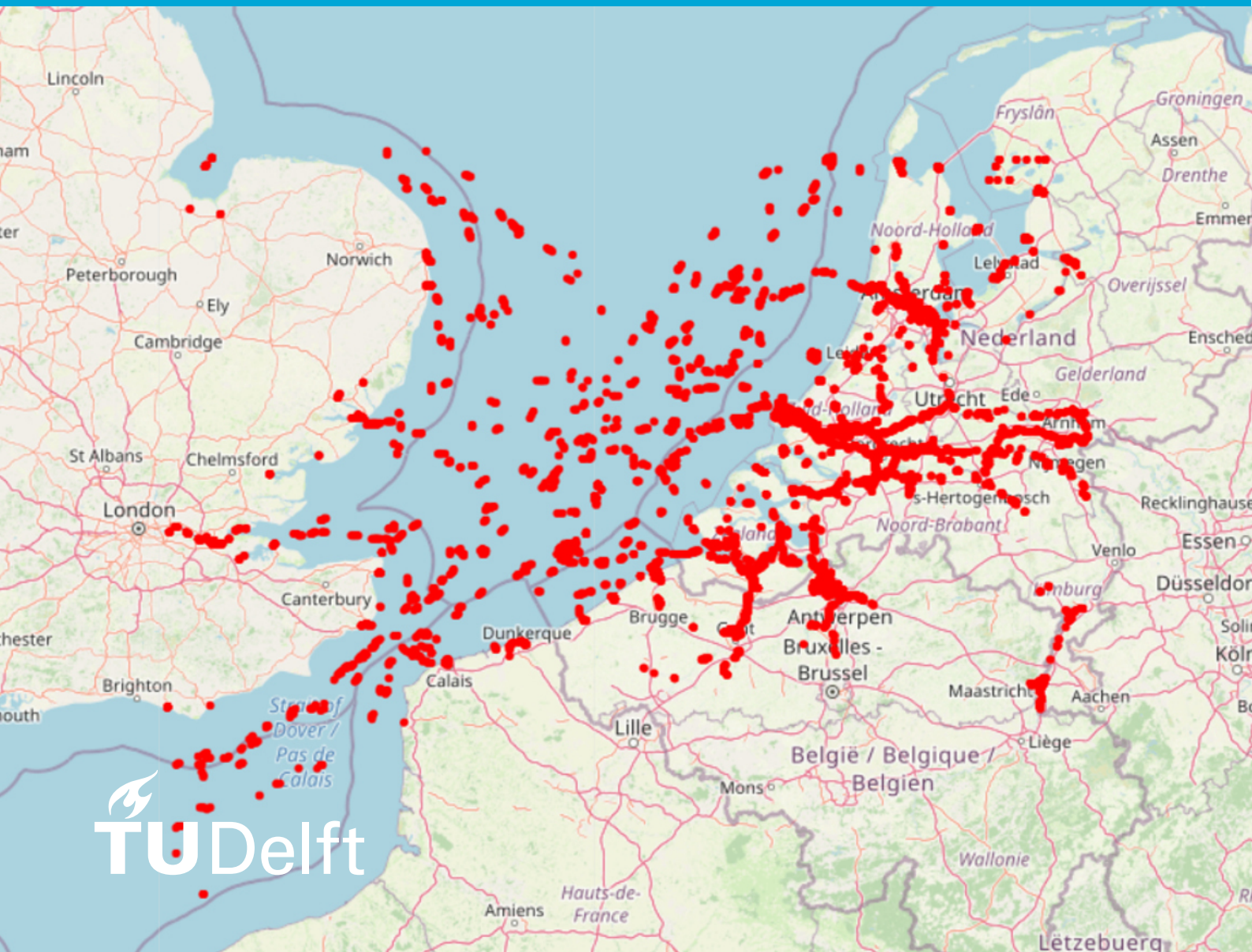
MSc thesis in Geomatics for the Built Environment

Manage 4D Historical AIS Data by Space Filling Curve

Jinglan Li

2020



TU Delft

MSc thesis in Geomatics

Manage 4D Historical AIS Data by Space Filling Curve

Jinglan Li

June 2020

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

Geo-Database Management Centre
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Dr.ir. Martijn Meijers
Dr. Haicheng Liu
Co-reader: Dr. Ken Arroyo Ohori

Abstract

This MSc thesis aims to research how to efficiently manage 4D AIS data (Longitude, Latitude, Time, and MMSI (the ID of the vessel)) to do the fast query by using the Space Filling curve in PostgreSQL. The AIS is the Automatic Identification System which is born because the frequent occurrence of maritime accidents has caused casualties and economic losses. AIS is intended to assist a vessel's watchstanding officers and allow maritime authorities to track and monitor the vessels' movement.

For now, the AIS has been used in various kinds of fields because the AIS data is really important and useful. The AIS data contains lots of useful information such as the dynamic information (including ship location, speed, heading, and so on), the static information (including ship name, ship type...), and some other types of data. Because of the useful information that AIS includes, there a great many useful applications based on the AIS data. For example, the AIS data is used for detecting the vessels' anomalies motions or tracking the vessels. While the studies mainly focus on the applications of the AIS data, the efficient management of the AIS data is neglected. Hence, I am going to study how to efficiently manage the multidimensional AIS data.

Space Filling Curve (SFC) will be used to manage the multidimensional AIS data. The SFC is a great method for indexing the multidimensional data. The SFC can map data in multidimensional space to 1D space. There are lots of kinds of SFC, such as Morton curve, Hilbert curve, Gray curve, and so on. And the Morton curve and Hilbert curve are used in this thesis because of the property of the locality form the nD space is preserved in the location on the curve [Dai and Su, 2003] and both are so-called quadrant recursive curves [Meijers and van Oosterom, 2018] which is the very significant property.

In my research, I proposed two kinds of methods to manage 4D AIS data. One is the 4D integrated approach that the 4D AIS data is encoded to SFC together. The other is the 3D integrated approach, only 3D AIS (Longitude, Latitude, and Time) data is encoded. To test the two approaches, bounding box query (to find the vessels in a given space and time range) and trajectory query (to find the position information of a specific vessel in a give time range) will be implemented in the database. To verify the usability and superiority of my approach, the benchmark is set.

The comparison between the two approaches I proposed will be done. And results prove that the SFC approach I used to manage the 4D AIS data is great after comparing it with the benchmark I put forward.

Acknowledgements

With the coming of summer vacation, my time of studying in TU Delft is coming to an end. 2020 is a turbulent year and also a challenging year. Due to the impact of Coronavirus-19, we can only work and study at home. Thanks to the help and care given by teachers and friends, I was able to happily complete my tasks during the outbreak.

First of all, I want to express my thanks to my main mentor, Martijn Meijers. From the beginning of my topic selection, he showed me the right path. During the completion of the thesis, he tirelessly answered all my questions and urged me to complete the periodic tasks on time. And he unreservedly passed on to me his technical know-how. His previous works help me to complete my thesis. Then, I want to give my thank to my second mentor, Haicheng Liu. Thanks him for explaining the things which I do not understand in detail and sharing his great ideas and comments for my research with me. And I would also thank my co-reader, Ken Arroyo Ohori. Thanks him for reading my thesis and giving me useful comments which help me a lot.

Finally, I want to show my thanks to my family and my friends. Thank you for encouraging and comforting me when my mood is low. Formally because of your presence, I am more motivated to move forward.

Contents

List of Figures

List of Tables

1 Introduction

With the growth of the global economy, the global maritime traffic is becoming increasingly busy, and the frequent occurrence of maritime accidents has caused casualties and economic losses. The ship Automatic Identification System (AIS) was born in the 1990s [Feng et al., 2019]. The AIS which is installed on vessels continuously sends ship static information, dynamic information, navigation-related information, etc., which greatly facilitates the information exchange between ships and between ships and shore management stations [Feng et al., 2019]. AIS aims to keep ships safe by avoiding the collision at sea. In December 2000, International Maritime Organization (IMO) 's navigation sub-committee officially issued a proposal on mandatory installation of AIS equipment on ships [Li, 2017]. AIS is really useful in real life. Generally speaking, AIS is an effective tool for accomplishing navigational safety goals, and by doing so, can provide critical pre-emptive maritime safety benefits, but also provides a data opportunity with which to understand and help mitigate the impacts of maritime traffic on the marine environment [Robards et al., 2016]. AIS is intended to assist a vessel's watchstanding officers and allow maritime authorities to track and monitor vessel movements. AIS integrates a standardized VHF[1] transceiver with a positioning system such as a Global Positioning System receiver, with other electronic navigation sensors, such as a gyrocompass or rate of turn indicator. And there are a lot of sources of the AIS data which are collected at several places around the world. Within Europe, a.o. EMSA, Kystverket, Hellenic Coastguard, Dirkzwager, and Marine Traffic collect the data [2].

The AIS data is the most important part that is worthy of in-depth studying. Autonomously broadcasted AIS messages contain dynamic information (including ship location, speed, heading, rate of turn, destination and estimated arrival time) and static information (including ship name, ship MMSI ID, ship type, ship size and current time), which can be transformed into useful information for intelligent maritime traffic manipulations [Mao et al., 2018]. The AIS data is encoded by the rule 'NMEA-0183, the useful information is acquired by decoding the different types of raw AIS data. In general, there are 27 kinds of message types in the payload of AIS data, the dynamic information is in the position report (type 1, 2, 3) which provides a common reporting structure for navigational information[3], which are the most popular information for decoding. And the static information is in the static report (type 5). Besides, The update frequency of AIS data is very fast. Overall, ship broadcasts AIS information every 2s to 6min while sailing[Liu, 2017]. And the update frequency of different types of data is also different. Considering the huge number of ships at sea and the extremely fast update frequency of AIS, we can imagine that the amount of AIS data should be quite large.

1.1 Problem statement

Because of the useful information that AIS includes, a great many useful applications based on the data from AIS. Especially the application about the analysis of the vessels. [Ristic et al., 2008] uses statistical analysis of AIS data for the detection of possible anomalies in vessels' motions. When the normal behavior of a vessel is assumed, a prediction of future vessel motions shall be made [Meijers et al.,

[1] VHF: very high frequency is the International Telecommunication Union(ITU) designation for the range of radiofrequency electromagnetic waves from 30 to 300 megahertz, wich corresponding wavelengths of ten meters to one meter.`https://en.wikipedia.org/wiki/Very_high_frequency`

[2] More information can be found in this website: `https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php?title=Special:Pdfprint&page=WP4_Deliverable_1`.

[3] More information about AIS data type can be found in: https://gpsd.gitlab.io/gpsd/AIVDM.html, which is written by Eric S. Raymond in Nov 2019.

2016]. As Rajabi et al. wrote, the port is an important hub station at sea, it is a connection between the sea and the land which provides facilities for the ships docking to load and discharge passengers and cargo. The better decisions could be made by the terminal operators in real-time with the help of data from AIS. And the smart port could be built which has a great impact on the global economy. Also, there are many algorithms about the trajectory prediction by using and analyzing the AIS data. As I mentioned before, the amount of the AIS data is huge, the historical AIS data which means the 'old' data would be discarded from the memory because they are relatively redundant compared to the real-time AIS data. While, If these "expired" data can be used properly, they can also play a considerable role in reality. Obviously, the value of these historical data has been discovered by quite a few people. Hexeberg et al. proposes a quite novel data-driven approach to predict the next position of vessels by recursively using historical AIS data in the neighborhood of a predicted position. Wang et al. through a large number of AIS data analysis of historical ships, the parameters such as ship traffic flow, navigable density, and passing capacity were calculated, and the navigable saturation model was built based on the field of stopped ship's visual distance. And the AIS data even can be used to forecast the development of the maritime industry [Lechtenberg et al., 2019]. From here we can also see the importance of proper use of AIS data. Although the AIS data can be used in various kinds of applications, there seems does not exist an approach to store or manage the AIS data. As Mao et al. wrote, there is no existing standard AIS benchmark database in the maritime research area, which makes it quite inconvenient for researchers and practitioners in the field since collecting a usable dataset will cost a lot of time and effort. AndDe Vreede also found this problem that so many researches are mainly focusing on the use of AIS data, the efficient management of the AIS data is neglected by most people. Fortunately, however, this area which is neglected when we deal with AIS data is included in the field of moving object [4]research.

Management of moving objects and be seen as structuring and storing of moving objects. When facing the moving objects, space and time data are focused on by people. Moving object data management is defined as the storage and structuring of data from vessels within a database to support data analyzes[De Vreede, 2016] in this thesis. Lots of research on moving objects found that the DBMS is a great way to deal with this data of moving objects (longitude, latitude, time). For now, there are some studies focus on manage the multidimensional AIS data in the database, while, the researches still focus on managing 3D (longitude, latitude, time) data from AIS data in the database. Based on the above mentioned and many unmentioned applications of AIS data, we can see that it is far from enough to do the Spatio-temporal query for AIS data. The storage and query of multidimensional AIS data need to be discussed in depth. For example, a variety of applications of AIS data for maritime safety is based on the trajectory analyze which needs 4D data which are the longitude, latitude, timestamp, and the Maritime Mobile Service Identify (MMSI) of the specific vessel. And there are lots of great methods to deal with multidimensional data such as R-tree and space filling curve. The space filling curve is a great way to deal with multidimensional data, it maps the data in multidimensional to 1D space and then the B-Tree indexing method can be used to support the efficient query. De Vreede used this approach to deal with the AIS data in MongoDB. This MSc thesis will give a conclusion that whether the 4D historical AIS data can be efficiently managed in PostgreSQL to support the efficient query. The method will introduce the way that I am going to deal with 4D AIS data. And the research questions of this thesis will be introduced in the following part.

1.2 Research questions

This MSc thesis aims to figure out whether 4D AIS data (longitude, latitude, time, MMSI) can be efficiently stored and indexed by Space filling curve in 1D space to support the fast query. The main research question is:

How to efficiently manage 4D data (Longitude, Latitude, Time, MMSI) of vessels to do the fast query by using Space Filling Curve in PostgreSQL?

To answer the main question, the following sub-questions should be focused on:

[4]Moving objects: objects whose position changes continuously

- How to better manage the 4D data to support the efficient query? Dealing with the integrated 4D data or dealing with the integrated 3D data (3D + 1D) ?
- How to scale data in each dimension properly to compute the SFC key?
- Which Space filling curve performs better in the 4D data querying? Morton curve or Hilbert curve?
- How will the BRIN index be applied to columns in database to support the queries? From which aspects can I compare the indexing method?

1.3 Thesis outline

The rest of the thesis is organized as follows:

- Chapter 2 will give a detailed introduction about the related work of the thesis. Some research on storing and managing moving objects and some commonly used indexing methods that I will use in this paper.
- Chapter 3 is going to start with an overview of the Space filling curve approach which describes the principle of using space filling curve to index multidimensional data and how to query the data which is encoded in SFC. And then the main methodology I will use to efficiently manage 4D AIS data to achieve the query which I am going to use will be introduced.
- Chapter 4 covers the description of the implementation of the SFC indexing approach and the two kinds of queries (bounding box query and trajectory query) will be applied to the different indexing methods.
- Chapter 5 concludes the methods that are used for managing 4D AIS data and the future work will be introduced finally.

2 Related work

The related work chapter mainly includes the studies which are relevant for the thesis. Section 2.1 is going to explain the management of data which is included in the research of management and storage of moving objects firstly. The two kinds of data of moving objects, one is the dynamic or updated data and the other is the static or historical data, will be introduced. Section 2.2 mainly talks about the existing way to store and manage the mass dataset in database using various kinds of structure in database, either the relational or non-relational database will be mentioned. Next, the two indexing methods which I am going to use in the thesis will be introduced in section2.4. And the chapter will end with some relevant research on storing and management AIS data in database these years.

2.1 Management of moving objects

The management of moving objects has been intensively studied in recent years. A wide and increasing range of database applications has to deal with spatial objects whose position changes over time[De Almeida and Güting, 2005]. An infrastructure is emerging that enables data management applications that rely on the tracking of the locations of moving objects such as vessels, vehicles, and so on [Jensen et al., 2004]. Indexes for moving objects need to satisfy the efficient query and frequent updates at the same time. Hence, studies about moving objects were proposed. Moving objects are the most common and important component in a diverse range of phenomena, such as urban transportation, ship logistics in the ocean[Feng et al., 2019].

In recent years, database applications are used to deal with these moving objects. The objective of moving objects databases is to extend database technology to support the representation and querying of moving objects and their trajectory [Frentzos, 2008]. There are two types of Spatio-temporal objects, one is discretely moving objects and the other is continuously moving objects. Regarding the first type, for example, land parcels, roads, pollution areas [Forlizzi et al., 2000], countries, rivers, and so on, which is quite easy to be managed. The continuously moving objects which change their position in extent continuously are much more difficult to deal with [Güting et al., 2000]. To efficiently and properly manage the moving objects, several major moving objects models have been proposed. The first generation model offers simple structure objects, which show in figure2.1a, like single points, continuous lines, and simple regions. To represent a variety of the complex region, the second generation model is proposed. it includes the complex points, complex lines, and complex, which show in figure2.1b, regions[Praing and Schneider, 2007]. There are two kinds of models, one works on the present and future positions of moving objects, and the other works on the past position of objects, asking historical queries[Frentzos, 2008].
And the model for dealing with the historical position is quite different from the model which is used for present and future positions. For managing historical position, Erwig et al. proposed a novel approach where moving points and moving regions are viewed as 3D (2D space + time) or higher-dimensional entities whose structure and behavior is captured by modeling them as the abstract data type. The data type they defined could be used integrated as the base (attribute) data types into various types of DBMS. And Erwig et al. also introduced about this methods. Reiss et al. gives an approach that the management of historical data from a data stream point of view and implies a solution to manage and query current and historical data at the same time by using bit map indices [De Vreede, 2016].

With regarding the current and future position of moving objects, many models have also been proposed. The objects Spatio-temporal (MOST) data model is proposed by Sistla et al.. They represent the

(a) simple type (b) complex type

Figure 2.1: Types of moving objects

position as a function of time. Moving objects are referred to as dynamic attributes that change over time even if its position is not updated [De Vreede, 2016]. And the moving objects are always represented as a series of observations that consist of an id, location, and time in the GIS data model [Hornsby and Egenhofer, 2002]. While these models need comprehensive and complicated computation to support and even for the same kind of data, different GIS data models need to be built when applied to different fields. In recent years, Feng came up with a new methodology to improve this current situation [Feng et al., 2019]. They used relative space to build the GIS data model. In relative space, relative dynamic relationships between moving objects are easy to build independently of whether they can be geocoded by coordinates. Importantly, the analysis of moving objects in relative space could easily follow structural requirements. Hence, the relative space-based GIS data model of moving objects changes the analysis of current absolute space-based GIS models and facilitates the efficient computation of real-time relative relationship dynamics [Feng et al., 2019].

With the increasing number of computer applications that rely on large Spatio-temporal data sets, it becomes essential to provide efficient query processing techniques for spatio-temporal databases. [Mokbel et al., 2004]. Benetis et al. propose an algorithm for answering the nearest neighbor query for moving objects in the plane. Mokbel et al. give an introduction about the scalable incremental hash-based algorithm which is used for evaluating a set of concurrent continuous spatio-temporal queries.

All in all, in maritime, the moving object data model can be applied for assessing collision risk [Bye and Aalberg, 2018], predicting vessel behavior[Zissis et al., 2016] and path planning[Cummings et al., 2010]. While the spatial-temporal data is mainly used in the moving object data model. More and more application in maritime need not only the spatial-temporal data but also the 4D data (with the MMSI) or even higher-dimensional data.

2.2 Mange and organize data in database

The indexing technology can improve the operation efficiency of the database, and it is the base to realize the efficient storage and query of massive data in the database. The main idea of the indexing is to avoid the whole table being scanned in order to reduce the query time. There are two types of databases that are most commonly used. One is the relational database which has been leading within the data management[De Vreede, 2016], the other is the relatively new NoSQL. In my research, the relational database will be used. Hence, the detailed approach of structuring and managing data in the relational database will be introduced.

2.2.1 Manage data in relational database

In traditional RDBMSs, the entries are stored in a table, and an additional spatial index is built separately (Fox et al., 2013). The spatial indices are used by spatial databases to optimize the spatial queries access to relational databases can be divided into two categories according to data types, point access method (PAM), and spatial access method (SAM) (Hao, 2010). PAMs are used to improve the access time in collections of spatial points. SAMs are more general and are used to improve the access time in collections of geographic objects (e.g. points, lines, polygons, etc.) [Volker and oliver, 1998]. With the

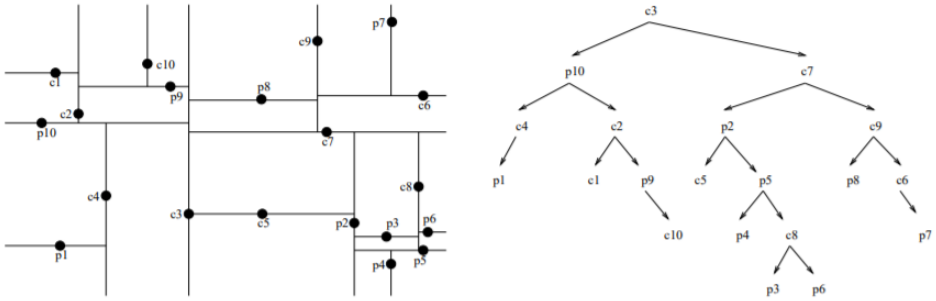


Figure 2.2: KD Tree

development of data quantity and storage, many different index schemes have emerged to help users access data more quickly and accurately.

Point access methods (PAM)

A point access method is a data structure that supports storage and retrieval of points in a multidimensional space [Shih and Wang, 2004]. Point access methods have primarily been designed to perform spatial searches on point databases (i.e., databases that store only points) [Volker and oliver, 1998].The function of the PAMs is to cluster the points which close in nD space into the same areas of the index structure[Shih and Wang, 2004] which could accelerate the efficiency of queries. There are various PAMs indexing technologies have been mentioned.

- K-D Tree The K-d-tree is a d-dimensional data structure which shows in figure2.2 and one of the most prominent PAMs [Brisaboa et al., 2009]. The K-D Tree is a binary search tree that represents a recursive subdivision of the universe(nD data structure) into subspace utilizing (D-1) dimensional hyperplanes[Volker and oliver, 1998]. It is a balanced binary search tree, the internal nodes of the tree represent axisymmetric hyperplanes pace. And the axis of symmetry is the median of the dimension with the smallest difference among the n dimensions. The axis of symmetry divides the space represented by its dimension into two halves which correspond to the two children of the inner node. Then, the recursive method is used to divide each space into several subspaces and perform relevant search operations in the subspaces[Xu, 2010].

  Although KD tree is suitable for discrete points as organizational objects, there are two main disadvantages of the KD Tree, One disadvantage of the k-d-tree is that the structure is sensitive to the order in which the points are inserted. Another one is that data points are scattered all over the tree[Volker and oliver, 1998].

Spatial access methods (SAM)

The main propose of SAM is to support efficient spatial selection, for example, range queries or nearest neighbor queries, of spatial objects. Besides, it is also used to implement efficient spatial analysis such as map overlay, and other types of spatial joins [van Oosterom, 1999]. The first spatial access method (SAM) was R-tree which was proposed in 1984 by Guttman[Xu, 2010], and then many varieties of R-tree were evolved according to different needs and application, such as R+-tree, R*tree, etc. Also, R-tree is the extension of B-tree to multidimensional point data. And the Space Filling Curve is another SAM, it maps high-dimensional data to one-dimensional data, and then uses one-dimensional index structures to store the data which improves the performance of the storage of multidimensional data.

- R-Tree
  One of the most popular spatial access methods and a paradigmatic example is the R-tree[Guttman, 1997]. The R-tree shows in figure2.3 is an index structure that was defined by Guttman in 1984[van

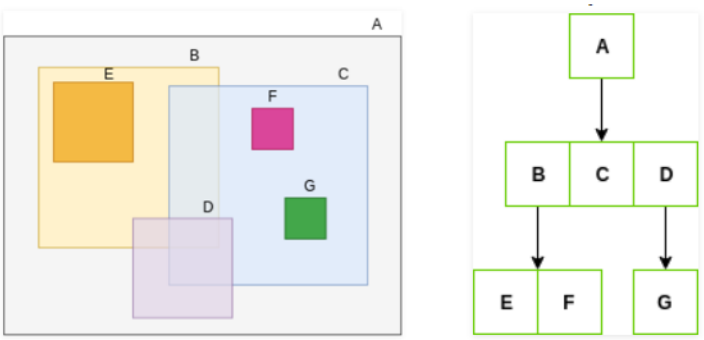


Figure 2.3: R Tree(`https://www.geeksforgeeks.org/introduction-to-r-tree/`)

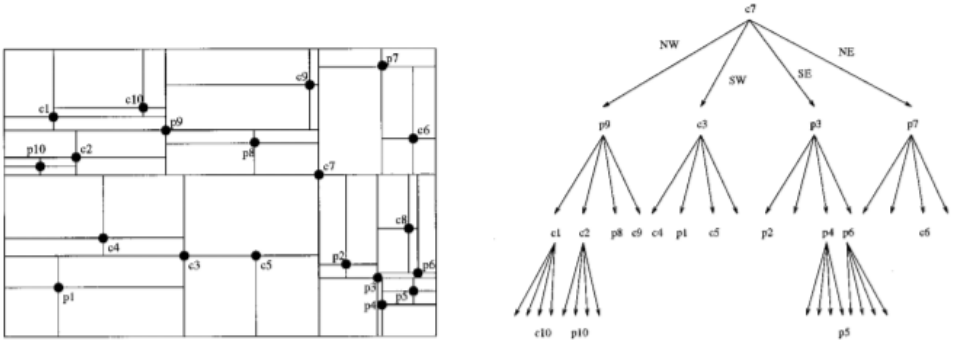


Figure 2.4: Point Quadtree [Volker and oliver, 1998]

Oosterom, 1999]. The tree is kept balanced by splitting overflowing nodes and merging underflowing nodes. There is an example that describes the R-Tree. Although the R-Tree can organize any dimensional data, when the data quantity of the index becomes large, the overlap between the depth of the tree and the index space will increase, and the search performance will decrease sharply [Kothuri et al., 2002].

Besides, there are many variants of R-Tree, such as R*-Tree, Hilbert R-Tree, and so on, which also play important roles in various fields.

- Quadtree
  The quadtree with its many variants is closely related to the k-d-tree. The Quadtree was definded by Finkel and Bentley in 1974 [Fox et al., 2013]. A quadtree is a tree where each non-leaf node has exactly four children. This structure allows one to split a rectangular region into quarters in a natural way [Fox et al., 2013]. Quadtrees are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The regions may be square or rectangular or may have arbitrary shapes. Quadtree spatial indexing was explored by lots of organizations and researchers [Kothuri et al., 2002]. There are lots of variants of Quadtree, such as point quadtree, region quadtree, PM quadtree. The first quadtree variant is the **point quadtree** which is a binary search tree [Volker and oliver, 1998], the figure shows a two-dimensional point quadtree (in figure2.4). It is very similar to kdtree but has a decisive difference. K-Dtree is the density balanced tree while Quadtree is the space partitioning tree which could cause the redundant of the space.

  The next variant I am going to introduce is the **region quadtree** (in 2.5), which is the most famous one. Region quadtrees are based on a regular decomposition of the universe; that is, the 2d subspaces resulting from a partition are always of equal size[Volker and oliver, 1998]. First, the area of interest is enclosed by a square. A square is repeatedly divided into four squares of equal size until it is completely inside (a black leaf) or outside (a white leaf) the polygon or until the maximum depth of the tree is reached [van Oosterom, 1999].

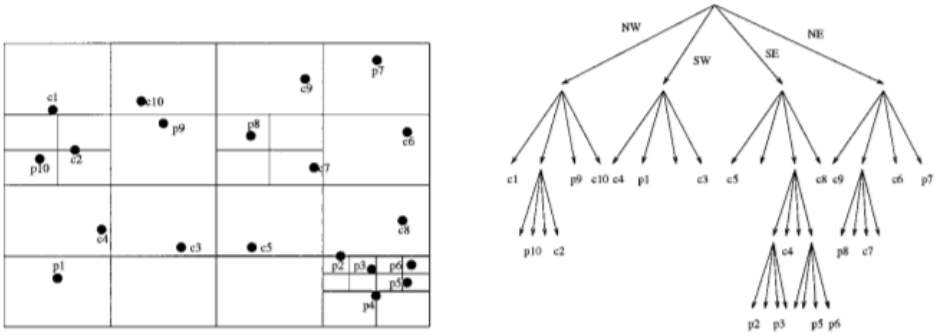


Figure 2.5: Region Quadtree [Volker and oliver, 1998]

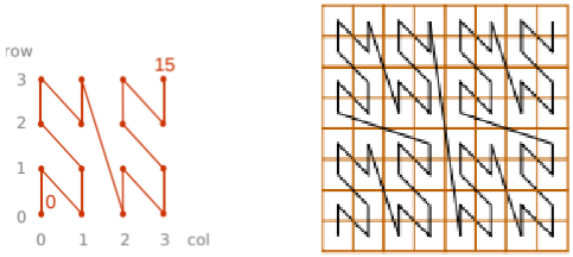


Figure 2.6: Morton curve

- Space filling curve
  A space filling curve is a great method for indexing the multidimensional data and it is also the method which is important in this thesis. The space filling curves map multidimensional space into the 1D space. The SFC is a thread that goes through all the points in the space while visiting each point only one time. Thus, the SFC imposes a linear order of points in the multi-dimensional space [Mokbel et al., 2003]. There are a variety of kinds of SFC, such as Morton, Hilbert, Gray, and so on. And the Morton curve (in figure2.6) and Hilbert curve (in figure2.7) are used in this thesis because of the property of the locality form the nD space is preserved in the location on the curve [Dai and Su, 2003] and both are so-called quadrant recursive curves [Meijers and van Oosterom, 2018] which is the very significant property. It is precise because of this property of these two curves that they can be connected to the quadtree so that the key value of SFC has a fixed relationship with the code of the quadtree. This is also the basis of the method I will use in my thesis. This clustering reduces the number of disk accesses and improves the response time[De Vreede, 2016] At the same time, the SFC can be extended to multidimensional space which is used to make the index for the multidimensional space.

  The Morton curve is relatively easy to make compared with the Hilbert curve. The principle of making the Morton curve is the bit-interleaving. The data in each dimension should be converted

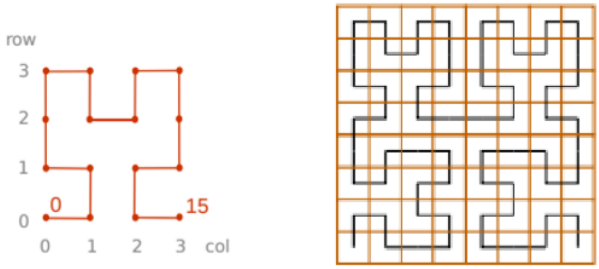
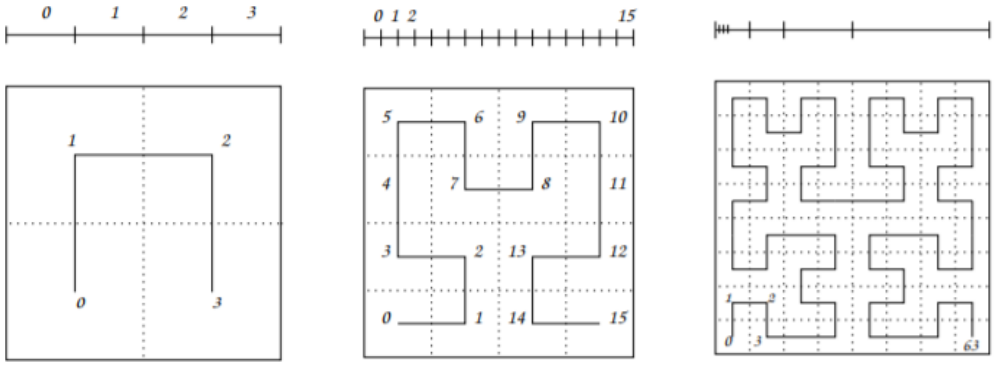


Figure 2.7: Hilbert curve



(a) a (b) b (c) c

Figure 2.8: The hilbert curve in 2D space

to the binary and interleave the bits of each coordinate. For example: (x, y, z) = (2, 6, 1) = (010, 110, 001) = (010 011 100) = 156th cell along the Z curve. While, for the Hilbert curve, it is not very easy to make. For the data in 2D space. the square is divided into 4 sub-squares, then the rotating and mirroring need to be done based on the first step. And the Hilbert curve with different depths in 2D space shows in figure2.8.

The great advantage of the SFC is that the one-dimensional data structure method which is more mature and much easier to implement can be used in the high-dimensional data.

- Relationship betwwen SFC and Quadtree
  The principle of decomposition of the space using quadtree or a 2^n tree can be combined with the space filling curve to map the nD space to 1D space [Psomadaki, 2016]. There is an existing relationship between the Morton keys in 2D space and the quadtree which is proposed by [Gargantini, 1982]. Because the Morton curve and the Hilbert curve are both recursive, the Hilbert key in 2D space also has a similar relationship with the quadtree.

2.2.2 Manage data in Not only SQL (NoSQL)

A NoSQL originally referring to non SQL or non-relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases `https://www.geeksforgeeks.org/introduction-to-nosql/`. NoSQL systems are distributed, non-relational databases designed for large-scale data storage and massively-parallel data processing across a large number of commodity servers[Moniruzzaman and Hossain, 2013]. Without exception,non-relational databases abandon the relational model and adopt column storage, key-value storage, and document-oriented storage, which effectively solves the problem of poor read-write performance and scalability in mass data management[PAN et al., 2016]. There are three main characteristics of NoSQL databases, they are strong consistency, high availability, and partition tolerance[Moniruzzaman and Hossain, 2013]. MongoDB has become the most widespread and popular solution due to its open nature, especially the feature of native support for geospatial indexes, which gives it a natural advantage in storing massive distributed geographic information sharing data[PAN et al., 2016]. PAN et al. proposed using batch data migration to migrate storage and AIS data in the Oracle database to MongoDB database. It uses the characteristics of the MongoDB database to improve the storage performance of AIS data, but the important export of tedious format data makes the data unavailable. It is used in applications with high real-time requirements, such as VTS ship monitoring. The rising interest in NoSQL databases does not mean that such a database is the best suited for all use cases[De Vreede, 2016]. Each database offers different solutions for specific cases. Hence, it is important to choose the proper database when facing a different situation. The study about the management of AIS data in the relational database is still important.

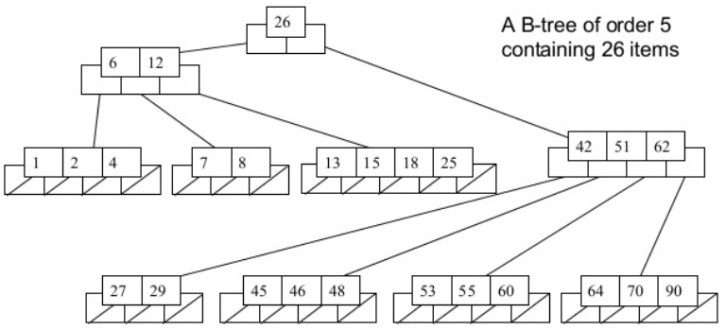


Figure 2.9: B-tree

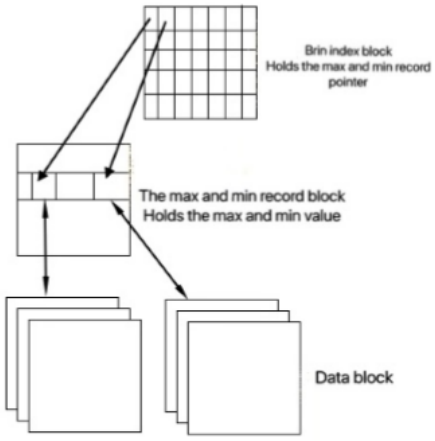


Figure 2.10: BRIN index

2.3 One-dimensional indexing technology

- B-Tree
  Today almost all database systems use B-trees as their main access method. And the structure of B-tree shows in figure2.9. It is the binary search tree and can store more than two keys per node and it is always balanced after insertion, deletion, or updating [Psomadaki, 2016]. A B-tree of order m is a search tree in which each non-leaf node has up to m children[1]. Here is a B-Tree example (m=5):

- BRIN index
  BRIN is the Block Range Indexes, which are, as the name implies, the Indexes to sections of a Block of data. The design idea is very simple, is to scan the whole table, record the data contained in each fixed section of the maximum and minimum values of the indexed fields, in turn into the index space. When a query needs to be processed to find records that match the query criteria, the BRIN index can be used to speed up the lookup by skipping sections that do not match the query criteria [database kernel group, 2015].

  The BRIN index consists of a set of identical index blocks, each containing a fixed number of index records, each containing a pointer to the most valuable block. Each record in the most valuable block holds the maximum and minimum values of the extents and the block number of the corresponding data section start block [database kernel group, 2015].

[1] from https://www.cs.cornell.edu/courses/cs3110/2012sp/recitations/rec25-B-trees/rec25.html

BRIN indexes are efficient if the ordering of the key values follows the organization of blocks in the storage layer. In the simplest case, this could require the physical ordering of the table, which is often the creation order of the rows within it, to match the key's order. Keys on generated sequence numbers or created data are the best candidates for BRIN index[Augustine, 2019].

2.4 Relevant research

Psomadaki proposes a method that uses a space filling curve for the management of dynamic point cloud data. De Vreede implemented Python to compute the SFC key technique in combination with the MongoDB database and tested this with moving ship trajectory data of the Automatic Identification System (system). Meijers and van Oosterom achieves the needed steps for making SFC approach available fully inside a DBMS, clustering, and indexing historic vessel movement data with SFC. And for now, the studies are mainly focusing on dealing with 3D data, how to manage the data in higher-dimensional still needs to be done.

3 Methodology

This MSc thesis is going to research how to efficiently manage massive moving objects (vessels) by using Space Filling Curve to achieve efficient data storage and data query. And the SFC approach will be fully used inside the DBMS to reduce the costly data transfer in the procedure when dealing with data. The 4D AIS data (longitude, latitude, MMSI of vessels, and the corresponding time) which is closely related to the behavior of the ship is the data I am going to deal with. And the bounding box query and trajectory query will be used to test the efficiency of different methods dealing with 4D AIS data. This chapter is used to introduce the main steps required to achieve this goal and to introduce the methods that need to be tested and compared in each step. The approach is divided into 3 main steps: data preparation, data clustering, and indexing, and performance test. The overflow of the procedure will be shown below. Section 3.1 introduces the detailed steps will be implemented in preparation of the AIS data to make it suitable for use in the next steps. Section 3.2 describes the different Space filling curve indexing methods that will be used to deal with 4D AIS data and the possible alternatives in the procedure of the SFC approach. Lastly, section 3.3 summarizes the queries which are intended to be tested and the key performance indicators which need to be compared to give the proper and correct results. Figure 3.1 shows the overflow of the approach.

3.1 Data preparation

As mentioned in related work, the original AIS is decoded by certain rules, and there is much useful information in the AIS data after interpretation that is of great significance to lots of applications. To study the moving object (vessels), three attributes need to be focused on, that are the longitude of vessels, the latitude of vessels, and the corresponding time, which are crucial to the spatial-temporal analysis. Nowadays, the analysis of the behavior of ships is becoming increasingly important, and simple spatial-temporal analysis is not enough. The processing of multi-dimensional data is becoming more and more important. As I mentioned in related work, There have been some studies on how to use three-dimensional AIS data. The three-dimensional AIS data here refers to the longitude, latitude, and corresponding time of the vessel. In my research, I will process four-dimensional AIS data (4D AIS data), adding MMSI data to the original three-dimensional data. Hence, the 4D AIS data I am going to use refers to the longitude, latitude, corresponding time, and MMSI of the vessel. And more analysis can be done based on managing 4D AIS data, such as the trajectory analysis of vessels. I am going to consider the MMSI which is also a piece of useful information together. Therefore, the data I intend

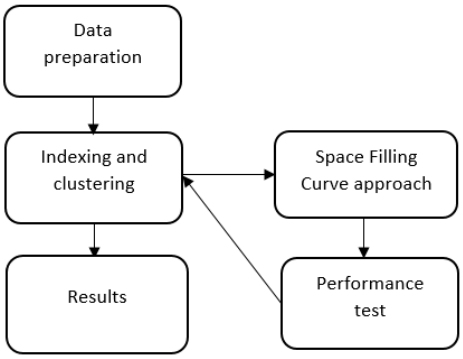


Figure 3.1: Flowchart of the approach

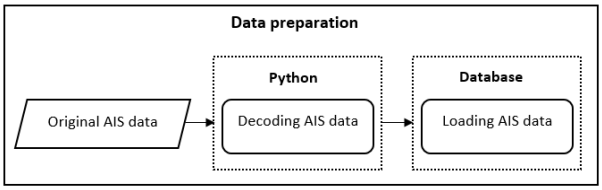


Figure 3.2: Flowchart of data preparation

```
!AIVDM,1,1,,A,13aENUhP00PBwE2Mfg3PPgw`P@00,0*2E,2016-12-10 00:00:00.056,/10.0.201.3
!AIVDM,1,1,,B,1CAO2j7Oic0G<d8NiSgnvmrJ2H7l,0*1B,2016-12-10 00:00:00.084,/77.163.66.94
!AIVDM,1,1,,A,13aGs:PP00PCQwlMEB8=:gv00802,0*4C,2016-12-10 00:00:00.113,/81.243.244.33
!AIVDM,1,1,,B,133v1kPP00PD9I<MDRWD5Own2@02,0*3B,2016-12-10 00:00:00.130,/81.243.244.33
```

Figure 3.3: Example of raw AIS data

to use are the longitude and latitude of vessels, the corresponding time, and MMSI of vessels in the AIS data. To make the SFC approach fully inside the DBMS which is used for avoiding the expensive data transfer to external programs during use [Meijers and van Oosterom, 2018], the data will be loaded into the database directly after decoding. And the overall flowchart of data preparation shows in figure 3.2:

3.1.1 Decoding AIS data

The AIS data is decoded by certain rules. AIS receivers report ASCII data packets as a byte stream over serial or USB lines, using the NMEA 0183 data formats [1]. As the figure 3.3 shows below, the AIS packets usually have the same introducer "!AIVDM" or 'AIVDO' [2]. These encoded lines of data ask for decoding and reformatting pursuits before implementation in the specified data organization into database is needed[De Vreede, 2016]. There are a great many of message type in AIS data, such as position report (code:1,2,3), base station report (code: 4), static and voyage related data report (code: 5) and so on `https://en.wikipedia.org/wiki/Automatic_identification_system`, and each type has their unique code. Therefore, we can decode the AIS data regarding the rules and acquire a certain type of data that is needed. In my research, the 4D AIS data is used for analyzing the behavior of vessels. The position report is the information needed.

3.1.2 Loading AIS data

In order to make space filling curve fully inside database, it is crucial to load the decoded AIS directly to database. As we all know, the data types of the data need to be explicit when loading data to database. And the type of data is also important in the following steps. From the previous steps, we can know that all information in the position report will be loaded when loading. While, the 4D AIS data I want to use have been clarified in the previous article, in order to lay the groundwork for the subsequent experiments, I need to clarify the data type of 4D AIS data, e.g. the data type of the MMSI is integer.

3.2 Clustering and indexing

Clustering and indexing are the most important step in the whole process. Retrieving fast output from queries makes cluster and index techniques for smart storage of the data necessary. Different cluster and indexing techniques exist to provide close storage on disk for related data and fast retrieval of this data[De Vreede, 2016]. The Space Filling Curve is the main indexing method I use in this thesis. And

[1] more information can be found in this website:`https://gpsd.gitlab.io/gpsd/AIVDM.html`
[2] More information can be found in this website: `https://gpsd.gitlab.io/gpsd/AIVDM.html`

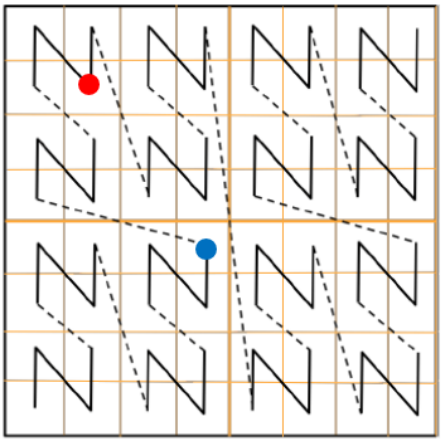


Figure 3.4: Morton curve in 2D space

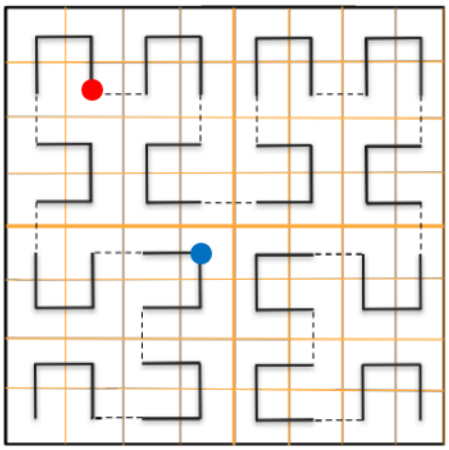


Figure 3.5: Hilbert curve in 2D space

the B-Tree index and BRIN index are often used as auxiliary indexes of the SFC index method. These two indexes are made on the SFC key which is in 1D space to support the efficient query. The section 3.2.1 will give the introduction about the Space filling curve approach, the section 3.2.2 will explain the details about the process of the different clustering and indexing method based on the space filling curve approach and the possible alternatives during the procedure.

3.2.1 Space filling curve appraoch

The space filling curve approach is one of the greatest indexing methods. Utilizing the SFC, the nD coordinates are mapped to a 1D coordinates which are the positions on the SFC curve[Meijers and van Oosterom, 2018], which is helpful when dealing with multidimensional data.

Motivation of using SFC approach

Space filling curves enable higher dimensional objects to be expressed, stored, analyzed, and categorized in one dimensional space[**?**]. There are many types of SFC, such as Morton curve, Hilbert curve, Moore curve, and so on. The main reason regarding why I am going to use Space Filling curve is that the SFC can reduce dimension. A space filling curve (SFC) is a mapping from a multidimensional universe to a single-dimensional universe. And it usually has great clustering properties. In addition to that, the SFC as an organization structure is to get fast access to selections of nD points as the records for the points can be sorted along the curve because of the preservation of the locality[Meijers and van Oosterom, 2018]. The curves I am going to use in my thesis are Morton curve and Hilbert curve (in figure 3.5) also because these two curves both have the great characteristic of locality preservation.

Principle of SFC approach

The key point in the SFC approach is the encoding of the data in each dimensional, in other words, it is to encode multidimensional data into the position on the curve used. There are examples in 2D space (the level is 3, which will be explained later) describe what encoding means in the SFC approach.

As the figure 3.5 shows, If the horizontal direction is regarded as the x-axis, the vertical direction as the y-axis, and the lower-left corner as the origin, then the coordinates of the red point in the figure are (1, 6), and the coordinates of the blue point are (3, 3). The red dot has a Morton code of 22 and a Hilbert code of 23. The blue dot has a Morton code of 15, and its Hilbert code is 10. The data in two dimensional

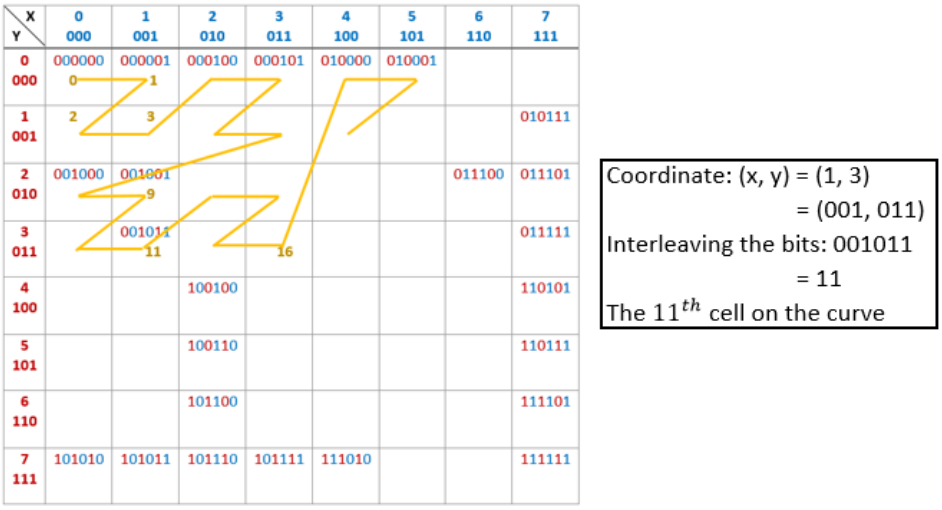


Figure 3.6: Principle of bit interleaving

converted to the 1D data.

There are already several existing algorithms that could calculate the Morton key and Hilbert key. The algorithms I am going to use are written by Martijn Meijers, which are fully implemented by python. The principle to calculate the key of the Morton curve is bit-interleaving. The value in each dimension has to been converted to binary and then are interleaved with each other. The figure 3.6 shows the principle of generating Morton code in 2D space.

And for Hilbert curve, it is the variant of the Peano curve. The method to compute the code for the Hilbert curve is a bit complex compared to the Morton curve. Lots of rotation and mirroring are needed when creating the Hilbert curve. It is not difficult to imagine that it takes a lot of loops to calculate the Hilbert code. In contrast, the Morton code obtained by bit interleaving is easier to obtain. Both Morton curve and Hilbert curve are so-called quadrant recursive curves[Meijers and van Oosterom, 2018] and both curves are useful in various kinds of application.

Integrated SFC appraoch and non-integrated SFC appraoch

Psomadaki's thesis gives a detailed description of these two SFC approaches. In short, the integrated SFC approach means that each dimension is going to be treated equally in the process of encoding the SFC key. The data in each dimension will be encoded in SFC key. The non-integrated SFC approach which means that data in a few dimensions is not encoded into SFC key, but stored as separate columns in the database. Both methods are the methods I will use when managing 4D AIS data.

Partial resolution key and full resolution key

To using the SFC approach, first of all, we need to be clear whether the SFC key which acquired through encoding is the full-resolution key or partial resolution key. There are two ways to calculate SFC key. One is using partial resolution key, the other is using full resolution key. Just as its name implies, full-resolution key is the key that stores and retain the complete original information. With full resolution keys the original attribute (= dimension) values can be calculated from the SFC key without loss of information, in this case, the values for the dimensions do not have to be stored explicitly [van Oosterom et al., 2018]. For partial resolution key, since we usually zoom out the values of each dimension when calculating the partial resolution key for efficient storage, and the value of each dimension must be an

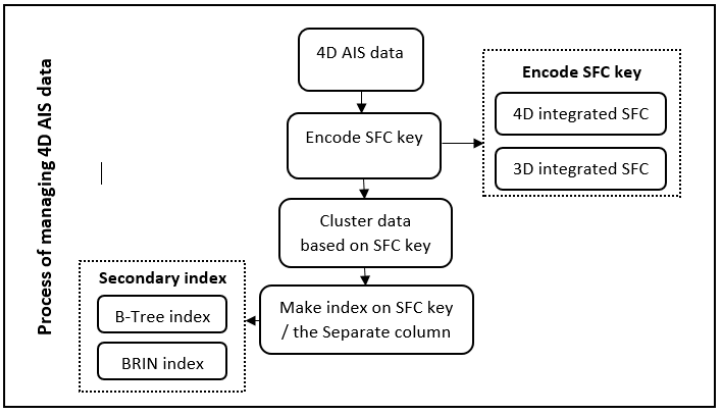


Figure 3.7: Process of managing 4D AIS data

integer when calculating the SFC key. Due to the problem of rounding the value, some information may be lost during scaling. Thus a key may correspond to many pieces of information. Therefore, if the partial resolution key is used, its original data cannot be discarded and needs to be stored in the database for the query.

In my research, the approach I am going to use is the full resolution key to avoid too much meaningless data being selected because their corresponding SFC keys are the same. As mentioned above, when we use the full resolution key, the values of each dimension can be obtained by decoding the SFC key, so they do not have been stored explicitly.

3.2.2 Managing 4D AIS data through SFC approach

The core of this thesis is how to efficiently manage 4D AIS data through space filling curve. There are two main ways to manage 4D AIS data. These methods are to use the 4D integrated SFC approach, 3D integrated SFC approach. While, there are many uncertain factors in this method, such as how to adjust and scale the original data of each dimension to calculate the SFC key to achieve fast query. This is also the part I need to focus on in my paper. And the figure 3.7 shows the brief process of managing 4D AIS data

- Define the type of SFC key
  According to the introduction above, I will use the full resolution key in this paper. And the integrated SFC approach and non-integrated SFC approach will be used to manage the 4D AIS data.

- Define the scaling method
  In order to keep the complete information and ensure that the value of each dimension is an integer to calculate the SFC key, the scaling is an important step. First of all, we must follow the principle when calculating SFC code, that is, the value of data in each dimension must be an integer. The quality of the scaling method can also determine the quality of the SFC method. There exist a great many methods to scale the data I am going to use, and the key to the problem is to ensure that all dimensions in the curve should have a similar size [De Vreede, 2016] which makes the data in each dimension will be treated equally.

- Define the indexing method
  The main indexing and clustering method I am going to use is the SFC approach. The indexing methods which are made on the SFC key are used for the better querying. B-Tree index and BRIN index both are great indexing method that utilized in 1D data.

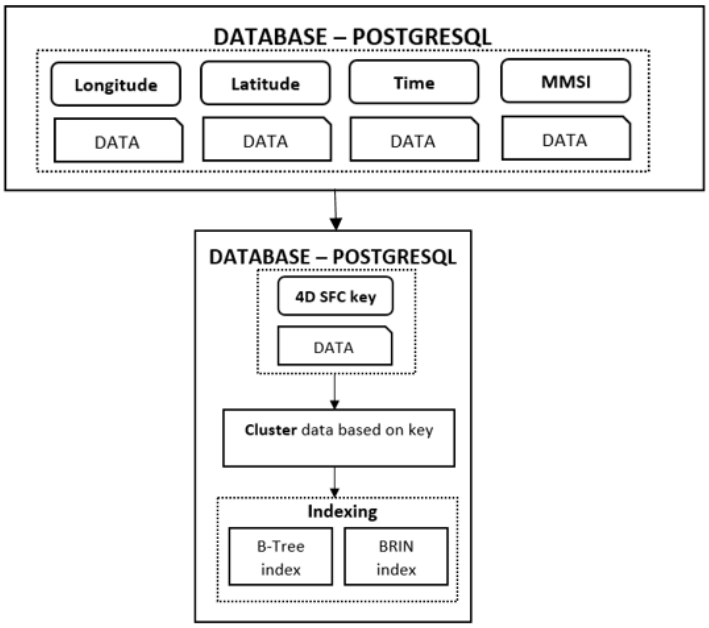


Figure 3.8: Process of 4D integrated method

From the introduction about the whole procedure of managing 4D AIS data, we can know that there are three main methods to manage the 4D AIS data. And there are two secondary indexes we can choose to make on SFC key. The comparison of two secondary indexes will be implemented in 4D integrated SFC approach because there is only one column (4D SFC key) needs to utilize the secondary index, the results of the comparison will be explicit. The rest method will use the B-Tree index as the secondary index.

4D integrated SFC approach

The 4D integrated SFC approach is to encode the data in 4 dimensions (longitude, latitude, time, MMSI) which have been scaled properly together and keep the 4D SFC key only in the database (shows in figure 3.8). For the 4D integrated SFC approach, the Morton curve and Hilbert curve will be used, and the comparison between these two curves will be introduced. Also, the B-Tree index and BRIN index will be made on the 4D SFC key to be compared. For the rest approach, only the Morton curve will be used and the B-Tree index will be implemented because the Morton curve is easy to implement and the B-Tree index is more common to be used. The main idea for the thesis is to see whether the SFC approach can be used to deal with 4D data.

3D integrated SFC approach + 1D data

This is the 4D non-integrated SFC approach, it encodes the data in 3 dimensions (longitude, latitude, time). I chose to put these three columns together because MMSI is a 9-digit data, and I use the full-precision SFC key, which will cause the key to be too long and take up too much storage space when calculating the sfc key. Not only that, but I also considered the query I used. This will also be introduced in the section 3.3. Hence, as the figure 3.9 shows, there are the 3D SFC key and a separate MMSI column in database.

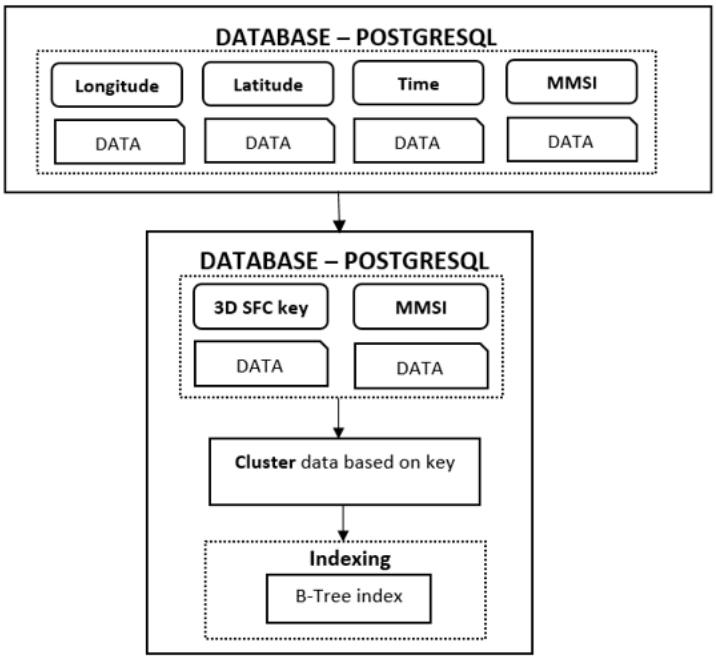


Figure 3.9: Process of 3D integrated method

3.3 Performance test

The querying is used as the performance test for indexing methodology. In this thesis, I am going to implement two kinds of query which is useful for lots of applications to test, one is the bounding box query and the other is the trajectory query. The bounding box query is to find which vessels are in the give space range during a period. And the trajectory query is to find the historical positions of a specific vessel in the given time range. The section 3.3.1 introduces the method of querying space filling curve and the section 3.3.2 gives the overview of the query procedure regarding the different indexing method.

3.3.1 Query space filling curve

To implement the query based on the SFC approach, the quite new but efficient query algorithm is proposed by [Psomadaki, 2016]. The core of the query algorithm is to connect the Quadtree structures with SFC curve.

Principle of query SFC

The Quad-code [3] has a special relationship with the Morton curve and Hilbert curve. Psomadaki uses the corresponding relationship between the quadcode and the SFC key. He used SFC at different levels to approach the query geometry by decomposing the Minimum Bounding Box of the dataset recursively. The Quadcodes are acquired by checking the spatial relationship between the Quadcode and the geometry, the Quad code will be selected if the Quadcode intersects with the geometry (see the example in figure). The quadcode will be transformed to the SFC key and the consecutive keys will be merged to reduce the time of query. Then we can acquire lots of ranges of SFC key which contains the whole

[3]Quadtree partition a two-dimensional space by recursively subdividing it into four quadrants or regions [4], Quad-code represents the 4 quadrants. In this thesis, '0' represents the SW quadrant, '1' stands for SE, '2' stands for NW, and '3' stands for NE. And each successive digit on the string represents the quadrant of a deeper level [Psomadaki, 2016].

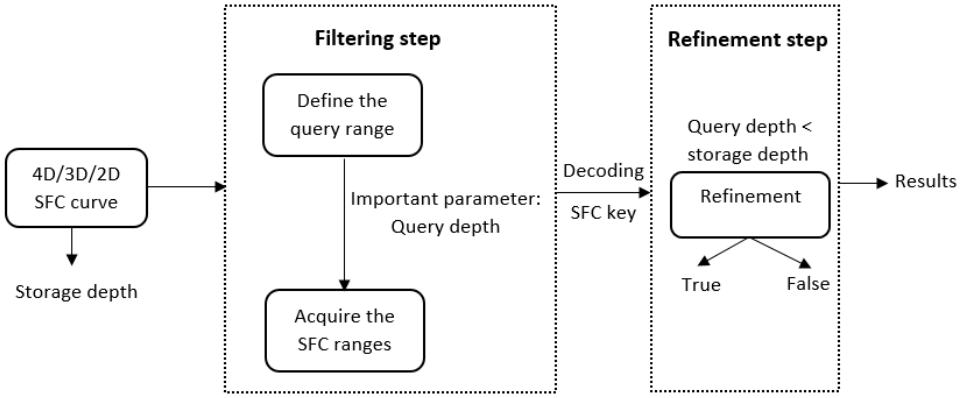


Figure 3.10: the brief query process

information after querying.

Process of query SFC

Here is the flow chart of the procedure of the query shows in figure 3.10. We can see from the figure that the query procedure is mainly divided into two steps: filtering step and the refinement step. And there are two significant parameters during the process which are the storage depth and query depth. The definition will be introduced in the following part.

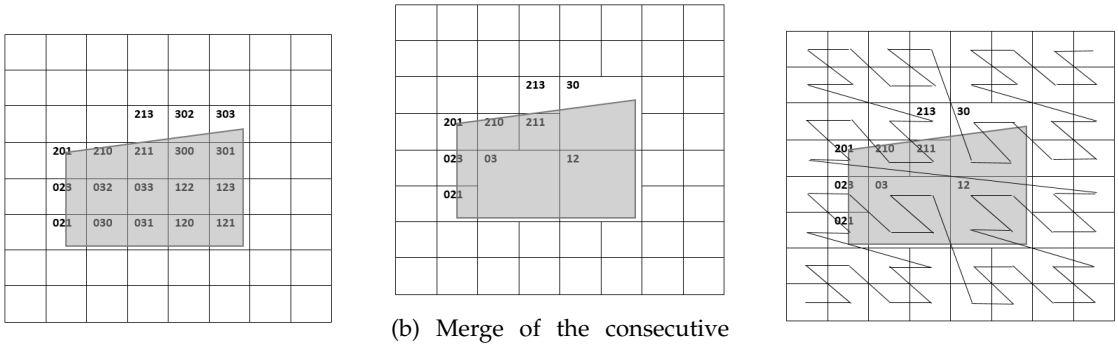
- Filtering step
  The first I am going to explain is the storage depth. The Quadtree with a depth of n is used to represent the region consisting of

  $$2^n \times 2^n$$

  small ranges [5]. Because the special characteristic (both are quadrant recursive curves) of the Morton curve and Hilbert curve, Morton curve and Hilbert curve can also have the different depth to represent the data (That is why I wrote the 'level 3' in the previous figure, the depth of the curve I used is 3). To query the SFC, the specific level used to store the whole information need to be defined, which is determined by the largest number among each dimension regarding using full resolution key (the storage depth is 3 in the following figure). The query depth is the depth that I am going to use when querying.

  Here I will give an example to introduces the importance of the storage depth and query depth. The main difference between these two depths is that the storage depth is determined by data itself and it is set beforehand. While the query depth can be decided ourselves and the query depth could influence the results of query. As I mentioned in the previous paragraph, the storage level in figure 3.11 is 3. Then the query range is given which is the grey polygon. Then the decomposition algorithm determining the quadcodes strings which represent the query range regarding to the query depth, and different SFC ranges will be selected because of the query depth. In figure 3.11a we can see that the query range is cover several quadcode string, they are represented by the bold numbers in the figure. And then the neighbor will be merged regarding to the query depth, the query depth in figure 3.11c is 3 which is as same as the storage depth. In the figure 3.11, the quadcodes are connected with code of Space filling curve (here is the Morton curve). As we can see that, the quadcode 021 represents the Morton code 9, the quadcode 03 stands for the Morton range 12- 15, and the rest can be deduced by analogy.

[5] from Wikipedia(Quadtree)

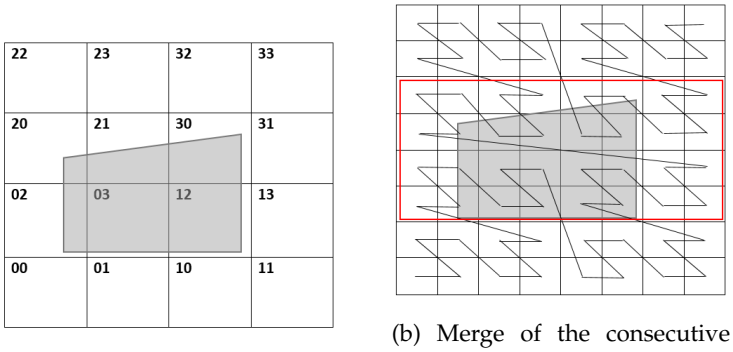


(a) Query range with quadcode

(b) Merge of the consecutive range

(c) Connect with SFC code

Figure 3.11: Example of query SFC



(a) Query range with quadcode

(b) Merge of the consecutive range

Figure 3.12: Query depth =2

We can see from the figure 3.11 that when the query depth is as same as the storage depth, no redundant SFC ranges will be selected. While, what happens when the query depth is less than the depth used for storage, we can see the following example. As the figure 3.12 When the query depth is 2, the SFC code in the red rectangle will all be select. And when the query depth is 1 (shows in figure 3.13), all SFC code will be selected. In practical applications, when the data is very large, the depth used when storing data is very large. If we use the same depth when querying, the query time will be greatly increased. Although the selected SFC code is not redundant, there will be many SFC ranges used for describing the query range. However, when the query depth is too small, too many SFC codes will be selected and the query efficiency will be reduced. Therefore, to improve query efficiency, we should find the proper query depth that best matches the data we use. And this step which aims to select the real SFC range names filtering step.

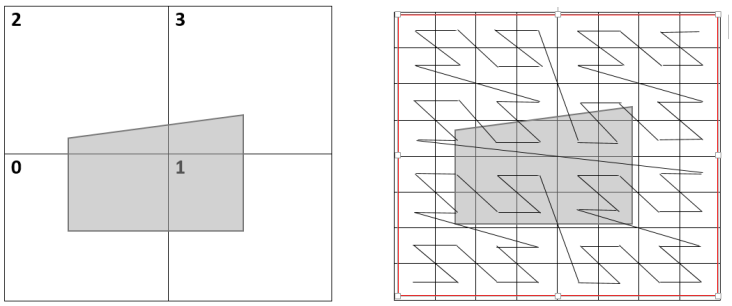


Figure 3.13: Query depth = 1

- refinement step
  From the above explanation, we can see that as long as the depth of the query is not equivalent to the depth used for data storage, there will be extra SFC code selected. Therefore, refinement is necessary, which is to find the SFC code that fully inside the query range. The SFC key will be decoded back to the original data regarding to the scaling method used before. And the data is going to be determined whether the point is within a query range (the shape of the query range I am going to use is a rectangle). Finally, we can get accurate results.

Summary of query steps

In the previous subsection, I introduced how to query SFC. Here I will summarize this process and its key points.

- Determine the query range
- Clear the storage depth according to dataset
  Because I use the full resolution SFC key, the storage depth which is determined by dataset itself depends on the maximum value in dataset.
- Determine the query depth
  The query depth can take any value (its value cannot be greater than storage depth), but it is worth noting that it will directly affect the efficiency of the query.
- Decompose the space
  Regarding to the query depth, the space will be divided into blocks of the same size. e.g. for the nD data (query depth is s), the space will be divided into

  $$(2^s)^n$$

  blocks. And each block has its quadcode which could be transformed into SFC key later.
- Determine the relationship between query range and the blocks (filtering step)
  Judge the geometric relationship between the query range and the block, and there are several possible results: intersection, inclusion, and separation. Then we can select the blocks which intersect with query range or be included in query range. These blocks are what we want.
- Acquire the truly SFC ranges (refinement step)
  Although the SFC range intersects with the query box, SFC keys in SFC range may not all in the query box. Hence, the SFC keys within a SFC range needs to be checked. And the SFC keys needs to be decoded to give an exact value which is used for the spatial check (whether the point is inside the query box).
- Decode the SFC ranges to the original data

3.3.2 Query regarding to different indexing approach

In my thesis, the bounding box query and the trajectory query will be implemented to test the efficiency of managing 4D AIS data I used. And first I will introduce the two queries:

- Bounding box query
  The bounding box query is to find all the vessels in an given area and time range.
- Trajectory query
  The trajectory query is to acquire the position information (here is the longitude and latitude) of a specific vessel in a period of time.

To achieve these two queries, The data needed is the longitude, latitude, time and ID of the vessels. And this is also the four-dimensional data we want to process.

The query method is a little bit different when dealing with the different indexing approach. And the detailed steps will be mentioned in next chapter.

4 Implementation

This chapter mainly describes the implementation to present whether the SFC approach can provide an efficient way to manage the 4D AIS data (longitude, latitude, Time, MMSI). The detailed principle and the process of the whole approach have been described in the previous chapter. The specific implementation and testing of the method will be highlighted in this chapter. And the chapter is organized as follows: The tools and datasets will be firstly introduced in Section4.1. Then, Section 4.2 will mention the use cases which are used to test the SFC approach. After that, the detailed steps of the implementation will be given in Section 4.3. Finally the section 4.4 will be shown.

4.1 Tools and the datasets

As mentioned in the previous chapter, some software has been used during the procedure. Some Python scripts have been used to fulfill the demands. The scripts are used to decode the decoded AIS data and also used to encode and decode the SFC. In addition, there is also the scripts that select the SFC ranges according to the given range and query depth. The Python codes are implemented in PostgreSQL by PL/PYTHON.

4.1.1 Software

The software which is used in this research are:

- PYTHON
  The python programming language is used for encoding and decoding the SFC, making the query range, and selecting the SFC ranges according to the query range. The python scripts are embedded in PostgreSQL through PL/PYTHON extension. The PL/Python allows PostgreSQL functions to be written in the Python language (`https://www.postgresql.org/docs/10/plpython.html`). The version I use is Python3.7

- POSTGRESQL
  From the data loading to the query procedure, the whole process is in PostgreSQL. PostgreSQL is a general-purpose and object-relational database management system, the most advanced open-source database system [1]. And the version I use is PostgreSQL12.1.

4.1.2 Hardware

The laptop I am using is the HP ZBOOK STUDIO G5. The processor is Intel(R) Core(TM) i7-8750H. The RAM is 16GB and it is the 64-bit operating system.

[1] from (`https://www.postgresqltutorial.com/what-is-postgresql/`)

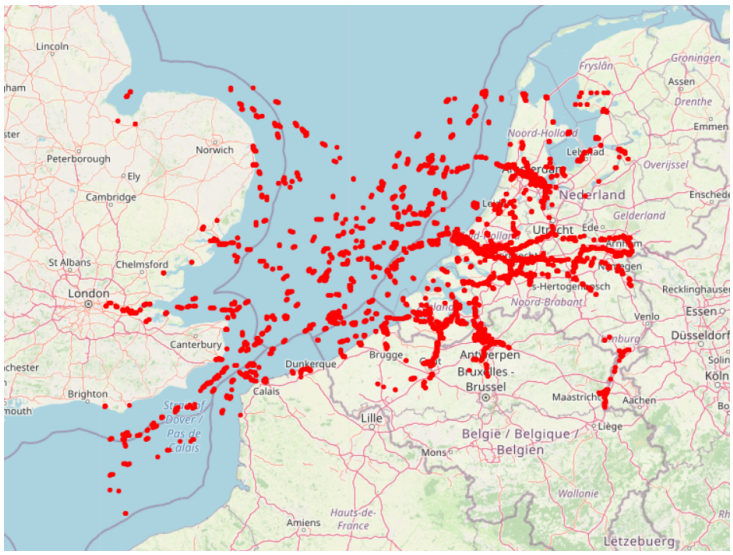


Figure 4.1: Dataset

4.1.3 Datasets

I will use the datasets that is the AIS data within 3 days (from 2016-12-10 to 2016-12-13). Here I select 2000 points to show the distribution of the vessels (shows in figure 4.1). And the data I am going to use is the longitude, latitude, time, and MMSI in AIS data which are acquired after selecting in database. The exact ranges of data in 4 dimensional are:

longitude: 0.00003 - 5.99999
latitude: 50.00030 - 52.99999
time: 2016-12-09 23:59:59.865 - 2016-12-12 23:59:59.957
MMSI: all 9-digit data

4.2 Metrics of performance

In order to clarify whether the SFC approach can efficiently manage 4D AIS data regarding to the method I propose, the metrics are important to be set. In my research, four metrics play important roles: 1. Fetching time, 2. Percentage of redundant points, 3. Query depth. 4. storage size for the data and index. Whether we can find a balance between these four metrics is our main concern.

Firstly, for querying in the database, fetching time is very important data. A quick query is what we want to achieve. And to see whether the data is well structured in the database, the fetching time is a vital key performace indicator(KPI). Then the percentage of the redundant points is another crucial metric in my research. As I mentioned in the methodology, some points which are not truly inside the query range will be selected due to the difference between the storage depth and query depth. Obviously, if the query depth is small, lots of SFC ranges will be selected and the step of the refinement which is made to select the points truly inside the database will take much more time. Hence, the more redundant time, the more time will be taken on refinement. From the description above, the importance of the query depth is clear. So, the balance between the percentage of redundant points and the query depth is attractive. And it is also interesting to find that whether the data in SFC form can be efficiently managed.

implementation plan		
Methods	Implementation	Number of ranges will be used
4D integrated approach	index comparison (B-Tree/BRIN)	2
	SFC comparison (Morton/Hilbert)	2
	bounding box query	4
	trajectory query	4
3D integrated approach	bounding box query	4
	trajectory query	4

Table 4.1: Implementation plan

mmsi	ts	imo	callsign	shipname	shiptype	bow	stern	port	starboard	month	day	hour	minute	draught	destination
244270394	2016-12-10 00:00:00.148 UTC	0	PE7278	STERN	69	4	7	1	1	0	0	24	60	0.7	HEEN EN TERUG
244670835	2016-12-10 00:00:00.861 UTC	0	PD3343	MARCHIENA	89	85	0	4	4	1	8	21	25	1.0	-
305648000	2016-12-10 00:00:01.184 UTC	9504047	V2FE2	PHOENIX J	71	136	15	11	12	12	10	0	0	7.1	ROTTERDAM
235192000	2016-12-10 00:00:01.216 UTC	9143506	ZNBJ6	HAPPY BEE	80	94	20	12	4	12	7	20	20	4.7	ZEEBRUGEE
244700938	2016-12-10 00:00:01.443 UTC	0	PD7425	P1	99	13	15	3	4	0	0	24	60	1.8	SCHIEDAM
235076272	2016-12-10 00:00:01.000 UTC	9398723	2CWA7	PARAMOUNT	HANOVER	80	207	43	28	16	12	8	4	0	11.3 TEXAS CITY
229630000	2016-12-09 23:59:59.194 UTC	9365960	9HA3465	X-PRESS MULHACEN		74	133	9	11	9	12	9	20	30	7.7 CARTAGENA

Figure 4.2: Screenshot of the static data

4.3 Implementation

This section will give a detailed description of the method that is theoretically mentioned in the methodology. And first I will give the implementation plan in table 4.1 which will give the answer to the main question and the sub-questions.

4.3.1 Data preparation

This step is the base of the whole process and it includes two parts: decoding data and loading data. And before that, there is one thing important needed to be done when testing query in database and that is to make the data use the single query path. The following statement is used:

SET max_parallel_workers_per_gather = 0;

- Decoding data
  The basic process is like this:

  $$\text{Original data} \rightarrow \text{Python script} \rightarrow \text{CSV file}$$

  Two kinds of CSV files will be acquired after processing by the python script which is written by Martijn Meijers. One is the static file (shows in figure 4.2) and the other is the dynamic file (shows in figure 4.3). The information in the static file is about the static and voyage related data which gives the such as length, width data of the vessels. And the dynamic data is the navigational information which includes the longitude, latitude, heading, speed of vessels.

  As we can see that the AIS data indeed has a great amount of useful information. As I mentioned before, the data I am going to use are longitude, latitude, time, and MMSI. Obviously, the file which includes dynamic information is what I want. Hence, I am going to import the dynamic data into database through the 'COPY' sentence and select the data that I use. Here I record the decoding time of the dataset (shows in table 4.2)which is computed through computing the start time and end time.

mmsi	ts	longitude	latitude	status	turn	speed	course	heading		
244660423	2016-12-10 00:00:00.034 UTC	4.22004	51.51316	0	128	9.4	11.1	511		
244700446	2016-12-10 00:00:00.035 UTC	4.8121	52.47067	0	128	0.0	360.0	511		
244670396	2016-12-10 00:00:00.049 UTC	4.2229	51.52581	0	128	0.0	181.6	511		
244740304	2016-12-10 00:00:00.052 UTC	4.82312	52.39269	0	128	0.0	334.9	511		
244670103	2016-12-10 00:00:00.056 UTC	4.14832	51.95736	0	128	0.0	13.0	511		
244710186	2016-12-10 00:00:00.113 UTC	4.26666	51.26235	0	128	0.0	337.0	511		
205488590	2016-12-10 00:00:00.130 UTC	4.40113	51.24207	0	128	0.0	104.5	511		

Figure 4.3: Screenshot of the dynamic data

Decoding time	Size of dynamic data
591s (9min51s)	812MB

Table 4.2: Decoding process

MMSI	Integer
Ts	Time without time zone
Longitude	real
Latitude	real
status	integer
turn	integer
speed	real
course	real
heading	integer

Table 4.3: Column types in database

import time
start = time.process_time()
decoding...
end = time.process_time()
decoding time = end - start

- Loading data The basic process is like this:

Create table in PostgreSQL → Copy data to database → Create new table → Extract the information I need

To create a table in the database, we must give the type of each column in the table. Based on the dynamic information, there are 9 kinds of information in dynamic data and I will assign the following types to each column:

Then we can create the table in database and select the data we need properly:

create table table_name(mmsi int, ts timestamp without time zone, longitude real, latitude real, ... , heading integer);

create table table_name(select mmsi, longitude, latitude, ts from table(the table created before));

The 'copy' instruction in PostgreSQL is to copy data between a file and a table which is described in detail in the PostgreSQL documentation.

\copy file_name from 'path' delimiter e\'t' csv header;

After extracting the information I need, the table in database looks like in figure 4.4. And we can also see the total rows that in table 4.4.

4.3.2 Clustering and indexing

The main method I will use in this paper is the space filling curve method. The SFC key will show in a separate column after computing. And the type of the column is NUMERIC in case the value would be

datasets	total rows
small datasets	11389217

Table 4.4: Rows in database

```
test=# create table ais_4d_data as (select mmsi, ts, longitude, latitude from ais_3ds);
SELECT 11389217
Time: 14472.198 ms (00:14.472)
test=# select * from ais_4d_data limit 5;
   mmsi    |          ts           | longitude | latitude
-----------+-----------------------+-----------+----------
 244660423 | 2016-12-10 00:00:00.034 |   4.22004 | 51.51316
 244700446 | 2016-12-10 00:00:00.035 |    4.8121 | 52.47067
 244670396 | 2016-12-10 00:00:00.049 |    4.2229 | 51.52581
 244740304 | 2016-12-10 00:00:00.052 |   4.82312 | 52.39269
 244670103 | 2016-12-10 00:00:00.056 |   4.14832 | 51.95736
(5 rows)
```

Figure 4.4: Data to be used in database

Longitude		
Step	Example	Conversion time in database
original data	51.51316	0
Multiply 100000	5151316	33.815s
Minus the minimum value of the column	(5151316 - 5000030) 151286	29.439s
Alter column type	Real to Integer	17.010s

Table 4.5: Scaling Longitude

large (using full resolution key).

Encoding data using SFC

The encoding step is an very important step in whole process which is achieved by python scripts. And the script is used in PostgreSQL through PL\PYthon. Before encoding the data to SFC key, the first thing to do is to scale the data. The reason why scale data is needed is because there are certain rules when calculating the SFC key. The most important thing is that the data of each dimension must be an integer. We can see from the above that the data types of the four-dimensional data I use are different, e.g. the type of the 'longitude' is the 'real', so they need to be unified.

- Scaling data
  In order to use the full resolution key, the information in each dimension should not be discarded. And the value in each dimension should be an integer at the same time. In addition, the size in each dimension should be similar in order to make the data in each dimension equal. According to the analysis of the 4D AIS data, MMSI consists of nine digits. The longitude and latitude values are all five digits after the decimal point and the Time is described by timestamp.

  For the longitude and latitude, I multiply 100000 to get the integer and I convert the time into UNIX time to get the integer. The UNIX time is a way to track time as a running total of seconds and the count starts at the Unix Epoch on January 1st, 1970 at UTC which has been introduced detailed in this website (`https://www.unixtimestamp.com/`). As I mentioned before, the type of the Time column is the timestamp without the time zone. While I have to create a new column to store the UNIX time after converting and the type of the new column is double precision. And I do not scale the MMSI because it is already integer. Because the value is a bit large after encoding to SFC key, I minus the minimum value of each column to reduce the space the data needs in database. The steps are the basic scaling method which show in table 4.5, table 4.6, and table 4.7. And the MMSI does not need scale because it is the 9-digits value and it is discontinuous data.

  After the basic scaling of the data in each dimension, some further steps will be implemented. And for the three methods I am going to use, scaling will be slightly different. In order to make

Latitude		
Step	Example	Conversion time in database
orginal data	4.22004	
Multiply 100000	422004	27.894s
Minus the minimum value of the column	(422004 - 3) 422001	22.822s
Alter column type	Real to Integer	17.113s

Table 4.6: Scaling Latitude

Time		
Step	Example	Time (small datasets)
orginal data	2016-12-15 00:00:00.034	0
Convert UNIX time	1481756400.034	34.712s
Multiply 1000	1481756400034	35.050s
Minus the minimum value of the column	428400169	33.113s
Alter column type	Double precision to Integer	16.312s

Table 4.7: Scaling Time

the data in each dimension equally (in a similar size), some further steps will be done. After the basic scaling, the range of data in each dimension is different from the past. The new range shows in table 4.8:

As we mentioned in the methodology, it is important to make the data in each dimension equal when encoding the SFC. For the different indexing methods, the scaling method is slightly different.

  - 4D integrated method
    When I intend to encoding 4D SFC, the data in 4 dimensional (longitude, latitude, time, MMSI)should be treated equally. Hence, I multiply 1000 for the longitude and latitude to make the data have a similar size.
  - 3D integrated method
    The 3D data (longitude, latitude, time) should be treated equally. Obviously, the longitude and latitude need to be multiplied by 1000.

- Encoding SFC
  As we mentioned before, there will be a new column to store the SFC and the type of the column is NUMERIC. The SFC I am going to use is the Morton curve and Hilbert curve. There are the Python scripts named 'test_sfc' about encoding and decoding the SFC and I implemented the

Ranges		
Longitude	0	599996
Latitude	0	299969
Time	0	259200092

Table 4.8: Ranges of data in each dimension

SFC	dataset
Morton key	7min32s
Hilbert key	21min42s

Table 4.9: Time spent on encoding SFC key

scripts in database by PL/PYthon. Here I give an example to describe how I use the PL/PYthon (the names of python functions which are used for encoding the Morton key and Hilbert key are nenc and henc):

CREATE FUNCTION hencode (ARR BIGINT []) AS $$
FROM test_sfc IMPORT henc
RETURN henc(ARR)
$$ LANGUAGE PLPYTHON3U

After encoding, the multidimensional data is converted to one-dimensional SFC key. Through 4D integrated SFC approach, 4D data (longitude, latitude, time, MMSI) is converted to SFC key. Similarly, 3D data (longitude, latitude, time) is converted to SFC key by applying 3D integrated approach. And I also counted the running time of encoding the Morton curve and Hilbert curve which shows in table 4.9.

Clustering data

I cluster the data by ordering the data in the table based on the SFC key. The ORDER BY operator is used to cluster the data in the table.

create table table_name (select * from orginal_table order by SFC_key);

Making index

The BRIN index and the B-Tree index are made on SFC key and separate column to support the efficient query.

To create B-Tree index in PostgreSQL:
create index index_name on table_name(column_name);

To create BRIN index in PostgreSQL:
create index index_name on table_name using brin (column_name) with (pages_per_range = 32);

As we can see that there is a parameter needed when creating BRIN index. And the parameter is the pages_per_range. It is the important parameter which determine the number of data in one block and it effects the efficiency of query as well. In my research, I will test the BRIN index with different pages_per_range to see which one is more suitable for my query. And the BRIN index will be compared with the B-Tree index in different aspects, such as creating time, storage space, and whether they can accelerate the query speed. As I mentioned before, I will use 4D integrated approach and 3D integrated approach. And there are different columns left in database when using different approaches. Hence, I will introduce the specific method.

- 4D integrated SFC
  As I mentioned in the methodology, the comparison of the BRIN index and BTREE index is done by using 4D integrated SFC. Hence, the BRIN index and BTREE index will be made on SFC key

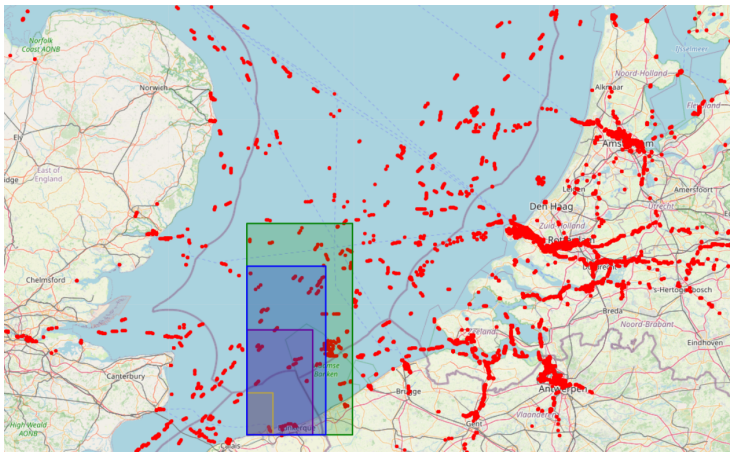


Figure 4.5: Different query box

one by one. And there is only one column left in database (4D SFC key), the index will be made on the 4D SFC key.

- 3D integrated SFC
  There are two columns (3D SFC key & MMSI) left after the encoding of SFC, the B-Tree index will be made on MMSI and 3D SFC key.

4.3.3 Querying

The performance of the different indexing approach is going to be tested by implementing the designed queries. There are two common queries that I am going to use: bounding box query and trajectory query. The queries are fully executing in PostgreSQL.

Query range

The query boxes show in figure 4.5 is the ranges that I am going to use. The yellow one is the smallest and the green one is the biggest.

Bounding box query

In order to do the query using SFC, some functions have been proposed to achieve it. Combining the principle of querying SFC mentioned in Methodology, an important function should be focused on. The function is to get the SFC ranges according to the given ranges.

The bounding box query is to ask all vessels in a certain range of an area in a period. The data the query needs is the longitude, latitude, time, and MMSI. The results of the bounding box query can be used to analyze the channel saturation in a certain area at sea. Because the bounding box query can not ask for the exact time and the AIS message is sent every 6s to 2 mins according to different types of vessels, the time interval of 2mins is going to be implemented to find all vessels in a given geographical area. For the different integrated SFC method, the queries would be a bit different to find the best way to do the query efficiently.

- 4D integrated approach Only the 4D SFC key is left in database after using 4D integrated approach and the index is made on the 4D SFC key. As I mentioned the 'summary of the query process' in the previous chapter, the following steps need to take to do the bounding box query:

dimension	minimum value	maximum value
longitude	lon1	lon2
latitude	lat1	lat2
time	t1	t2
MMSI	102044598	989799838 (the whole range of MMSI)

Table 4.10: The example of bounding box query using 4D integrated approach

– Create the query box
ultilizing the 4D integrated approach, what I am going to query is the 4D SFC. So the query box must be the 4D hyper box, that is, I have to give the query range in four dimensions to generate the 4D query box. The ranges show in figure 4.5 will be implemented. Here I will give an example to show how to generate the query box. If I am going to query which vessels are in the area ((lon1, lat1) - (lon2, lat2)) in the time period (t1 - t2), the query range in 4 dimensions will show in table 4.10. And we can find that the whole range of MMSI will be used to generate the query box because we can not make sure that which vessels are in the range we want to query. To process the hyper box in program, a special class named 'ndbox' is created for the hyperbox in Python.

– Filtering step
Filtering step is to find the SFC ranges regarding to the query depth applied by judging the relationship between the blocks and the query box. As I mentioned before, more SFC ranges may be selected when query depth is smaller that the storage depth. To acquire the SFC ranges, the function named 'nquery' and 'hquery' in python will be used. These two functions have the same parameters which are the ndbox and the query depth. The function returns the SFC ranges (by giving the minimum and maximum value of the SFC range) and a boolean value which means whether the range is fully inside the query box or intersect with the query box.

```
def nquery(query, query_depth)
...
result = []
result.append((start, end, added_value))
return result
```

Also, the PL\PYthon will be used to allow the function in Python can be used in PostgreSQL. Here is the code which shows the filtering function when using Morton curve in PostgreSQL:

```
CREATE FUNCTION sfc_nquery (lo integer[], hi integer[], query_depth integer)
RETURNS TABLE(lower numeric, upper numeric, in_out integer) as $$
from test_sfc import nquery
from test_relate import relate
import test_relate
return nquery(test_relate.ndbox(lo, hi), query_depth)
$$ LANGUAGE PLPYTHON3U
```

Hence, we can use the following instruction to acquire the SFC ranges in PostgreSQL:

```
select * from sfc_nquery(array[102044598, lon1, lat1, t1], array[989799838, lon2, lat2, t2],
query_depth)
```

if the query depth is smaller than the storage depth, more SFC ranges will be selected and the refinement step is needed. Through the above steps, we can get the number of SFC ranges and the range (minimum and maximum) of each SFC range. In order to understand more

dimension	minimum value	maximum value
longitude	lon1	lon2
latitude	lat1	lat2
time	t1	t2

Table 4.11: The example of bounding box query using 3D integrated approach

intuitively how many SFC keys are selected in order to better study the efficiency of query, we can get through the following SQL statement which join the two table together: (the filtering step datasets is the table that includes the SFC ranges, and there are three columns in this table, they are minimum value, maximum value, and a boolean value which shows the relationship between the query box and the ranges.)

select SFC_key from original_dataset a, filtering_step_datasets b
where a.morton_key $\geq$ b.lower and a.morton_key $<$ b.upper;

– Refinement step
The principle of refinement step is to decode the SFC ranges which intersect with the query box to find the SFC ranges that truly inside the query box. And the principle to find the 'truly data' is to find the orginal data according to the SFC keys which is achieved by decoding SFC keys, and the decoded data in each dimension will be compared to the each dimension of the query box which is achieved by function 'overlap'. To avoid us decoding all SFC ranges which acquire from the filtering step, we only decode the SFC ranges which are not fully inside the query box (can be known from the in_out value). Also, the function of decoding is in Python, we use it by PL\PYthon.

CREATE FUNCTION ndecode (value numeric, dims integer)
RETURNS bigint[] AS $$
FROM test_sfc IMPORT ndec
RETURN ndec(value, dims)
$$ LANGUAGE PLPYTHON3U

And the refinement step will be execute by following instructions. And the function overlap in following instruction is to judge whether the point is inside the query box.

The function overlap:
CREATE FUNCTION overlap (lo bigint[], hi bigint[], onekey_list bigint[])
RETURNS boolean AS $$
1. ***for i in range (ndims):***
2. ***if onekey_list[i] $\geq$ lo[i]***
3. ***continue***
4. ***else***
5. ***return False***
6. ***return True***

$$ LANGUAGE PLPYTHON3U

The refinement instrction:

select SFC_key from original_dataset a, filtering_step_datasets b
where (b.in_out = 0 and a.morton_key $\geq$ b.lower and a.morton_key $<$ b.upper) or
(b.in_out = 1 and a.morton_key $\geq$ b.lower and a.morton_key $<$ b.upper and overlap(array[102044598, lon1, lat1, t1], array[989799838, lon2, lat2, t2], ndecode(a.morton_key, 4));

- 3D integrated approach The 3D integrated approach is similar with the 4D integrated approach. The difference between these two approach aiming the bounding box query is the query box. For 3D integrated approach, only data in three dimensions is going to generate the 3D hyper box. And the ranges in each dimensions show in table 4.11.

dimension	minimum value	maximum value
longitude	minimum data	maximum data
latitude	minimum data	maximum data
time	t1	t2
MMSI	S	S+1

Table 4.12: The example of trajectory query using 4D integrated approach

dimension	minimum value	maximum value
longitude	minimum data	maximum data
latitude	minimum data	maximum data
time	t1	t2

Table 4.13: The example of trajectory query using 3D integrated approach

Trajectory query

The trajectory query is to find the historical position of vessels in a given time range. The necessary data to implement the trajectory query is the time and the MMSI.

- 4D integrated approach
  For the trajectory query, the different thing is the hyper box. To find the position information of a specific vessel S in the give time range (t1, t2), the ranges of data in four dimensions show in table 4.12.
- 3D integrated approach
  For the 3D integrated approach, to make the query much faster, MMSI will use as the first filter. Lots of records will be discarded by applying the MMSI filter. And the following step is as the same as the 4D approach, the ranges of data in three dimensions show in table 4.13

4.4 Benchmark

In order to evaluate the usability and superiority of my method, I will set up a baseline for comparison. I am going to do the bounding box query and the trajectory query just as the same as I have implemented by SFC approach on the plain table which does not have any indexes. There are two aspects that we can compare, one is the query time and the other the storage size of the table that is needed in database.

- Bounding box query
  In order to find the vessels in the space (lon_1 - lon_2 and lat_1 - lat_2) and time range (ts_1 - ts_2). I will use the following statement to query the plain table. And the original_table is the table that has 4 columns (longitude, latitude, time, and MMSI).

  select mmsi from orginal_table where longitude $\geq$ lon_1 and longitude $<$ lon_2 and latitude $\geq$lat_1 and latitude $<$ lat_2 and ts_unix $\geq$ ts_1 and ts_unix $<$ ts_2;
- Trajectory query
  To find the position information in an give time range (ts_1 - ts_2) of a specific vessel (MMIS = S), the following statement will be used:

  select mmsi from orginal_table where mmsi = S and ts_unix $\geq$ ts_1 and ts_unix $<$ ts_2;

5 Analysis and results

Chapter 4 describes the detailed implementation that I have done. And it also gives a presentation of the test plan for the comparison of the different approaches (4D integrated method, 3D integrated method, and 2D integrated method), different SFC (Morton curve and Hilbert curve), and secondary index (BRIN index and B-Tree index). This chapter will give the results of the comparison in detail. Section 5.1 will give the results of the comparison of the index. Then the comparison of the SFC will be shown in section 5.2. The Comparison of the different approach regarding to different SFC method will present finally.

The memory of the CPU and database should be taken into account because it will influence the fetching time directly. Therefore, it is necessary to clear the cache every time. The way I am going to use is to shut down the server every time and clear the cache by using RAMMap [1] at the same time. After a lot of testing, I think this is a credible method.

5.1 4D integrated SFC approach

The 4D integrated SFC approach is the main approach in my research. The data in four dimensions will be encoded to the 4D SFC key together to execute the query. As I mentioned in the previous chapter, the comparison of the Morton curve and Hilbert curve and the comparison of the secondary index will be implemented in the 4D integrated SFC approach by executing the bounding box query. And also, the 4D integrated SFC approach will be compared with another method (3D SFC method) regarding to different query.

5.1.1 Bounding box query

Firstly, the secondary indexes will be compared and then the comparison of the Morton curve and the Hilbert curve will be presented. Finally, the analysis of the 4D integrated SFC approach about the bounding box query will be given. And as I mentioned before, there are two steps of the query, one is the filtering step and the other is refinement step. In order to remove redundant variables to make the comparison clearer. Some comparisons will only be made in the first step. For example, regarding the comparison of indexes, only the index variable is used in the filtering step, and the decoding of SFC is involved in the refinement step. Therefore, the index is only compared in the filtering stage.

Comparison of indexes

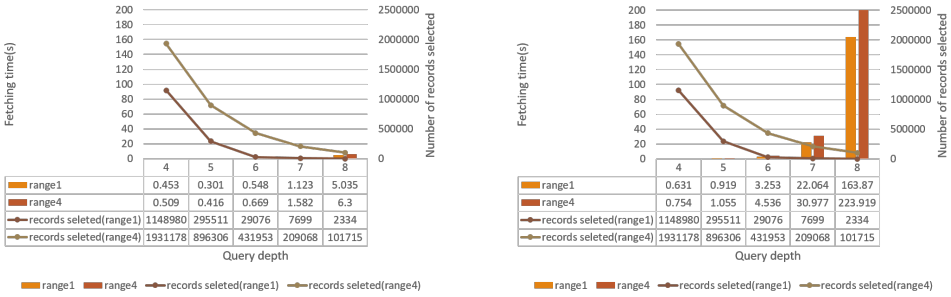
The different secondary indexes which are made on columns influence the efficiency of the query directly. For the BRIN index, an important parameter named 'pages per range' which influence the size of the BRIN index and the query efficiency will be introduced and compared.

For the 4D SFC, the B-Tree index, BRIN index(pages per range =32/64) is made on the SFC key. Here is a table 5.1 describes the time of creating indexes and the size of indexes. It is also worth noting that

[1] RAMMap is an advanced physical memory usage analysis utility for Windows Vista and higher. It is used to manage the memory of the system which can be acknowledged further in this website(`https://docs.microsoft.com/en-us/sysinternals/downloads/rammap`)

index	creating time	size
B-Tree index	11.185s	441MB
BRIN index (pages per range = 32)	2.461S	184KB
BRIN index (pages per range = 64)	2.851S	112KB

Table 5.1: B-Tree index and BRIN index



(a) BRIN index:pages per range = 32

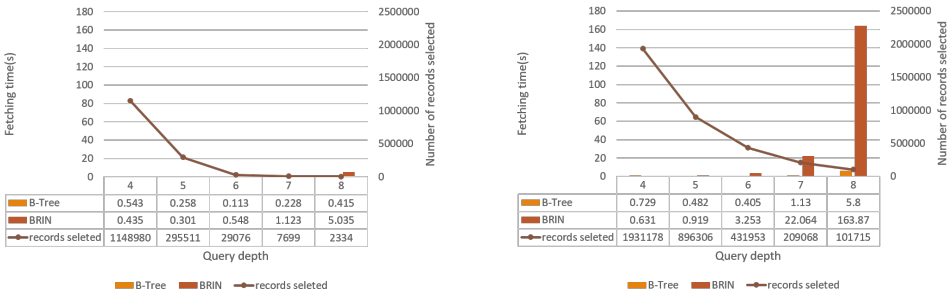
(b) BRIN index:pages per range = 64

Figure 5.1: Querying using BRIN index

before creating the b-tree and brin index, the table I used was sorted. This is why it takes less time to create a BRIN index. The table will be sorted automatically when creating B-Tree index while we need to sort the table ourselves when creating the BRIN index, and the sorting time is about 40s.

Firstly, I should compare the BRIN index with different pages per range in order to select the better one to make the comparison with the B-Tree index. I use the smallest range (range1) and biggest range (range4) to compare the BRIN index with different 'pages per range' in filtering step. The reason why I just compare the index in filtering step is that the decoding of the SFC is included in the refinement step. In order to avoid interference with the decoding process, I compare the index in the first step, that is the filtering step, of the query, which is to find the SFC keys relevant to the query box. The figure 5.1 shows the results, the line chart in the figure shows the number of SFC keys selected after the first step of query.

As we can see from the picture, the BRIN index with a smaller 'pages per range' could support a more efficient query. The principle of the querying using the BRIN index is it divides the data into blocks of equal size and gives the extreme value of each block. When querying, the machine will first query the extremum of each block in order. When the value to be queried is within the extremum of the block, it will access the block and find the value. And when the 'pages per range' increases, the records in each block increase, and the number of the blocks decreases. More records will be accessed to do the same query and the fetching time increases. Although the size of the index would be a little bigger, the efficiency of the query using the BRIN index with a smaller 'pages per range' increases greatly.



(a) query box1

(b) query box4

Figure 5.2: Comparison between B-Tree index and BRIN index

SFC	encoding time	decoding time
Morton curve	7min11s	7min40s
Hilbert curve	22min49s	20min09s

Table 5.2: The time for encoding/decoding SFC

The figure 5.2 shows the comparison between the B-Tree and BRIN index. The figure 5.2a shows the comparison when using a relative small range (range1) and the figure5.2b uses the relative big range (range4). we already know that the storage space required by the BRIN index (184KB) is much smaller than the B-Tree index (441MB). For this figure, we can see that the table using the BRIN index, as the query depth increases, the time required for the query greatly increases. We know that when the query depth increases, the number of SFC keys filtered out will decrease. However, the number of SFC ranges filtered out will increase, that is to say, for this query, the number of queries will increase, so its efficiency will decrease. However, this result is only valid for the filtering stage, because the decoding of SFC key is involved in the refinement stage. The fewer the number of SFC keys, the less the decoding time required. Therefore, it is important to find a balance between query depth and query time, which will also be discussed in the following comparison. If we only focus on comparing the B-Tree index and the BRIN index in the filtering stage, we can say that the BRIN index is a very good choice, its size is very small but can still support efficient queries.

Comparison of SFC

The Morton curve and Hilbert curve will be implemented in the filtering step of the bounding box query. And also, the smallest range (range1) and the biggest range (range2) will be used to test. To use the SFC approach, the encoding and decoding step is necessary. Here is the table 5.2 shows the time of encoding and decoding the 4D SFC (encoding/decoding all data in dataset).

In order to comprehensively compare the Morton curve and Hilbert curve, I will elaborate from the following three aspects: *Locality*, *Compaction*, and *Fetching time*.

- Locality
  The reason why I choose the Morton curve and Hilbert curve is that both curves have the great locality attribution which preserve the positions of the point in reality. And the this attribution could help increase the efficiency of the query at the hardware level. As I mentioned before, the SFC ranges will be obtained after filtering step. And the number of the ranges could convey an important information, that is the locality of the curve. For the same query range, if the number of SFC ranges is large, it reflects that these SFC keys are not compact, and it also indicates the bad locality. Also, the number of the ranges has the relationship with the query depth. Hence, the table 5.3 shows the number of the ranges I get after first filtering step by using the query box1 and query box4 with different query depth. From the table 5.3 we can find that the number of Morton ranges and Hilbert ranges we obtain is equal when facing different query depth. And the reason is that the storage depth of our dataset is quite big (storage depth = 30). The query depth we use is from 4 to 7 which is much smaller than the storage depth which led to the entire SFC keys being selected (the explanation is shown in figure 3.12 and 3.13). In this way, the locality of morton curve and hilbert curve cannot be analyzed. Hence, the analysis of the final results by using these two curves is needed.

  To get the final results, we need to do not only filtering step but also the refinement step. Because the final results will be the same when using the same query box to do the query. Hence, I use the query box1 and query box4 to do the bounding box query (with query depth = 5). And the SFC keys will be obtained after the query. I find that there is no continuous SFC keys, which shows that there are too few vessels in this range. However, from the range of the obtained SFC keys, it can reflect a little difference in the locality of these two curves. The table 5.4 and 5.5 show the range of the SFC keys we get after query. And we can find that the Hilbert curve has the better locality.

query box	query depth	Morton ranges	Hilbert ranges
query box1	4	28	28
	5	108	108
	6	318	318
	7	848	848
query box4	4	84	84
	5	432	432
	6	2597	2597
	7	17914	17914

Table 5.3: The number of SFC ranges after filtering step

curve	(max-min) value
Morton curve	12116959349677267343076078363237l2
Hilbert curve	28922156432209559939459520298l386

Table 5.4: The range of SFC keys after query (query box1)

- Fetching time
  The fetching time is the crucial indicator for the query. The efficient query also means the fast query. And as I mentioned before, the query depth effects the efficiency of the query, to find the balance between query depth and the fetching time is what I have done. To compare the fetching time of the bounding box query by using Morton key/Hilbert key, similarly, the query box1 and the query box4 will be used. The figure 5.3 shows the results. And we can see from the figure that the fetching time is much smaller when using Morton curve.

  And as I mentioned in subsection4.3.3, there are two main steps in query. One is the filtering step and the other is the refinement step, and the range selecting and table joining are in the filtering step. To find out why the fetching time is quite large when using Hilbert curve, the time required for each step in the query needs to be clear. The figure 5.4 and 5.5 shows the time required for each step in the query. Similarly, I still use query box1 and query box4 for the query.

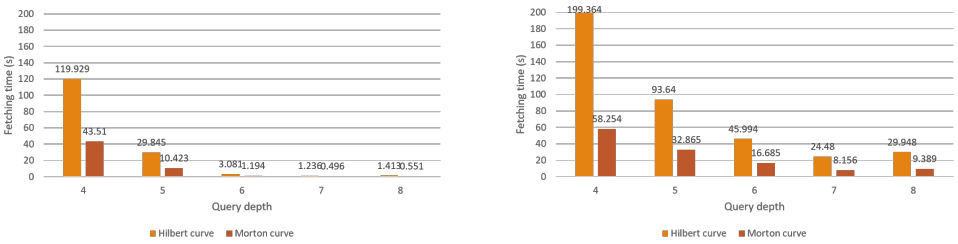
  The Hilbert curve is as good as the Morton curve when facing the filtering step. While the fetching time increases greatly when the refinement step is done by using the Hilbert curve. Through my analysis, I think the reason for the increase in fetching time is mainly because a lot of time is spent decoding SFC key. As we mentioned before, compared to the Morton curve, the encoding and decoding of the Hilbert curve are more complicated, because the Hilbert curve is obtained by a series of mirroring and rotation. Due to a large amount of time spent in the decoding process, the overall query efficiency is reduced. We can think that the Hilbert curve is not suitable for the query we used.

Query results

Finally, I am going to use the Morton curve and B-Tree index to do the bounding box query. And I use 4 query ranges that in different sizes to do the bounding box query to test the 4D integrated method. And for the bounding box query, we want to know which vessels are in the given space and time range. Hence, the whole range of MMSI should be put in the hyper box which is not very efficient. The figures 5.6 shows the fetching time that the query required. The rate of the redundant point represents the

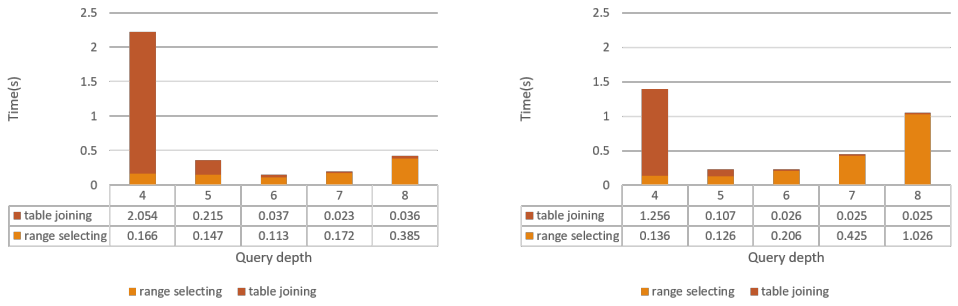
curve	(max-min) value
Morton curve	6633516056022991677869415806908476l4
Hilbert curve	2099413839166022087688401964794l8653

Table 5.5: The range of SFC keys after query (query box4)



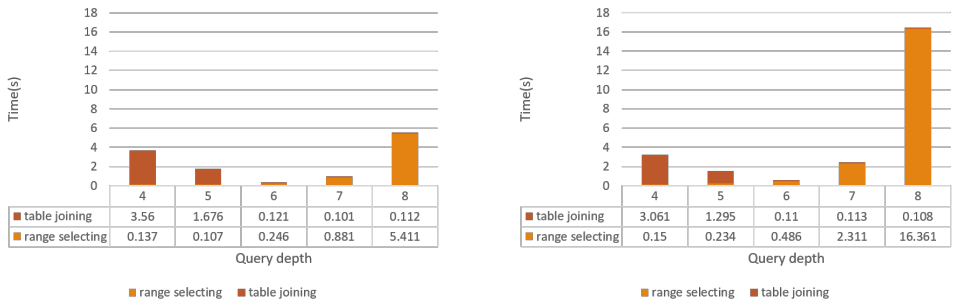
(a) querybox1 (b) querybox4

Figure 5.3: Query using Morton curve and Hilbert curve



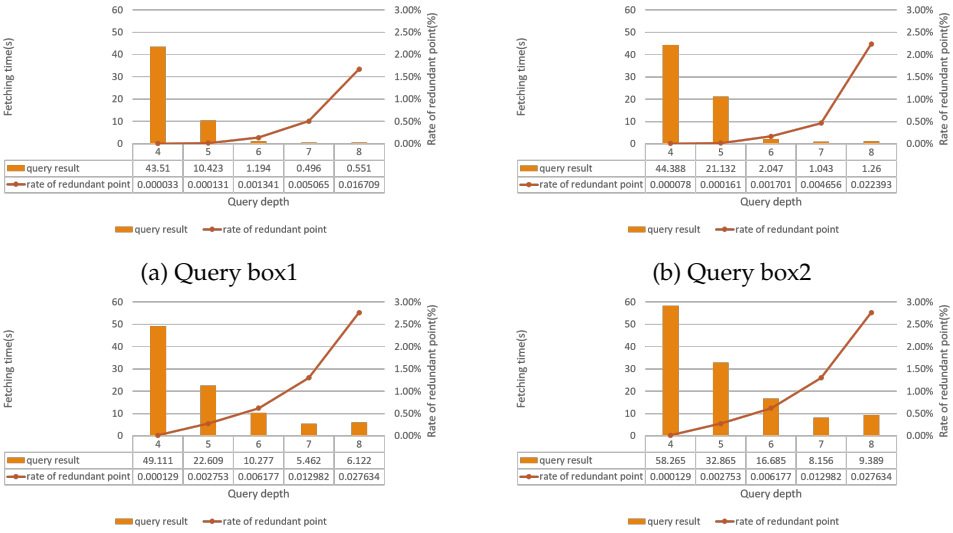
(a) Using Morton curve (b) Using Hilbert curve

Figure 5.4: Time needed for each step in query (using query box1)



(a) Using Morton curve (b) Using Hilbert curve

Figure 5.5: Time needed for each step in query (using query box4)



(a) Query box1

(b) Query box2

(c) Query box3

(d) Query box4

Figure 5.6: Bounding box query using 4D integrated approach

proportion of the SFC keys obtained in the filtering phase in the final selected SFC key. The formula to compute the rate of the redundant point is shows below. At the same time, it can reflect the proportion of SFC keys selected in the filtering stage for different query depths.

$$\textit{rate of redundant point} = (\textit{SFCkeys(final result)}) \div (\textit{SFCkeys(filtering step)}) \quad .$$

We can see that the query depth plays an important role in querying and it influences the efficiency of the query greatly. From the test I do, the query is most efficient when the query depth is 7. When the query depth is not enough large, lots of the SFC key will be selected after filtering, So a lot of time is spent on refinement due to the decoding. While if the query depth is quite large, the time of generating SFC ranges in filtering step will increase. And the rate of the redundant point is also very important. As we can see in the figure 5.6, the rate of the redundant point is quite low. While, if the rate increases, the efficiency of the query won't keep increasing. Therefore, for different queries, the best proportion of redundant points will also change, but eventually it will reach a balance at a certain point in the query.

5.1.2 Trajectory query

The trajectory query is to find the position information of a specific vessel. For the trajectory query, the data needed is MMSI and time range. If we use the 4D integrated method, the 4D hyper box should be given. While there are data in two dimensions (longitude and latitude) that we cannot give the range. Hence, the whole range of the data should be used to compute the query range. Obviously, the hyper box will be super large and the efficiency might be influenced. Hence, I will give a small time range (1 hour) to test if the 4D integrated SFC approach can be used to do the trajectory query. The following test is to find the historical positions of a specified vessel whose MMSI is 244670079 in 1 hour (from 2016-12-09 23:59:59.865 UTC - 2016-12-10 0:59:59.865 UTC). And the figure 5.7 shows the results.

As we can see, even I only ask for the positions of a vessel in 1 hour, the query is not very efficient. To figure out which step takes too much step, I record the time for generating the SFC ranges and joining the table which is to obtain the SFC keys. The figure 5.7 shows the result. And as we can see that, these two steps does not take too much time. And we can find that too many SFC keys which is because of the query box which is used for the trajectory query. To use the 4D integrated approach to do the trajectory query, the whole range of longitude and latitude should be considered in generating the 4D query box. Lots of redundant SFC keys will be selected. And the decoding of the SFC keys in refinement

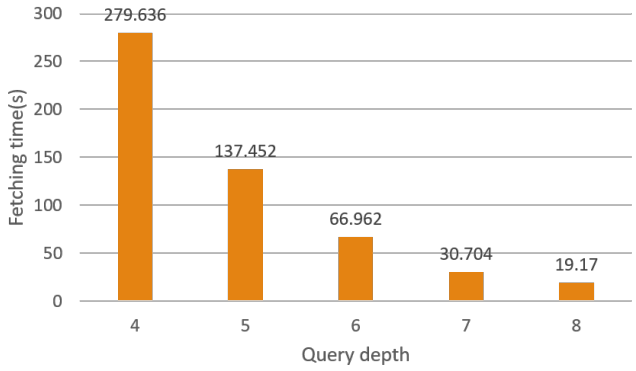


Figure 5.7: Trajectory query using 4D integrated approach

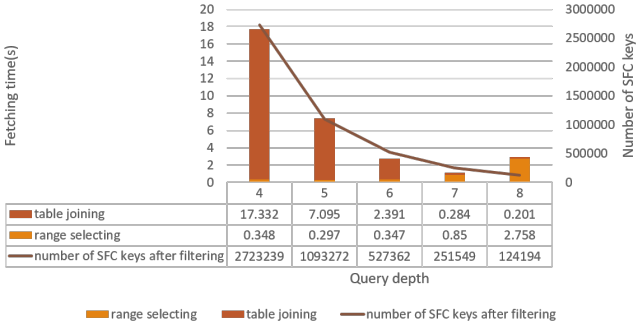


Figure 5.8: The time used in each step

step takes lots of time which makes the query inefficient. Hence, I think the 4D integrated method does not suitable for trajectory query.

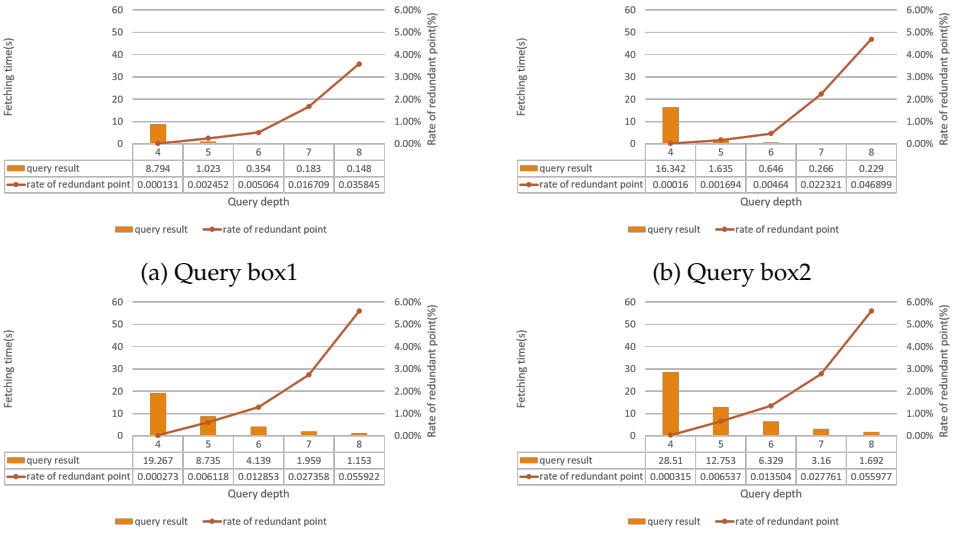
5.2 3D integrated SFC approach

The 3D integrated SFC approach means that the data in three dimensions (longitude, latitude, time) will be encoded to the 3D SFC key to execute the query. The SFC that I am going to use is the Morton curve and the index I will use is the B-Tree index. And for the query range, I will use the same range as I use in the 4D integrated approach in order to make a reliable comparison.

5.2.1 Bounding box query

For the bounding box query, I use the same query range in 4D integrated approach to test. The MMSI do not have to make the query hyperbox. And the efficiency of the query might be higher. And the results shows in the figure 5.9:

It is not difficult to find that the efficiency of the query is very high whether it is aimed at a large range or a small range. I think this is because, in the 3D integrated method, we only use longitude, latitude, and time to determine the 3D hyper box. We no longer add the entire MMSI to it. This greatly reduces the SFC keys we get in the filtering step, and the time for decoding SFC keys in refinement step reduces. That is why query is more efficient. And to proof my idea, here is the table 5.6 and 5.7 shows the number of SFC keys obtained after filtering step. Hence, if we can choose a suitable query box which means that the data in each dimension should to be meaningful, the efficiency of the query will increase. For example, for the bounding box query, we want to query which vessels are in the given space and



(a) Query box1

(b) Query box2

(c) Query box3

(d) Query box4

Figure 5.9: Bounding box query using 3D integrated approach

query depth	4D integrated SFC approach	3D integrated SFC approach
4	1148980	297461
5	295511	29077
6	29076	7700
7	7699	2334
8	2334	1088

Table 5.6: The number of SFC keys after filtering (query box1)

time range. If we use the 4D SFC approach, we have to make the 4D query box to do the query. The MMSI here is meaningless because we have put the whole range of the MMSI into query box. On the contrary, only the 3D query box is used when using 3D SFC approach to do the bounding box query. The meaningless data (MMSI) which is used for generating query box can be discarded. Hence, there are fewer redundant SFC keys, thus, the query become efficient.

5.2.2 Trajectory query

For the trajectory query using 3D integrated SFC approach. The first I am going to do is to update the table by filtering the table using MMSI. Then the records in the table will decrease sharply. In the following test, only 3187 records left in 11329211 records after filtering by using 'select morton_key from table where mmsi = 244670079'. Then the B-Tree index is created on the morton_key in the new table. I still use the same time range as I use in 4D integrated approach (1 hour). The result show in figure 5.10. The efficiency of the trajectory increases compared with using 4D integrated approach.

query depth	4D integrated SFC approach	3D integrated SFC approach
4	1931178	903096
5	896306	435338
6	431965	210739
7	209068	102517
8	101715	50842

Table 5.7: The number of SFC keys after filtering (query box4)

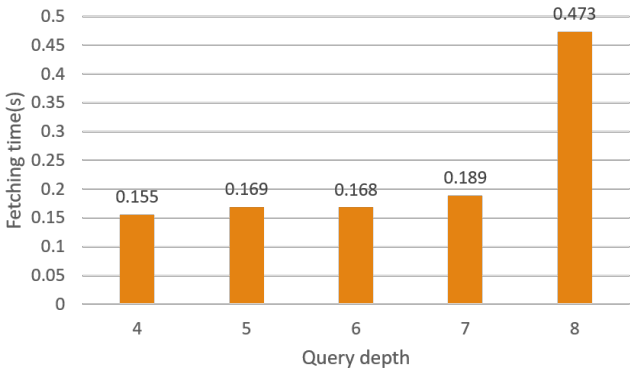


Figure 5.10: Trajectory query using 3D integrated approach

query box	query time (s)	number of records(final result)
query box1	3.286	39
query box2	3.279	90
query box3	3.339	1787
query box4	3.608	2846

Table 5.8: Query time using plain table in database (bounding box query)

Obviously, the efficiency of the trajectory query is high, it is a good way to do the trajectory query. The reason why the efficiency increases is because the total number of the SFC keys reduces by filtering the MMSI. It is undeniable that this is indeed a way to improve efficiency of the trajectory query. But what I want to say is that if the data set after filtering is still very large, this method can only relatively improve the efficiency of the trajectory query (compared with using the original data set). Using the full range of longitude and latitude to generate the query box is still not very good.

5.3 Benchmark and comparison

In order to explain the superiority of the SFC approach I proposed. I set the plain table as the baseline to do the comparison. As I mentioned in section 4.4, I will compare the query time and the storage size of the table in PostgreSQL. The query box I use is as the sam as I used in SFC approach, and the four query box will be tested.

For the bounding box query, the result shows in table 5.8. And the comparison will be done between the 3D SFC approach (better than 4D SFC approach) and the plain table, the result shows in figure 5.11. We can clearly see from the figure that the query time is shorter when using 3D SFC approach. For the trajectory query, the result shows in table 5.9. In addition, using these two methods, the final result I get is the same, it can also be said that the SFC approach is feasible. Obviously, the SFC approach performs better.

As I wrote before, the SFC key I used in thesis is the full resolution SFC key and that means that the

query approach	query time (s)
3D SFC approach	0.155
plain table	3.560

Table 5.9: Query time using plain table in database (trajectory query)

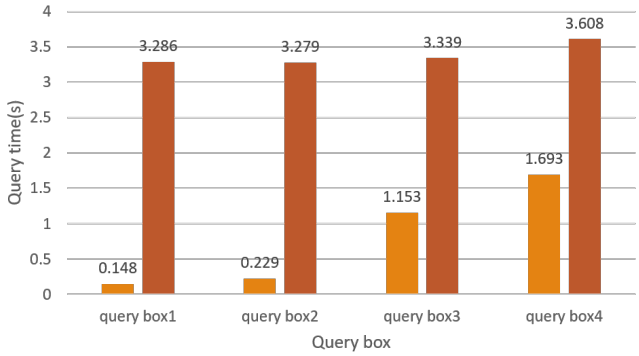


Figure 5.11: The comparison between 3D approach and plain table using bounding box query

query approach	table size
3D SFC approach	567MB
plain table	1724MB

Table 5.10: Table size in databse

data which is encoded in the SFC keys can be discarded. For the 3D integrated method, there are only two columns kept in database (mmsi, 3D SFC key). And for the plain table, all the columns should be stored in database. The storage size of the table is shown in table 5.10. In summary, the SFC approach is a great approach that deal with the 4D AIS data.

6 Conclusion

This thesis has researched whether the Space filling curve can be used to manage the 4D AIS data in database to support the efficient query. Concluding will be done by answering the research questions that I proposed firstly and the pros and cons of the approach will subsequently be discussed. Finally, the ideas about future work based on my research will be mentioned.

6.1 Research overview

For now, the proposed methodology has been tested by using the bounding box query and trajectory box with different size of query box. It is nesessary to review the research questioned that I mentioned at the beginning. One main question and four sub-questions will be answered one by one. And the main question is:

How to efficiently manage 4D AIS data (Longitude, Latitude, Time, MMSI) of vessels to do the fast query by using Space Filling Curve in PostgreSQL?

For efficiently manage 4D AIS data, clustering and indexing is significant. Firstly, the type of the space filling curve is crucial. The Morton curve and Hilbert curve are chosen because of the locality and recursive attribution of these two curves. Secondly, how to organize the 4D data is important. I used the 4D integrated approach which encoding the data in four dimensions to the SFC keys. And the 3D integrated approach which put the longitude, latitude, and time together to compute the SFC keys. Then, the indexing technology is also necessary, B-Tree index and BRIN index are both great for our methodology. Therefore, selecting the proper curve (Morton key/Hilbert key) and the clustering and indexing method (B-Tree index and BRIN index) are important for doing the fast query. And the sub-questions will be answered following:

- ***How to manage the 4D AIS data to support the efficient query? Using 4D integrated approach or 3D integrated approach?***

  For how to manage the four-dimensional AIS data, there are several ways to manage 4D AIS data, such as regarding the 4D AIS data as a whole, just like the 4D integrated method I used in the thesis. Or we can think of the 3D data as a whole, or 2D data as a whole. For the 3D intergated method or the 2D integrated method, which of the 4D data is considered as a whole still need to think. Taking the data I used in the thesis (MMSI, longitude, latitude, time) as the example, I integrated longitude, latitude, and time when using the 3D integrated method and kept MMSI as the separate column. Because whether it is for the bounding box query or the trajectory query I want to do, MMSI is relatively independent. For example, in the bounding box query, I want to know which vessels are within the specific query range. MMSI has nothing to do with the query conditions. And in trajectory query, I can use MMSI to filter the data set first to improve query efficiency. I think which data can be taken as a whole depends on the type of query. For different queries, we can combine different data into a whole.

  In my thesis, I used two kinds of approach to manage the 4D AIS data, one is the 4D integrated SFC approach and the other is the 3D integrated SFC approach.

In order to determine which method I proposed can manage 4D data well, I used two queries to test, one is bounding box query and the other is trajectory query. Judging from the fetching time the query needs, it is better to use 3D integrated approach. Of course, this result is not absolute. It depends on the specific method I use. For example, when using the 3D integrated approach for trajectory query, I first filter the entire table through MMSI, the query efficiency increases because the total data volume is greatly reduced. But if I use the 3d SFC key to query first and then filter the MMSI, the efficiency of the entire query will be affected. Therefore, aiming at the method I use, the 3D integrated method performs better.

- ***How to scale data in each dimension properly to compute the SFC key?***
  I converted it to an integer based on retaining all its information, which is also what must be done to calculate the SFC key. Secondly, to make the query selective and more efficient, what I have to do is to ensure that the data size of each dimension has roughly equal bit length. The SFC key calculated in this way can make the query more efficient.

  However, If the dimensions of data increase, the full resolution key would be super large and it will be hard to store. And as we mentioned before, in order to make the data support the efficient query, the data in each dimension should in roughly equal bit length which may increase the value of the SFC key. And the storage size will increase as well. Therefore, if the data in each dimension is too large, scaling the data to a small range is also a good choice.

- ***Which Space Filling Curve performs better?***
  Both the Morton curve and Hilbert curve can be used to manage the 4D AIS data. I have compared these two curves in two aspects. One is the fetching time of the bounding box query when by using Morton curve and Hilbert curve, the other is the locality of the curve which can preserve the positions of the points. From the first aspect, Morton curve performs better. This is because the decoding of Hilbert keys spends lots of time. For the second aspect, I would say that the Hilbert curve performs better.

- ***How about the BRIN index and B-Tree index?***
  The BRIN index is great due to the small size of itself and it also can support the efficient query. We can see that the creating B-Tree index and BRIN index take almost the same time because I ordered the Morton/Hilbert key first which cost around 40s. If the Morton key/Hilbert key is unordered, B-Tree index is created faster than BRIN index. The very small storage space required by the brin index is enough to show that it is a good index and is very suitable for assisting the SFC method we use.

6.2 Future work

- The data may not only be limited to AIS data or the data I use. There is a lot of useful information in the AIS data, which can be used as research objects.
- The geometry of the query range could be changed in order to be suitable for the different kinds of query. What I used is the hyper box, query boxes of other shapes can also be studied, for example, the circle, the triangle and so on.

Bibliography

Augustine, J. (2019). Brin index for postgresql: Don't forget the benefits.

Benetis, R., Jensen, C. S., Karciauskas, G., and Saltenis, S. (2002). Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Proceedings International Database Engineering and Applications Symposium*, pages 44–53. IEEE.

Brisaboa, N. R., Luaces, M. R., Navarro, G., and Seco, D. (2009). A new point access method based on wavelet trees. In Heuser, C. A. and Pernul, G., editors, *Advances in Conceptual Modeling - Challenging Perspectives*, pages 297–306, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bye, R. J. and Aalberg, A. L. (2018). Maritime navigation accidents and risk indicators: An exploratory statistical analysis using ais data and accident reports. *Reliability Engineering & System Safety*, 176:174–186.

Cummings, M. L., Buchin, M., Carrigan, G., and Donmez, B. (2010). Supporting intelligent and trustworthy maritime path planning decisions. *International journal of human-computer studies*, 68(10):616–626.

Dai, H.-K. and Su, H.-C. (2003). On the locality properties of space-filling curves. In *International Symposium on Algorithms and Computation*, pages 385–394. Springer.

database kernel group, C. R. (2015). 9.5 new features - brin index. `http://mysql.taobao.org/monthly/2015/05/05/`.

De Almeida, V. T. and Güting, R. H. (2005). Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33–60.

De Vreede, I. (2016). Managing historic automatic identification system data by using a proper database management system structure.

Erwig, M., Güting, R. H., Schneider, M., and Vazirgiannis, M. (1998). Abstract and discrete modeling of spatio-temporal data types. In *Proceedings of the 6th ACM international symposium on Advances in geographic information systems*, pages 131–136.

Feng, M., Shaw, S.-L., Fang, Z., and Cheng, H. (2019). Relative space-based gis data model to analyze the group dynamics of moving objects. *ISPRS Journal of Photogrammetry and Remote Sensing*, 153:74–95.

Forlizzi, L., Güting, R. H., Nardelli, E., and Schneider, M. (2000). A data model and data structures for moving objects databases. *ACM SIGMOD Record*, 29(2):319–330.

Fox, A., Eichelberger, C., Hughes, J., and Lyon, S. (2013). Spatio-temporal indexing in non-relational distributed databases. In *2013 IEEE International Conference on Big Data*, pages 291–299. IEEE.

Frentzos, E. (2008). Trajectory data management in moving object databases. *PhD Book, Department of Informatics, University of Piraeus*, 10.

Gargantini, I. (1982). An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910.

Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems (TODS)*, 25(1):1–42.

Guttman, A. (1997). A dynamic index structure for spatial searching. In *Proceedings of the 13th ACM SIGMOD International Conference on Management of Data*, pages 47–57.

Hexeberg, S., Flåten, A. L., Eriksen, B. H., and Brekke, E. F. (2017). Ais-based vessel trajectory prediction. In *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8.

Hornsby, K. and Egenhofer, M. J. (2002). Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):177–194.

Jensen, C. S., Lin, D., and Ooi, B. C. (2004). Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 768–779. VLDB Endowment.

Kothuri, R. K. V., Ravada, S., and Abugov, D. (2002). Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557.

Lechtenberg, S., Braga, D., and Hellingrath, B. (2019). Automatic identification system (ais) data based ship-supply forecasting.

Li, d. (2017). Research on dangerous encounter hotspots mining based on ais big data. Master's thesis, South China University of Technology.

Liu, t. (2017). Research on storage and indexing technology of ais data in cloud environment. Master's thesis, Wuhan University of Technology.

Mao, S., Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G.-B. (2018). An automatic identification system (ais) database for maritime trajectory prediction and data mining. In *Proceedings of ELM-2016*, pages 241–257. Springer.

Meijers, M. and van Oosterom, P. (2018). Clustering and indexing historic vessel movement data with space filling curves. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(4).

Meijers, M., van Oosterom, P., and Quak, W. (2016). Management of ais messages in a geo-dbms. Technical report, Technical report, Delft University of Technology.

Mokbel, M. F., Aref, W. G., and Kamel, I. (2003). Analysis of multi-dimensional space-filling curves. *GeoInformatica*, 7(3):179–209.

Mokbel, M. F., Xiong, X., and Aref, W. G. (2004). Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 623–634.

Moniruzzaman, A. and Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*.

PAN, M., GAO, L., SONG, P., and ZHAO, L. (2016). Mongodb based ship database construction and batch migration technology of its data. *Journal of Dalian Maritime University*, pages 39–44.

Praing, R. and Schneider, M. (2007). Modeling historical and future movements of spatio-temporal objects in moving objects databases. pages 183–192.

Psomadaki, S. (2016). Using a space filling curve for the management of dynamic point cloud data in a relational dbms.

Rajabi, A., Saryazdi, A. K., Belfkih, A., and Duvallet, C. (2018). Towards smart port: An application of ais data. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1414–1421. IEEE.

Reiss, F., Stockinger, K., Wu, K., Shoshani, A., and Hellerstein, J. M. (2007). Enabling real-time querying of live and historical stream data. In *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, pages 28–28. IEEE.

Ristic, B., La Scala, B., Morelande, M., and Gordon, N. (2008). Statistical analysis of motion patterns in ais data: Anomaly detection and motion prediction. In *2008 11th International Conference on Information Fusion*, pages 1–7.

Robards, M., Silber, G., Adams, J., Arroyo, J., Lorenzini, D., Schwehr, K., and Amos, J. (2016). Conservation science and policy applications of the marine vessel automatic identification system (ais)—a review. *Bulletin of Marine Science*, 92(1):75–103.

Shih, T. K. and Wang, P. P. (2004). *Intelligent Virtual World: Technologies and Applications in Distributed Virtual Environment*. World Scientific.

Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S. (1997). Modeling and querying moving objects. In *Proceedings 13th International Conference on Data Engineering*, pages 422–432. IEEE.

van Oosterom, P. (1999). Spatial access methods. *Geographical Information Systems Principles, Technical Issues, Management Issues, and Applications*, 1:385–400.

van Oosterom, P., MEIJERS, M., VERBREE, E., LIU, H., and TIJSSEN, T. (2018). Towards a relational database space filling curve (sfc) interface specification for managing nd-pointclouds.

Volker, g. and oliver, g. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2).

Wang, w., Chu, x., Jiang, z., and Liu, l. (2019). analysis of traffic saturation of bridge waters based on historical ais data. *Journal of Transport Information and Safety*.

Xu, h. (2010). *Research of High-Dimensional Space Query Algorithm Based on Space Filling Curves*. PhD thesis, Harbin University of Science and Technology.

Zissis, D., Xidias, E. K., and Lekkas, D. (2016). Real-time vessel behavior prediction. *Evolving Systems*, 7(1):29–40.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.