An Improvement for Spatial-Temporal Queries of ATMGRAPH

ZHANG Zhiyuan[1], HAN Boyang[2]
School of Computer Science and Technology, Civil Aviation University of China, Tianjin, China, 300300

***Abstract*—As a knowledge graph for the field of ATM (Air Traffic Management), ATMGRAPH integrates aviation information from various sources, and provides a new way to comprehensively analyze ATM data, but the storage schema of ATMGRAPH is inefficient for trajectory-related queries which have typical spatial-temporal characteristics, thus cannot meet the application requirements. This paper presents an improved storage model of ATMGRAPH, specifically, we design a cluster structure to connect trajectory points and spatial-temporal information to speed up trajectory-related queries, and we link flights, airports, and weather information in an effective way to speed up weather-related queries. We create a dataset of about 10,000 real domestic flights, and build a knowledge graph of it which contains about 11.66 million triplets. Experimental results show that ATM knowledge graph constructed by this storage model can significantly improve the efficiency of spatial-temporal related queries.**

***Keywords*—Air traffic management; knowledge graph; storage model; spatial-temporal query; ontology**

I. INTRODUCTION

With the rapid development of the economy, people are willing to travel by air due to its efficiency and convenience. The civil aviation industry generates a large amount of data every day, coming from multiple departments such as airports, airlines, Air Traffic Managements (ATMs) and meteorological bureaus, with varying data forms and coding rules, make it a great challenge for semantic data query and analysis. As Aviation data are scattered in different systems, integrating them into a big semantic database seems to be a good idea. The most representative work is ATMGRAPH (Air Traffic Management Knowledge Graph) constructed by NASA. This KG (Knowledge Graph) integrates multiple aviation information and is benefit for semantic data analysis. Flight trajectory information accounts for the vast majority in ATMGRAPH, it has obvious spatial-temporal characteristics, and data analysis on trajectory is often about spatial and temporal. However, in practical applications, ATMGRAPH encounters great scale problems, especially when facing spatial-temporal related data queries, i.e. its performance decreases dramatically for huge data volumes.

There are few works to address this problem, in order to fill this research gap, this paper conducts on spatial-temporal query optimization of ATMGRAPH. A knowledge graph can be logically divided into two layers: the data layer and the schema layer. The data layer stores knowledge facts, and the schema layer defines ontology to standardize a series of fact expressions in the data layer [7].This paper designs an improved storage model for ATMGRAPH to solve the problem of slow and inefficient processing of spatial-temporal related queries. Specifically, we design a cluster structure to connect trajectory points and spatial-temporal information to speed up trajectory-related queries, and we link flights, airports, and weather information in an effective way to speed up weather-related queries. Experimental results on real aviation data show that the query efficiency using our model is significantly improved in typical application scenarios.

The rest of this paper is organized as follows. Section II is the related work. Section III is the problem definition, which introduces NASA's original ATMGRAPH model and analyzes its shortcomings in spatial-temporal related queries. In Section IV, we introduce our improved ATMGRAPH model in detail. Section V is the experimental results and discussion, and we conclude our work in the final section.

II. RELATED WORK

With the rapid development of the global transportation industry, air traffic flow has significantly increased. There were lots of research works on air traffic management such as airspace saturation, flight accidents, flight delays, and air control difficulties. The Federal Aviation Administration (FAA) used big data analysis to identify operational patterns, which can support the identification and prediction of airport data [2]. Rezo [3] introduced a paradox in aviation data processing and proposed a probable solution. Dorota [4] discussed the requirements of aviation data in Polish regulations and gave a practical proposal. Keller et al. [5] introduced a system for combining heterogeneous air traffic management with semantic integration techniques, which transforms data from disparate source formats into a unified semantic representation of ontology-based triplets. Liu et al. [6] implemented seamless communication and mutual cooperation between civil aviation systems through information sharing, which could support collaborative decision-making of air traffic management and improve the capacity of airspace systems. Lu et al. [7] proposed an integration architecture of cloud computing and blockchain for ATM systems, in which it pointed out the advantages of the new technology architecture over the traditional architecture of existing ATM systems. Europe and the United States are trying to use ontology technology to integrate and fuse aviation data from multiple sources, so as to provide a unified data exchange mechanism with semantic information for all participants in the aviation industry. For example, the Single European Sky Program launched the BEST project (http://www.project-best.eu), which designed AIRM (ATM Information Reference Model) and constructed an ontology

model for aeronautical and meteorological information [8]. At the same time, NASA constructed ATMONTO (ATM Ontology), involving ATM core data such as aircraft, flight, airport, airline, route, and navigation facility [9]. It includes over 150 classes, over 150 datatype properties, and over 100 object properties. Based on ATMONTO, NASA also built ATMGRAPH, a knowledge graph containing 260 million triplets [10]. Many information of ATM has temporal and spatial characteristics, e.g. when an airport is temporarily closed due to snow conditions, the airport operation status in KG should be changed to CLOSED, and the start and end time should also be indicated. Therefore, Schuetz et al. [11] proposed the concept of Contextualized Knowledge Graphs by adding semantic dimensions such as time, space, and data source in KG to solve the problem of information distribution and acquisition for all participants in the aviation industry.

III. Problem Definition

As the latest achievement of symbolism, knowledge graph is an important milestone of artificial intelligence. Knowledge graph can provide valuable structured information by data integration and standardization, and it has been widely used in information retrieval, automatic question answering, decision making and other fields, and it is also an important basic technology to promote data mining and intelligent information services [12]. With the growing scale of the knowledge graph, data management issues become increasingly prominent [13]. KG is generally divided into general knowledge graph and domain knowledge graph, and the latter usually needs to carefully design the storage model according to the industry data's characteristics in order to meet the retrieval requirements under large-scale data.

Consider the following two representative queries in ATM：

- Find all flights passing through the ZBAAAR20 sector of Beijing on July 20, 2022 and landing at Beijing Capital International Airport under strong wind conditions.
- Find which sector controlled the most flights between 9am and 10am on July 16, 2022.

ATMGRAPH consists of one month's flights (approximately 100,000 flights) and weather data in the New York metropolitan area， which includes eight classes: airspace structure and facilities, flight routes and procedures about takeoff and landing, traffic management measures, flight carriers and aircrafts related, airport and ground operations, weather, sequence related, and spatial-temporal related. Fig. 1 is a segment of ATMGRAPH, with a specific flight instance at its center: UAL535, which took off at 00:19:00 on July 15, 2014. Connected to it includes the departure and arrival airport of the flight, the carrier airline, the aircraft model, the planned route and the actual route. The lower part of the figure represents the track points of the flight, each contains information such as time, longitude, latitude, altitude, speed etc. (not listed in the figure for brevity). Although there are classes about weather and sectors in ATMGRAPH, getting the results of the two representative queries above is very inefficient, cause it must check all points one by one whether it matches the corresponding constraint. For example, when querying the workload of a sector during a certain period of time (i.e. the number of flights flying within the sector during this time), at first we must find all track points within that period of time, then for all of them we need to check whether their positions are within that sector, and finally output the corresponding flight information. Obviously, these kinds of operations are quite inefficient.

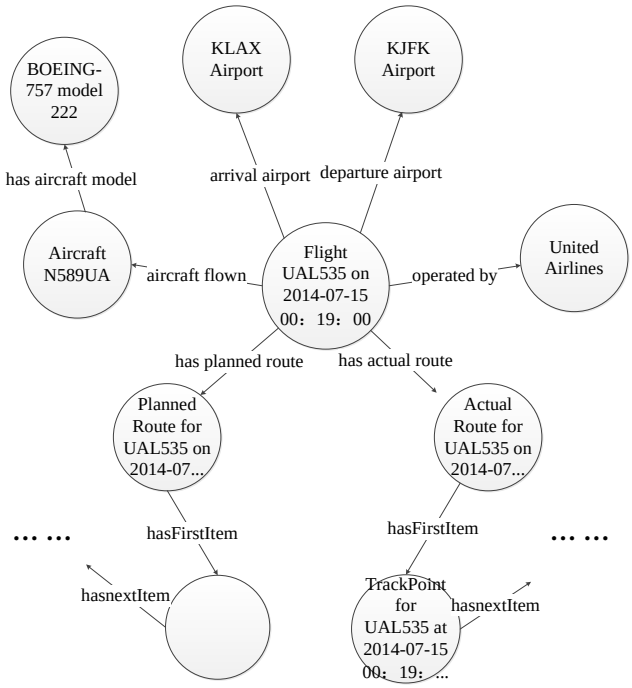

Fig. 1. Flight information storage segment.

This paper designs an improved storage model, which not only considers the strongly correlated characteristics between flight and weather information, flight and spatial-temporal information, but also links the trajectory points with spatial-temporal entities, so as to speed up spatial-temporal related queries. Experimental results on real flight data show that our proposed model greatly improves the query speed for representative queries and for some queries which the original model may take hours, our model can finish them in just a few seconds.

IV. Improved Storage Model

Although there are already eight major classes in ATMGRAPH to represent various knowledge in ATM field, some of them are relatively independent, making it difficult to obtain results using a single query statement involving multiple classes. The nodes of flight trajectories in ATMGRAPH account for nearly 70% of the entire graph, and each track point is only connected in chronological order using the hasNextTrackPoint relationship. This kind of storage model not only occupies a large amount of storage space but also reduces query efficiency. On the premise of being consistent with the original structure of the ATMGRAPH, this paper extends it to express more spatial-temporal information without taking up more storage space.

Fig. 2 illustrates our improved storage model, where ellipses represent the newly added classes and dashed edges represent the newly added relationships. TimeInterval is a new class for standard time segment, which connects the Trackpoint class through belongToTimeInterval relationship to express track points with the standard time segment information. The relationship belongToSector connects the Trackpoint class with the Sector class representing which sector the track point is located in. The new class WeatherInterval represents weather conditions of each airport in different time periods, and it also connects to the Flight class through two new edges: hasArriveWeather and hasDepartureWeather. The class ActualRoute represents the actual flight route, which contains the first and last track points of the trajectory through the edge of hasFirstTrackpoint and hasLastTrackpoint.

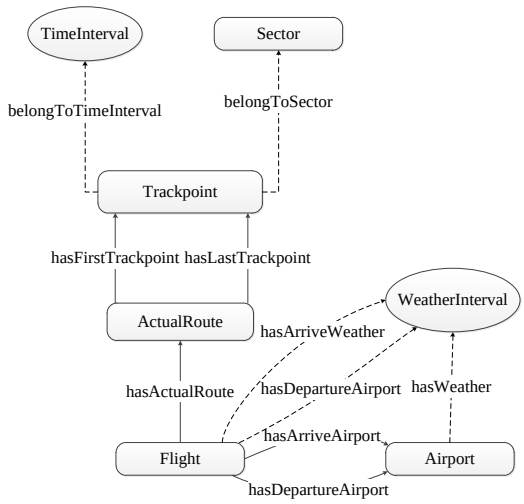

Fig. 2. Improved storage model of ATMGRAPH.

Through this storage model, track points has a direct connection to time and sector information, thus alleviates the problem of inefficient spatial-temporal related queries in the original ATMGRAPH. At the same time, weather information also has a direct connection to flights, which can solve the problem of slow query speed for weather and flight related queries.

A. Standard Time Interval

In the real world, many facts have time attributes, which play an important role in knowledge graph [14]. For example, the fact represented by a triplet (Steve Jobs, diedIn, California) is that Steve Jobs died in California, which occurred on October 5, 2011; The fact (Ronaldo, playing for A.C. Milan) was only valid between 2007 and 2008. In air transportation, when an aircraft performs a complete flight mission, time information cannot be ignored. In our experiment, the flight data comes from ADS-B, which broadcasts real-time information including aircraft position, speed, identification code, flight number, and air-ground status to ATC (Air Traffic Control system) or other aircrafts through the air-to-air and air-to-ground data links [15]. Table I shows an ADS-B data fragment that contains three track points of flight EPA6206 during its mission on July 27, 2020. The specific information includes the flight number, aircraft number, and the current position (longitude, latitude, altitude), speed, heading, and data transmission time expressed in UTC (Universal Time Coordinated).

In ATM data analysis, usually we do not care much about the instantaneous state of an aircraft at a specific time point, and the time unit in queries is mostly hours or days. For example, finding the number of flights flying at altitudes above 6000m from 8:00 to 10:00 on July 10, 2022. To quickly retrieve the flight status of many flights within a same time segment, creating standard time intervals seems to be a feasible and effective method. This paper takes ten minutes as a standard time interval. If it is too long, it will lead to too many track points within a time interval, which will affect the query performance. If it is too short, it will cause too many TimeInterval nodes in the graph, and waste the storage space. Track points belonging to a same time interval are all linked to the TimeInterval entity representing that time segment, forming a cluster structure. Fig. 3 shows some track points of flight KNA8202 and flight CSZ9106 during the time interval from 7:00 to 7:10 on July 18, 2022. It can be seen that this cluster structure can gather all track points within the standard time interval without damaging the original relationships in the graph. Due to the fact that each track point is connected to its corresponding standard time interval node, related track points can be directly retrieved, without checking all track points one by one to judge whether they meet time constraints. This provides a more efficient way for time related query tasks.

The process of adding standard time intervals is as follows: Create all standard time intervals in the graph, and then calculate the corresponding TimeInterval for each track point according to UTC Time, and connect it with the relationship belongToTimeInterval. After that, the above query can be solved through a single Cypher query statement:

match(n:Trackpoint)-[r:belongToTimeInterval]-(m:TimeInterval)

where n.height >= 6000

and m.startTime >= 2022/07/10 08:00:00

and m.endTime <= 2022/07/10 10:00:00

TABLE I. ADS-B DATA SEGMENT

Fnum	UTC Time	Latitude	Anum	Angle	Speed	Height	Longitude
EPA6206	2022/7/27 11:59:16	30.5709	B204N	21	361.14	1013.46	103.94346
EPA6206	2022/7/27 11:59:31	30.61871	B204N	22	355.584	1226.82	103.96566
EPA6206	2022/7/27 11:59:46	30.63263	B204N	22	357.436	1325.88	103.97208

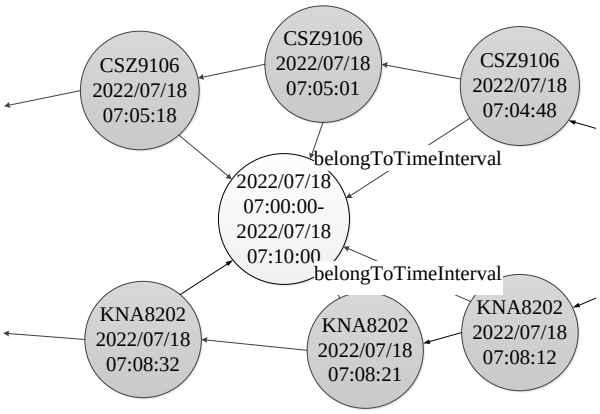

Fig. 3. Temporal cluster structure based on TimeInterval.

B. Spatial Sector Clusters

Spatial is an essential attribute of geographic data, which is mainly used to describe the spatial features of geographical entities, including position, shape, and spatial relationships [16]. Sector is a major geospatial entity in ATMGRAPH, usually formed as a polygon with height range, and the polygon is consisted of multiple points with longitude and latitude coordinates connected from head to tail. Sector is a fundamental unit of air traffic management services and is an important component for airspace planning and allocation [17]. In the field of ATM, many typical queries focus on the workload of a sector over a period of time (i.e. the number of flights in that sector within a specific time period). For example, finding the workload of sector ZBAAAR18 from 8:00 to 10:00 on July 10, 2022.

Traditional way to answer this kind of question in ATMGRAPH is to check trajectory points one by one if it is located in the given sector, which is very time consuming. To address this, this paper takes the sector as a central node and connects all track points within the sector to it, thus forming a cluster structure. When executing above queries, we only need to search the corresponding sector first, and find all track points connected to it. After filtering out duplicate flight numbers, the query results can be obtained immediately. Due to the fact that adjacent trajectory points may belong to two different sectors, how to determine whether a track point is within a sector? The ray crossing number method is generally used to determine whether a point falls inside a polygon. Specifically, firstly we draw a ray emitted from that point, and then we calculate the number of intersections between the ray and the polygon boundary: if the number of intersections is odd, then the point is inside the polygon, otherwise it is outside the polygon. In Fig. 4, a ray passes through an irregular polygon, and if the starting point of the ray is located in the thin line section, it has an even number of intersections with the polygon, and if it is located in the thick line section, it has an odd number. According to the above method, the points in the thick line section are inside the polygon, and the points in the thin line section are outside.

Fig. 5 shows a cluster structure fragment centered on sector ZBAAAR18 in the Beijing flight control area, with surrounding nodes of trajectory points. The names displayed in the nodes are the flight numbers and instantaneous times. Similar to the temporal cluster, this structure also does not disrupt the original track point connection relationship in the graph. When conducting a query about workload of an air traffic control sector, it is possible to directly find the Sector node and use the belongToSector relationship to reversely find its connected track points, and it is not necessary to calculate the position of each track point any more, thus greatly reducing the query time.

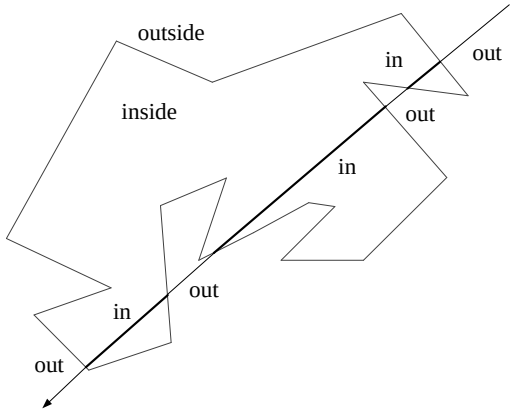

Fig. 4. Ray crossing number method to determine whether a point is inside the polygon.

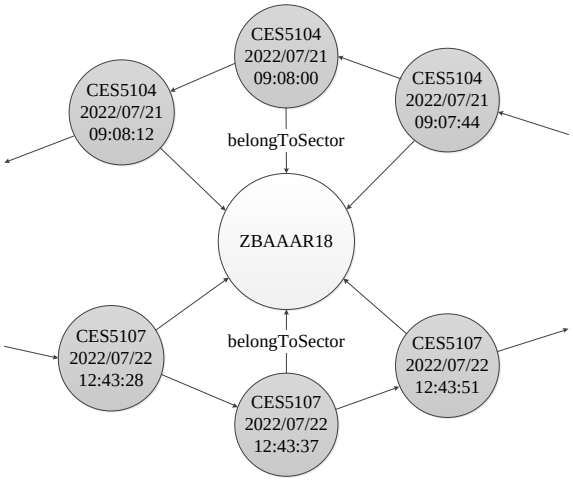

Fig. 5. Spatial cluster structure based on sector.

The process of adding spatial sector nodes is as follows: First, import the information of all sectors into the knowledge graph, then use ray crossing number method to calculate each track point to judge its relationship to the sectors, and finally connect each track point with its sector through the belongToSector relationship. With the spatial cluster structure, it is very easy to answer sector related queries. For example, when solving the query mentioned in this section, we can obtain the results in Neo4j by a single Cypher query statement:

match(n:Trackpoint) -[r1:belongToTimeInterval]-(m:TimeInterval)

where m.startTime >= 2022/07/10 08:00:00

and m.endTime <= 2022/07/10 10:00:00

with n match(n)-[r2:belongToSector]-(o:sector{ name:'ZBAAAR18'})

return count(distinct(n.fnum))

After adding temporal and spatial clusters to ATMGRAPH, all track points are connected with their corresponding temporal and spatial information. Fig. 6 shows a segment about the connection relationship between TimeInterval, sector and Trackpoint of our improved knowledge graph. At this point, when considering a query like 'Which sector controlled the most flights between 9:00 am and 10:00 am on July 16, 2022', we can simply find the six TimeIntervals that represent this period of time, filter out all the track points within them, and then count the number of flights included in each sector to obtain the final results.

C. Airport Nodes with WeatherIntervals

Weather conditions are very important for aircraft takeoff and landing. For the first representative query mentioned in Section II about finding all flights that pass through the ZBAAAR20 sector and land at Beijing Capital International Airport (BCIA) under strong wind conditions on July 20, 2022, the usual processing method requires two query operations (querying the time span of strong wind conditions at BCIA that day, and querying all flights that land at BCIA that day) and one comparison operation (check those flights one by one if its landing time is in the time period of strong wind). If the weather information when an aircraft arrives at or departures from an airport is directly stored in the knowledge graph, then the speed of answering such questions will be significantly improved.

This paper adds weather information to each airport based on the class WeatherInterval, and each flight is also directly connected to its weather information during departure and arrival. Considering that weather generally does not change frequently in a short period of time, unlike TimeInterval, the span of the WeatherInterval is set to 12 hours. As shown in Fig. 7, flight CHH7810 landed at Beijing Capital International Airport on July 18, 2022, and the weather when landing was rainy. Airports may have different weather conditions at different time periods and are connected to WeatherInterval by the relationship of hasWeather. Flights are also connected to WeatherInterval by relationships of hasArriveWeather and hasDepartureWeather. Using the new schema, when processing queries related to weather conditions, there is no need to match the landing time and corresponding weather information of flights one by one anymore, and it can be obtained directly through WeatherInterval. For the typical query mentioned in this section, we can obtain the results in Neo4j by a single Cypher query statement:

match(n:flight)-[r:arriveAirport]-(m:Airport{code{'PEK'})

where n.endtime >= 2022/07/20 00:00:00

and n.endtime < 2022/07/21 00:00:00

with n match(n)-

[:hasArriveWeather]- (:weatherInterval{weather：'strong wind'})

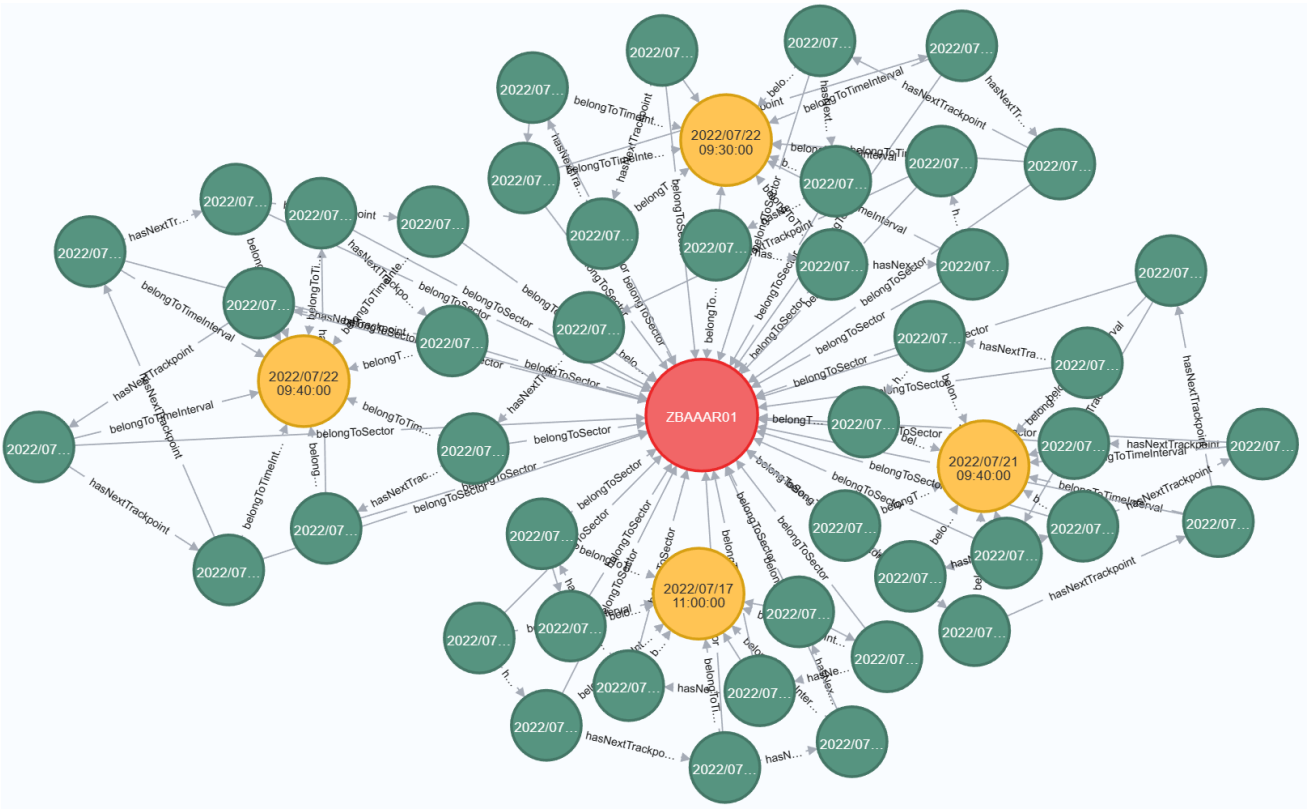

Fig. 6. A fragment of our improved ATMGRAPH.

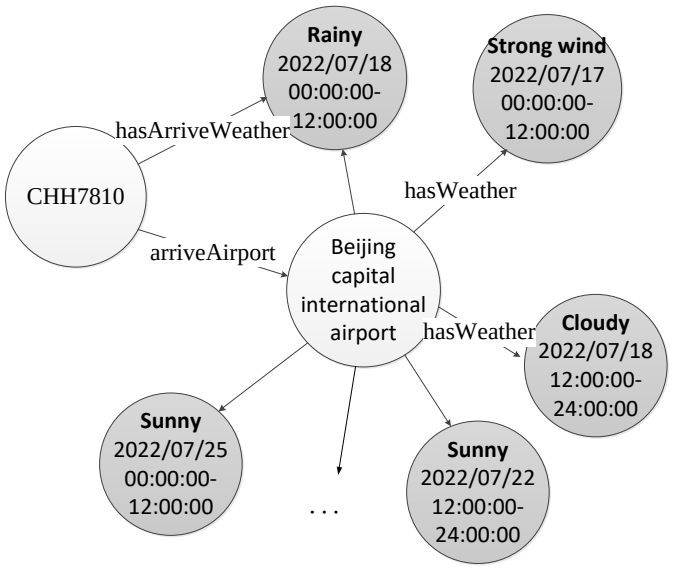

Fig. 7. A fragment of flights, airports and WeatherIntervals.

V. Experimental Results and Discussion

The experimental environment is a 64bit Windows system (Intel i7-7700HQ CPU, 16GB memory), 4.2.2 community version Neo4j, implemented using Python language.

We crawled ADS-B data from varflight website (https://www.variflight.com) about 10,000 flights from July 16 to 27, 2020. The data of airports, flight information regions, and sectors are from AIP (Aeronautical Information Publication), and weather data is randomly set. Using these data, we built ATMGRAPH of two versions: NASA's original version and our improved version, with the latter containing 5.5 million nodes and 11.66 million relationship edges. We then evaluate their performance using three typical query cases, and the results are shown in Table II. The evaluation metric is query time. The first query case is only temporal related, the second query case is spatial-temporal related, and the third query case is about time, airport, and weather condition.

Table II shows the comparison results between NASA's original ATMGRAPH and our improved version on commonly used spatial-temporal related queries. The first is a common time related query in ATM data analysis. Using our storage model, due to the existence of standard time segment clusters, it is very easy to find all track points belonging to the TimeInterval from 9:00 to 12:00 on July 27, 2020. On the contrary, for the original ATMGRAPH we must compare the UTC value in each track point. From the results in Table II, we can see that our model is about eight times faster than ATMGRAPH. The second query adds spatial constraint to the first one. For ATMGRAPH, because there is no direct connection between track points and flight information regions, to get the query results, we must calculate all track points in the graph and judge the topological relationship between each track point and each flight sector. Because the number of track points is very huge and grows lineally with flight numbers, plus the position calculation is also very complex, thus it takes hours to obtain the query result. After adding a spatial cluster structure in our improved model, track points in the specified region can be directly found through the relationship belongToSector, and then the corresponding number of flights can be quickly obtained. The third query is related to the weather conditions at landing time. For this query, our model can directly get the flights that meet the conditions through a simple Cypher statement, while the original ATMGRAPH can be very complex: it should first identify the flights that land at the airport, and then find weather information of the airport during the landing time of the flights. Due to the fact that weather and flights in the original ATMGRAPH are not connected, the analyzer must manually check these flights one by one which is very time consuming, or develop a program to handle it which is very inconvenient. The result of ATMGRAPH for the 3rd query in Table II is gotten in a program way, which is about two seconds, while our model only uses five milliseconds, more than 400 times faster.

The above experimental results and discussion indicate that adding the spatial-temporal cluster structure proposed in this paper to ATMGRAPH can quickly process queries related to spatial-temporal features and improve data analysis speed.

TABLE II. Performance Comparison of Typical Queries

Query cases	ATMGRAPH Version	Query Time
Find the number of flights from 9:00 to 12:00 on July 27, 2022	ORIGINAL	3748ms
	IMPROVED	421ms
Find the number of flights passing over Beijing on July 25, 2022	ORIGINAL	2.5h
	IMPROVED	1346ms
Find the number of flights landed at Beijing Capital International Airport from July 16 to 27, 2022	ORIGINAL	2160ms
	IMPROVED	5ms

VI. CONCLUSION

In order to solve the problem of low efficiency of spatial-temporal related queries in ATMGRAPH, this paper proposes an improved storage model, which uses spatial-temporal clusters to represent flight information regarding time and location. In our improved model, trajectory points are connected to standard time intervals and sectors, and flight and airport entities are connected to weather intervals. Experimental results show that after adding the spatial-temporal cluster structure to the knowledge graph, the speed of relevant queries is greatly improved.

Due to the fact that the track data in ATMGRAPH accounts for approximately 70% of the total data volume, this article only focuses on improving the mode layer and cannot solve the redundancy problem of a large amount of track data. Aircraft trajectory points are stored as an unidirectional chain structure in Neo4j, and we can study a new storage structure for this typical kind of data in the future to save storage space and to optimize data query speed.

REFERENCES

[1] Z. Xu, Y Sheng, L He, and Y Wang, Review on knowledge graph techniques[J]. Journal of University of Electronic Science and Technology of China, 2016, 45(04): 589-606. (in Chinese)

[2] FAA.Terminal area forecast (TAF) [EB/OL]. [2017-05-18]. https://taf.faa.gov

[3] R. Zvonimir, M. Tomislav, S. Sanja, and T. Andrea, A Paradox in aeronautical data processing: A case study review[J]. Case Studies on Transport Policy, 2022, 10(2).

[4] D. Marjańska, Aeronautical data requirements and geodetic data – a case study on regulations in Poland[J]. Aircraft Engineering and Aerospace Technology, 2022, 94(5).

[5] R. M. Keller, S. Ranjan, M. Wei, and M. M. Eshow, "Semantic representation and scale-up of integrated air traffic management data," SBD '16, 2016.

[6] L. Liu, H. Yang, and Y. Huang, Sharing big data storage for air traffic management[C]. 2022 IEEE 8th International Conference on Computer and Communications (ICCC), Chengdu, China, 2022, 1199-1203, DOI: 10.1109/ICCC56324.2022.10065656.

[7] X. Lu, and Z. Wu, "ATMCC: Design of the Integration Architecture of Cloud Computing and Blockchain for Air Traffic Management." 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) (2021): 37-43.

[8] I. Kovacic, D. Steiner, C. Schuetz, B. Neummayr, and S. Wilson, Ontology-based data description and discovery in a SWIM environment[J], ICNS 2017, 1-22, doi: 10.1109/ICNSURV. 2017.8012006.

[9] R. M. Keller, The NASA air traffic management ontology: Technical Documentation Technical Memo NASA/TM-2017-219526, National Aeronautics and Space Administration, https://ntrs. nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170006095.pdf, 2017.

[10] R. M. Keller, Building a knowledge graph for the air traffic management community[C]// Companion The 2019 World Wide Web Conference. 2019.

[11] E. Gringinger, B. Neumayr, S. Christoph, M. Schrefl, and S. Wilson, The case for contextualized knowledge graphs in air traffic management[C]//CKGSemStats@ ISWC. 2018.

[12] T. Hang, J. Feng, and J. Lu, Knowledge graph construction techniques: Taxonomy, survey and future directions[J]. Computer Science, 2021, 48(02): 175-189. (in Chinese)

[13] X. Wang, L. Zou, C. Wang, P. Peng, and Z. Feng, Research on knowledge graph data mnagement:A Survey[J].Journal of Software, 2019, 30(07): 2139-2174. DOI: 10.13328/j.cnki.jos. 005841. (in Chinese)

[14] T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, and S. Li, Towards time-aware knowledge graph completion[C]//Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016: 1715-1724.

[15] K. Li. Research on the application status and trends of ADS-B abroad[J]. Aerodynamic Missiles Journal, 2018, 1(12): 60-66. (in Chinese)

[16] J. Liu, H. Liu, X. Chen, X. Guo, and X.Zhu, Construction of knowledge graph based on Geo-Spatial data[J]. Journal of Chinese Information Processing, 2020, 34(11): 29-36. (in Chinese)

[17] C. Xu, Y. Tian, K. Niu, W. Gong, and G. Li, Review of optimization for airspace sectorization[J]. Aeronautical Computing Technique, 2022, 52(01): 126-130. (in Chinese)