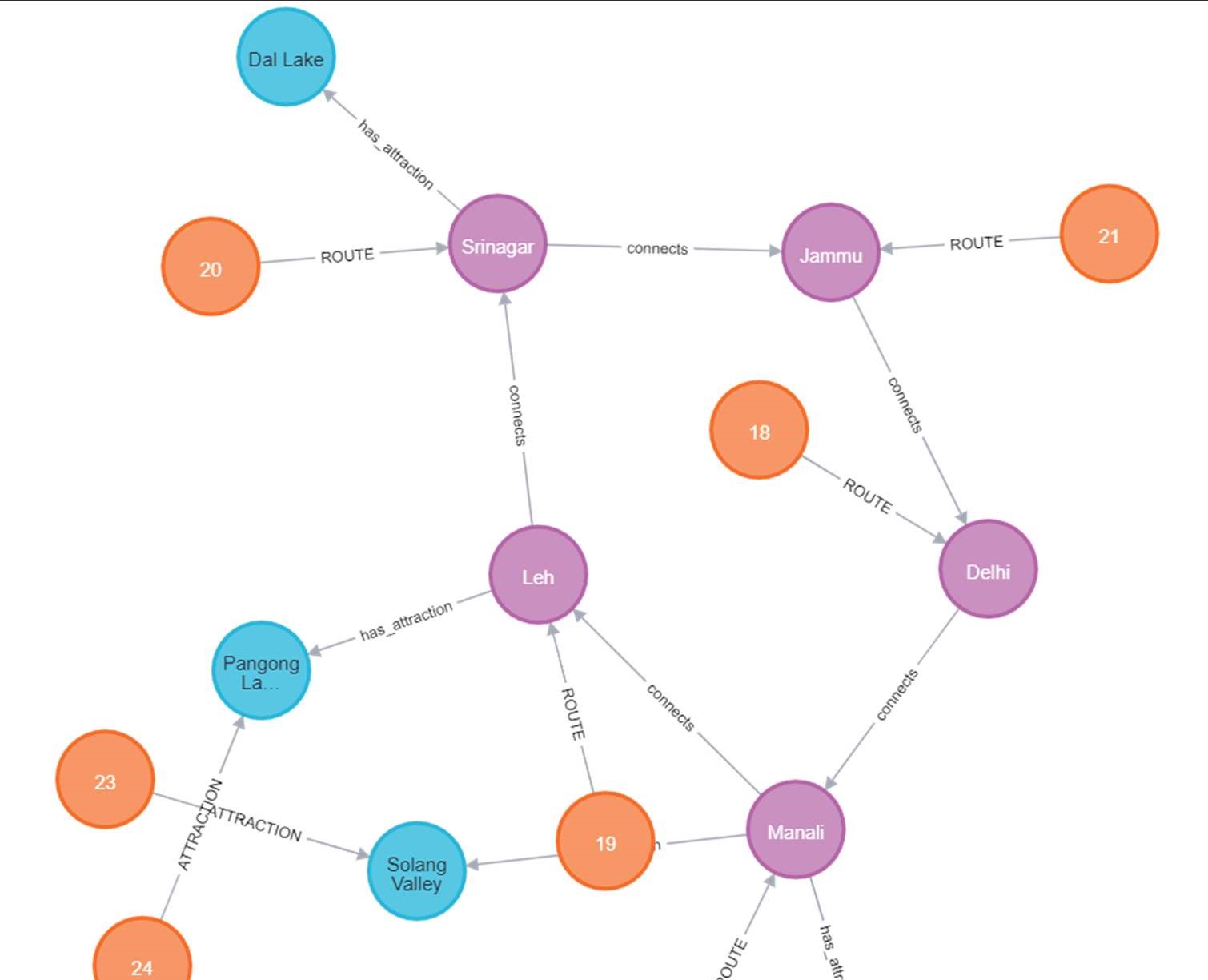
CREATE (n)-[:ATTRACTION {type: 'dummy'}]->(d:Dummy)")

session.run("MATCH (n:Dummy) WHERE NOT ()-[:ROUTE]->(n) CREATE

(n)-[:ROUTE {distance: 0}]->(d:Dummy)")

session.run("MATCH (n:Dummy) WHERE NOT (n)-[:ATTRACTION]->()

CREATE (d:Dummy)-[:ATTRACTION {type: 'dummy'}]->(n)")



* 1. Inferences
     + If there is a direct road from city A to city B, it is possible to travel from A to B by car.
     + If there is a direct train connection from city A to city B, it is possible to travel from A to B by train.
     + If there is a direct flight connection from city A to city B, it is possible to travel from A to B by plane.
     + If there is a direct bus connection from city A to city B, it is possible to travel from A to B by bus.
     + If there is a direct ferry connection from city A to city B, it is possible to travel from A to B by ferry.
     + If a city is a popular tourist destination, there are likely to be several hotels and other accommodations available there.
     + If a city is a major transportation hub, it is likely to have connections to many other cities by various means of transportation.

1. **Booking accommodations and transportation:** Once you've planned your itinerary, you'll need to book accommodations and transportation. This involves researching and booking hotels, rental cars, or other modes of transportation such as trains or buses. You can use online booking platforms or contact the accommodations or transportation companies directly to make your reservations. Be sure to read the terms and conditions carefully, and ask any questions you may have before making a booking.

* 1. FOL
     + ∀x,y,z [booking(x,y,z) ↔ (hotel(x) ∨ car\_rental(x) ∨ transportation(x)) ∧ city(y) ∧ date(z)]
     + ∀x,y,z [hotel(x) → (name(x) = y ∧ city(x) = z)]
     + ∀x,y,z [car\_rental(x) → (name(x) = y ∧ city(x) = z)]
     + ∀x,y,z [transportation(x) → (mode(x) = y ∧ city(x) = z)]
     + ∀x,y [contact(x,y) ↔ (hotel(x) ∨ car\_rental(x) ∨ transportation(x)) ∧ company(y)]
     + ∀x,y,z [reservation(x,y,z) → booking(x,y,z)]
     + ∀x,y [read\_terms(x,y) ↔ (hotel(x) ∨ car\_rental(x) ∨ transportation(x)) ∧ terms\_and\_conditions(y)]
     + ∀x,y,z [ask\_question(x,y,z) → (hotel(x) ∨ car\_rental(x) ∨ transportation(x)) ∧ question(y) ∧ answer(z)]

These statements define a few concepts such as booking, hotel, car rental, transportation, city, date, company, reservation, terms and conditions, and questions/answers. They also define relationships between these concepts, such as a booking requiring a city and date, a hotel or car rental having a name and city, a contact being between a company and a hotel/car rental/transportation provider, a reservation being a type of booking, and reading terms and conditions or asking questions being associated with a hotel/car rental/transportation provider.

* 1. Prolog

//Accommodation options accommodation(hotel). accommodation(hostel). accommodation(apartment). accommodation(villa).

//Transportation options transportation(rental\_car).

transportation(train). transportation(bus). transportation(flight). //Book accommodation book\_accommodation(AccommodationType, OnlineBooking) :- accommodation(AccommodationType),

(OnlineBooking = true ; OnlineBooking = false).

//Book transportation book\_transportation(TransportationType, OnlineBooking) :- transportation(TransportationType),

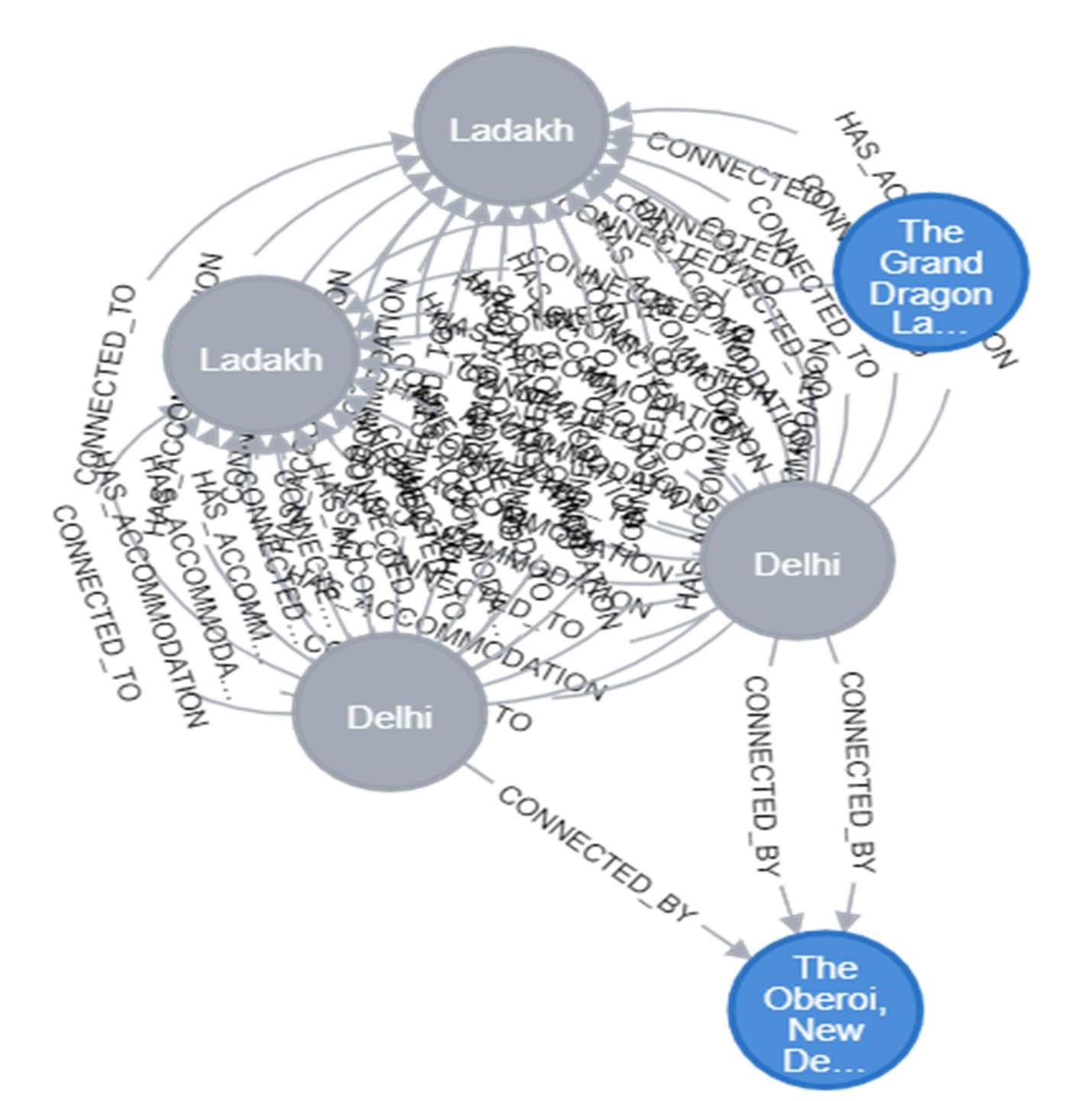
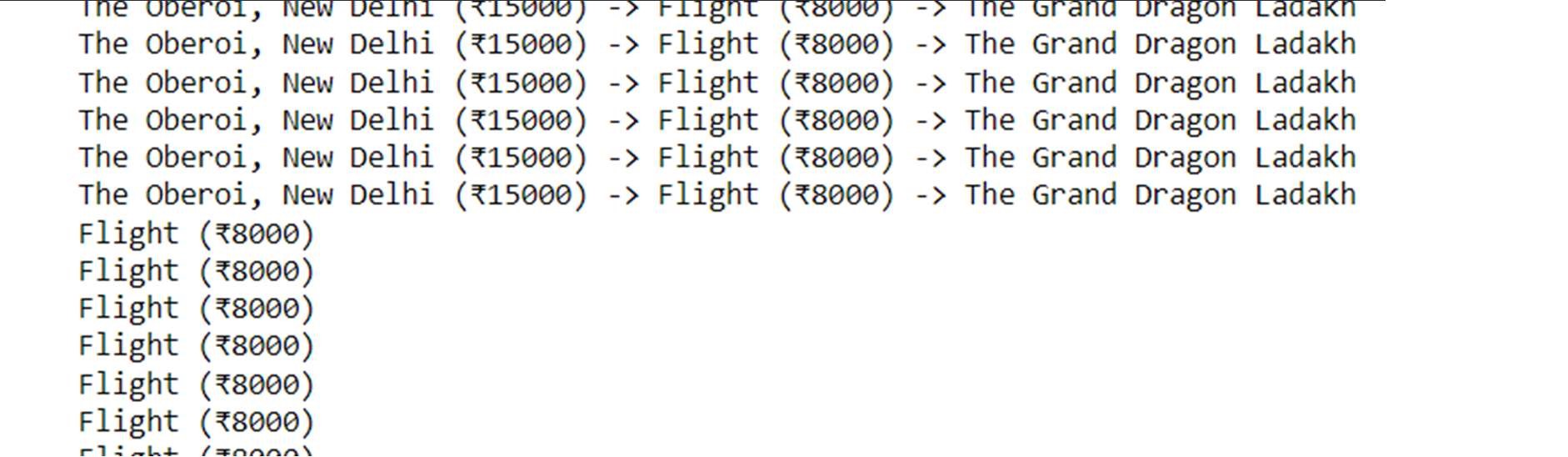
(OnlineBooking = true ; OnlineBooking = false).

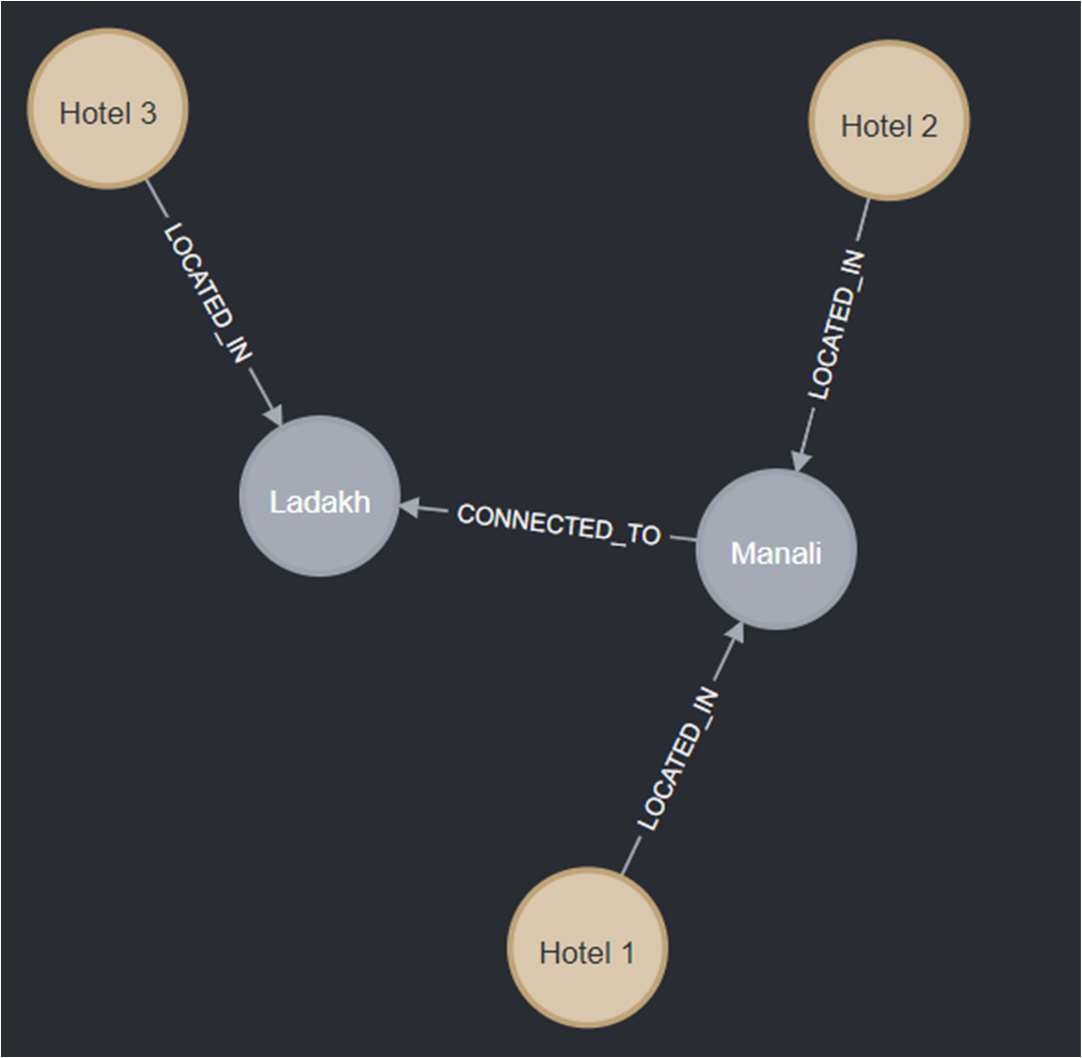
//Read terms and conditions read\_terms\_and\_conditions(Terms) :- Terms = true.

//Ask questions ask\_questions(Questions) :- Questions = true.

2.3 Knowledge Graph using Neo4j

|  |
| --- |
| from neo4j import GraphDatabase  # Connect to the Neo4j database uri = "bolt://localhost:7687" user = "neo4j" password = "123456kav"  driver = GraphDatabase.driver(uri, auth=(user, password))  # Define the Cypher queries  create\_graph = """  MERGE (d:City {name: 'Delhi'})  MERGE (l:City {name: 'Ladakh'})  MERGE (h1:Hotel {name: 'The Oberoi, New Delhi', price: 15000})  MERGE (h2:Hotel {name: 'The Grand Dragon Ladakh', price: 12000})  MERGE (t1:Transportation {type: 'Flight', price: 8000}) MERGE (t2:Transportation {type: 'Train', price: 6000})  MERGE (d)-[:CONNECTED\_BY]->(h1)-[:CONNECTED\_TO]->(t1)-  [:CONNECTED\_TO]->(h2)-[:CONNECTED\_BY]->(l) """  query\_hotels = """  MATCH (d:City {name: 'Delhi'})-[:CONNECTED\_BY]->(h:Hotel)-  [:CONNECTED\_TO]->(t:Transportation)-[:CONNECTED\_TO]->(h2:Hotel)-  [:CONNECTED\_BY]->(l:City {name: 'Ladakh'})  RETURN h.name AS hotel, h.price AS price, t.type AS transport\_type,  t.price AS transport\_price, h2.name AS hotel2  """  query\_transportation = """  MATCH (d:City {name: 'Delhi'})-[:CONNECTED\_BY]->(h:Hotel)-  [:CONNECTED\_TO]->(t:Transportation)-[:CONNECTED\_TO]->(h2:Hotel)-  [:CONNECTED\_BY]->(l:City {name: 'Ladakh'})  RETURN t.type AS transport\_type, t.price AS price """ |
| # Create the graph with driver.session() as session:  session.run(create\_graph)    # Query the hotels and transportation options with driver.session() as session:  result = session.run(query\_hotels) for record in result:  print(f"{record['hotel']} (₹{record['price']}) -> {record['transport\_type']} (₹{record['transport\_price']}) -> {record['hotel2']}")  result = session.run(query\_transportation) for record in result:    print(f"{record['transport\_type']} (₹{record['price']})") |





2.4 New Inferences

* Online booking platforms and direct communication with accommodations and transportation companies are the two methods for making reservations.
* Reading the terms and conditions carefully before making a booking is important to avoid any misunderstandings or issues.
* Asking questions and seeking clarification from accommodations and transportation companies before making a booking can help ensure that your needs and preferences are met.
* Booking accommodations and transportation in advance can help ensure availability and possibly result in lower prices.
* Considering factors such as location, amenities, and price can help in making informed decisions while booking accommodations.
* Factors such as duration of travel, budget, and preferred mode of transportation can influence the decision-making process while booking transportation.

1. **Managing the logistics:** During the road trip, you'll need to manage the logistics of your trip. This involves navigating the route, managing your budget, and dealing with any unexpected issues such as weather conditions, road closures, or car trouble. You'll need to ensure that you have enough cash or credit cards to cover your expenses, and be prepared for emergencies. It's a good idea to have a backup plan in case something goes wrong, such as having a list of alternative accommodations or routes. With careful planning and preparation, you can enjoy a memorable and stress-free road trip.

* 1. FOL
     + For all road trips, managing logistics is a requirement.
     + Managing logistics includes navigating the route, managing budget, and dealing with unexpected issues.
     + In order to manage logistics, it is necessary to have sufficient funds or credit cards.
     + Being prepared for emergencies is also a necessary part of managing logistics during a road trip.

* 1. Prolog

3.2.1.1 managing\_logistics(RoadTrip).

3.2.1.2 includes(RoadTrip, navigate\_route).

3.2.1.3 includes(RoadTrip, manage\_budget).

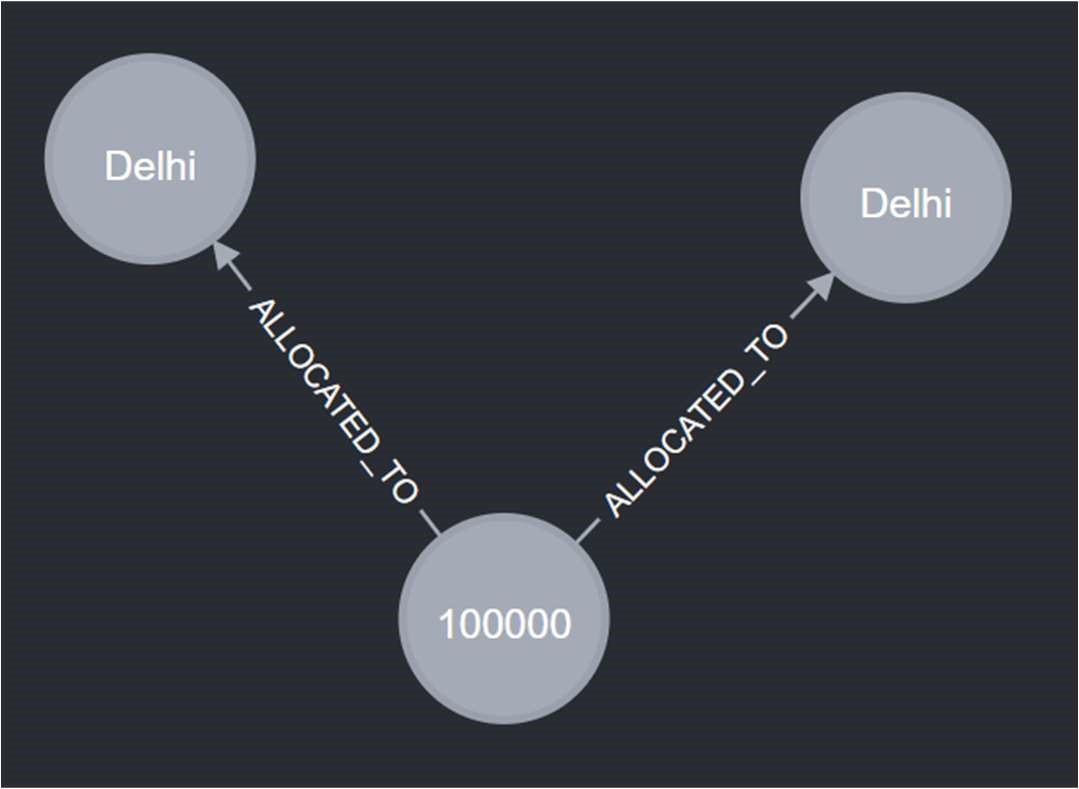
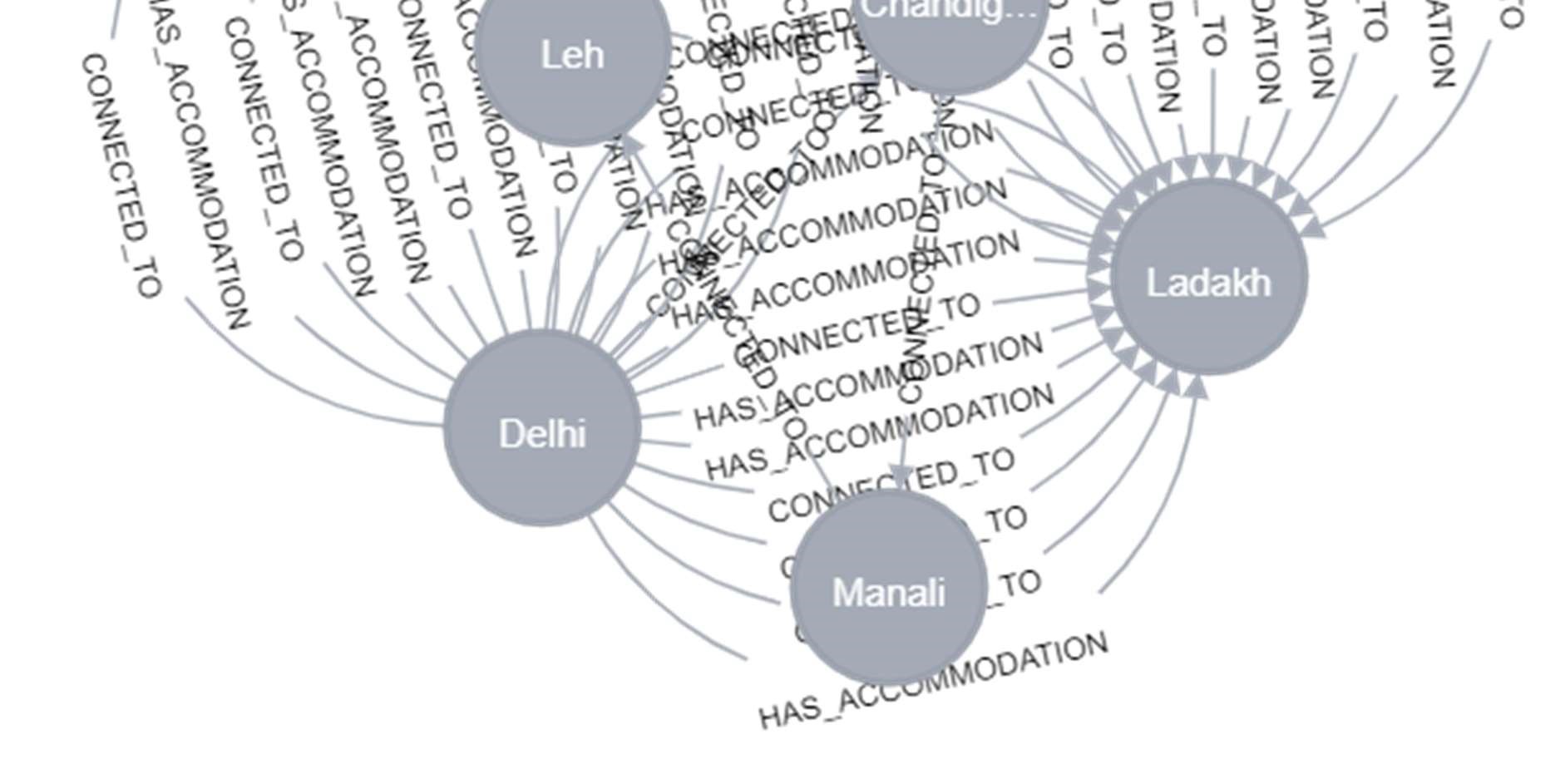
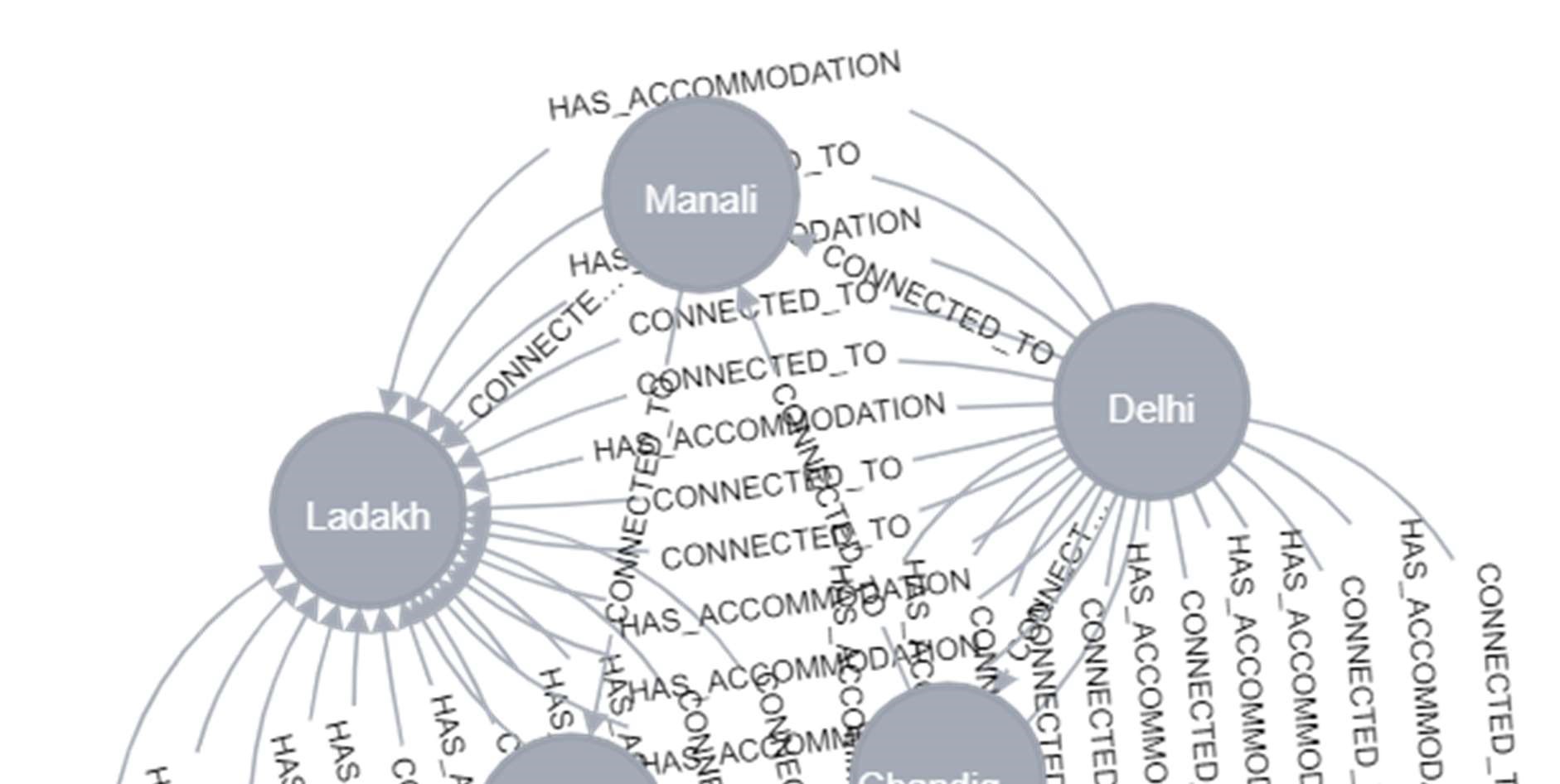
3.2.1.4 includes(RoadTrip, deal\_with\_issues).

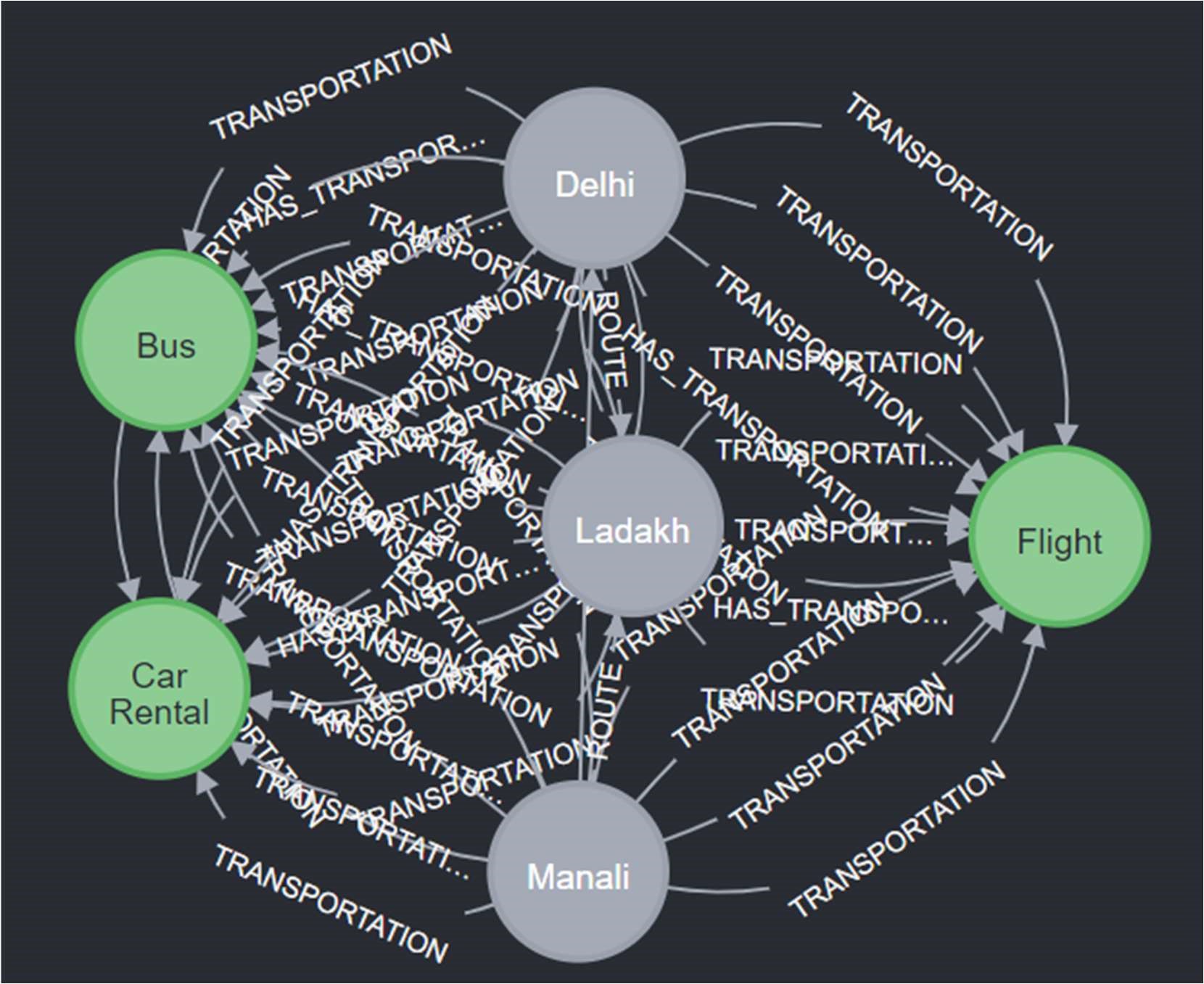
3.2.1.5 has(RoadTrip, sufficient\_funds\_or\_credit\_cards).

3.2.1.6 necessary(RoadTrip, be\_prepared\_for\_emergencies).

* 1. Knowledge Graph using Neo4j

|  |
| --- |
| # Importing necessary libraries from neo4j import GraphDatabase    # Initializing the driver uri = "bolt://localhost:7687" username = "neo4j" password = "123456kav"  driver = GraphDatabase.driver(uri, auth=(username, password))      # Defining function to manage logistics def manage\_logistics(): # Start session with driver.session() as session: # Querying for route information  route\_query = '''  MATCH (delhi:City {name: 'Delhi'}), (ladakh:City {name:  'Ladakh'})  CALL algo.shortestPath.stream(delhi, ladakh, 'distance')  YIELD nodeId, cost RETURN algo.asNode(nodeId).name AS city, cost  '''  route\_results = session.run(route\_query)  # Printing route information print("Route Information:") for result in route\_results:  print(f"{result['city']} - {result['cost']} km") print("\n")    # Querying for budget information  budget\_query = '''  MATCH (delhi:City {name: 'Delhi'})-[:HAS\_BUDGET]-  >(budget:Budget)-[:HAS\_BUDGET]->(ladakh:City {name: 'Ladakh'})  RETURN budget.amount AS budget  '''  budget\_result = session.run(budget\_query).single() budget = budget\_result['budget'] # Printing budget information print("Budget Information:") print(f"Total budget: {budget} INR") print("\n") |
| # Querying for emergency information  emergency\_query = '''  MATCH (delhi:City {name: 'Delhi'})-[:HAS\_EMERGENCY]-  >(emergency:Emergency)<-[:HAS\_EMERGENCY]-(ladakh:City {name:  'Ladakh'})  RETURN emergency.name AS emergency\_name, emergency.contact AS emergency\_contact '''  emergency\_result = session.run(emergency\_query).single() emergency\_name = emergency\_result['emergency\_name'] emergency\_contact = emergency\_result['emergency\_contact']  # Printing emergency information print("Emergency Information:")  print(f"Emergency contact: {emergency\_name} -  {emergency\_contact}") print("\n")    # Querying for backup plan information  backup\_query = '''  MATCH (delhi:City {name: 'Delhi'})-[:HAS\_BACKUP]-  >(backup:Backup)<-[:HAS\_BACKUP]-(ladakh:City {name: 'Ladakh'}) RETURN backup.name AS backup\_name, backup.details AS backup\_details '''  backup\_results = session.run(backup\_query) backup\_plans = [] for result in backup\_results: backup\_plans.append(f"{result['backup\_name']}:  {result['backup\_details']}")  # Printing backup plan information print("Backup Plan Information:") for backup in backup\_plans:  print(backup) print("\n")    # Calling the manage\_logistics function manage\_logistics()    # Closing the driver driver.close() |





* 1. New Inferences
     + - Proper management of logis cs is crucial for a successful and stress-free road trip.
       - It is important to plan and prepare for unexpected issues that may arise during the trip.
       - In addi on to having enough funds or credit cards, it is also wise to have some emergency cash or a backup payment method.
       - Good communica on and problem-solving skills are valuable in managing logis cs during a road trip.