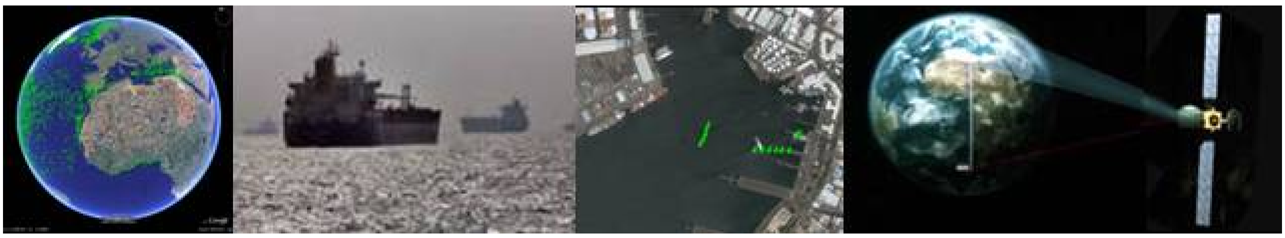
in cooperation with

Institute of
Space Systems

Flight
Operations



Technical Note TN 2: Description of AIS project database

Preparatory Action for Assessment of the Capacity of Spaceborne Automatic Identification System Receivers to Support EU Maritime Policy

DG MARE Service Contract : No MARE/2008/06/ SI2.517298

Doc.Type: Technical Note Description of AIS project database			DRD N°: TN
Doc. N°:	Issue:	Date:	Page 1 Of 15
Title :			

	Name & Function	Signature	Date
Prepared by:	Tommy Vibe Raunstrup		
Approved by:			
Application authorized by:			
Customer / Higher Level Contractor			
Accepted by:			
Approved by:			

DISTRIBUTION LIST	N	A	I
Consortium Internal To all partners			
External (DG MARE) Iain Shepherd (DG MARE)			
N=Number of copy A=Application I=Information			

Data Management:	File: 6030_PASTA_MARE_Technical_Note_Template.doc
Signature Date	

DOCUMENT CHANGE RECORD			
ISSUE	*DATE*	*CHANGE AUTHORITY*	*REASON FOR CHANGE AND AFFECTED SECTIONS*
1	15/06/09	FDC	Initial version, reviewed by FDC
2	15/07/09	FDC	Comments from FCD included

TABLE OF CONTENT

1.1 PASTA MARE PROJECT

PASTA MARE project aims at increase the understanding of the potential performance of space-based AIS and its potential effectiveness as a tool to support the maritime policy of EU as well as to improve efficiency of governmental and commercial activities in the maritime sector.

To reach this objective the following main tasks have been established:

- Establishment of an improved understanding of global transport in terms of vessel traffic density information as input and benchmarking of the spaceborne and airborne AIS data performance tests
- Identification of legal and illegal sources of interference of the AIS signal
- Testing of space based AIS payloads already in orbit in terms of their performance versus other available vessel data (LRIT, ground based AIS)
- Development of a most advanced and innovative AIS receiver payload system and to test it through extended airborne campaigns and cross check results with ground and space based AIS as well as LRIT data
- Drawing up of conclusions of the different developments and tests in terms of lessons learned and provide recommendations for possible next steps towards a spaceborne AIS service

2. DOCUMENT OBJECTIVES

To better understand what space AIS is able to provide to Maritime Stakholders, the methodology adopted in PASTA MARE projet is compare its outputs to available data provided today by "ground" AIS and LRIT.

Beside the collection of data itself, the project needs an advanced tool to support comparison and analysis. Mainly this tool have to:

- read, format, and save data,
- provide statistics on defined periods and areas,
- provide filters on data sources, periods, type of vessels,
- provide means to output data on GIS

The objective of this document is to present and describres tools which will be used to fulfil project requirements.

3. APPLICABLE AND REFERENCE DOCUMENTS

Ref. 1	Preparatory Action for Assessment of the Capacity of Spaceborne Automatic Identification System Receivers to Support EU Maritime Policy – Pasta Mare Technical Proposal Call for Tenders No MARE/2008/06

4. DATABASE

The Gatehouse AIS database solution provides record, replay and statistics functionality. The solution is based on the Oracle DBMS (Oracle Enterprise Edition), and can be accessed using ODBC or Oracle native drivers.

4.1.1 DBLSS RECORD

The database LSS (DBLSS) is a stand-alone module that can be added to an existing AIS system. The only input needed by the DBLSS is an ITU-R M.1371 standard format compliant data stream.

The DBLSS is the foundation of the database features:

- Record & Replay.
- Statistics and reports.

The database is divided into two sub-databases with each its function:

- A statistical database used for analytic presentation of AIS data.
- A real time database containing all NMEA sentences received from the AIS network. This makes it possible to efficiently replay AIS data in the Subscriber Proxy application.

The architecture of the database is illustrated in Figure 1.

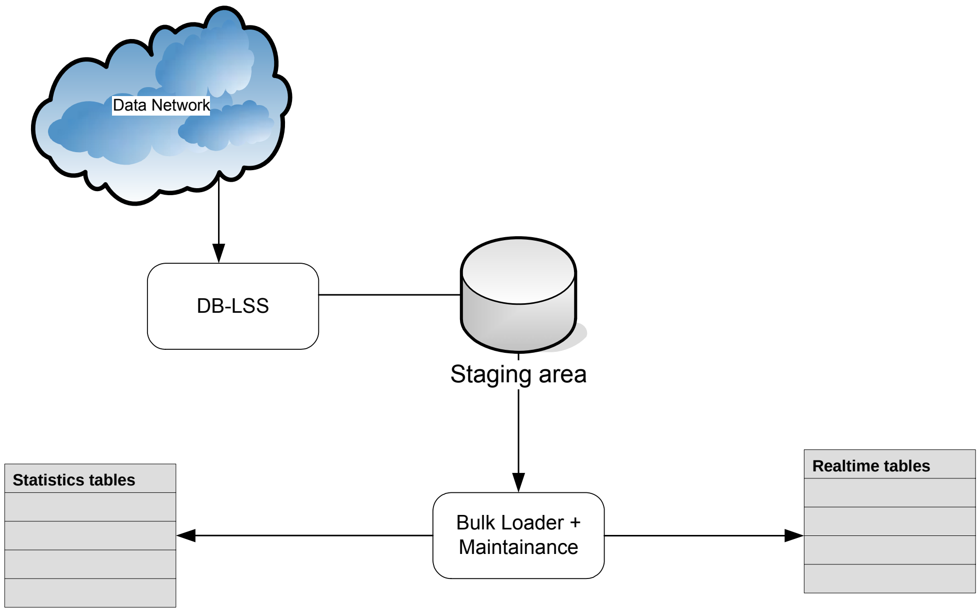


Figure 1: Architecture overview of the database.

All messages recorded in the database are automatically stamped with an UTC timestamp. A GPS time source can be used if necessary.

4.1.2 TAGGING OF DATA SOURCE

All data stored in the database is tagged with its data source ID and its data source type (e.g. AIS, LRIT, RADAR, AIS-Satelitte, etc). This information is determined by the DBLSS based on the format of the data received. E.g. meta data encapsulated in PGHP messages, XML. This allows filtering data based on data source when generating statistics against the database.

4.1.3 DATABASE MODEL

The database model will not be described in detail, however the main tables of the database model is described below.

TBL_LOG:

This is a special purpose table meant for storing all AIS messages received by the DBLSS, in their raw format. Each received NMEA sentence is stored in its raw format, along with information about the sentence type, the time the message was received, and the time it was inserted into the database.

This table is used for replaying data, and for providing storage of all AIS messages at the finest granularity.

If a message holds location data (e.g. message types 1, 2, 3, 4, 18, 19) it is decoded. This allows for replaying only data from inside a given geographical area.

The table holds the fields dimension_id1, dimension_id2, which refers to the IDs of the decoded NMEA sentence in one of the other tables – e.g. if a given row containing AIS data for a static and voyage report, the dimension_id1 would refer to the ID of the decoded message in tbl_entity_stat.

This table provides the opportunity of extending the system with new statistics, as all AIS data is stored here, and can be decoded into any appropriate database structure.

The following data is stored in the table

- Unique ID identifying the row
- Data source id
- Data source type
- Target AIS class if applicable (CLASS-A,CLASS-B,SAR, etc)
- Timestamp for when the report was received
- Timestamp for when the report was inserted into the database
- Sequence number from AIS message (if applicable)
- Repeat indicator from AIS message (if applicable)
- Nmea type
- Raw NMEA string
- Latitude of position report (if applicable)
- Longitude of position report (if applicable)
- Navigational status from AIS message (if applicable)
- Rate of turn (if applicable)
- Course over ground (if applicable)

- Speed over ground (if applicable)
- Altitude (if applicable)
- Heading (if applicable)
- Regional applications field from AIS message (if applicable)
- RAIM from AIS message (if applicable)
- Positional accuracy from AIS message (if applicable)
- Communication state from AIS message (if applicable)
- Foreign key to tbl_entity_stat
- Foreign key to tbl_entity_voy

TBL_ENTITY_STAT:

This table holds the static part of message type 5/19, which is referred to in the TBL_LOG and TBL_ENTITY_POS tables. If a ship keeps sending a message with the same static information, it is not inserted into this table multiple times.

The following data is stored in the table

- Unique ID identifying the row
- Data source type
- Target AIS class if applicable (CLASS-A,CLASS-B,SAR, etc)
- Timestamp for then the report was inserted into the database
- MMSI
- IMO (IMO ship identification number of the ship being tracked)
- Callsign
- Vessel name
- Type of ship and cargo field from AIS message (if applicable)
- Size a,b,c,d fields from AIS message (if applicable)
- Type of positional device from AIS message (if applicable)
- Size a,b,c,d fields from AIS message (if applicable)
- Latitude for the report
- Longitude for the report

TBL_ENTITY_VOY:

This table holds the voyage related part of message type 5/19. It is inserted into the database in the same manner as the static part (TBL_ENTITY_STAT) – i.e. a message is only inserted once, until it changes.

The following data is stored in the table

- Unique ID identifying the row

- Data source type
- Target AIS class if applicable (CLASS-A,CLASS-B,SAR, etc)
- Timestamp for then the report was inserted into the database
- Draught
- Destination
- Type of cargo field from AIS message (if applicable)
- ETA field from AIS message (if applicable)
- MMSI
- Latitude for the report
- Longitude for the report

TBL_ENTITY_POS:

This table holds decoded position reports, and is the main table used for position related statistics. References are kept to tbl_entity_stat and tbl_entity_voy, thereby identifying the static and voyage related information about the ship at the time of sending the position report.

The following data is stored in the table

- Unique ID identifying the row
- Data source type
- Target AIS class if applicable (CLASS-A,CLASS-B,SAR, etc)
- Timestamp for then the report was inserted into the database
- Timestamp for when the previous report from the vessel was inserted into the database
- MMSI
- Latitude for the report
- Longitude for the report
- Latitude for the previous report from the vessel inserted into the database
- Longitude for the previous report from the vessel inserted into the database
- Rate of turn
- Course over ground
- Speed over ground
- Heading
- Altitude (if applicable)
- Navigational status from AIS message (if applicable)
- Foreign key to tbl_entity_stat

- Foreign key to tbl_entity_voy

4.2 DATABASE PERFORMANCE

In the following it is described which steps are taken in the database architecture to ensure that scalability is ensured when the number of rows in the database grows.

These are:

- Bulk loading + index maintenance
- Downsampling of data
- Spatial Temporal Indexing
- Data partitioning

4.2.1 LOADING PROCEDURE OF DATA

The DBLSS process has the responsibility of converting the raw AIS data to a format which allows it to be loaded efficiently into the statistics database. The processed data is stored temporarily in a staging area. Periodically, the DBLSS starts a bulk loader process, which is responsible for moving the data from the staging area into the database. This architecture has been adopted in order to:

- Ensure optimal query performance
- Ensure efficient insertion of data into the database
- Avoid index fragmentation
- Allow for maintaining the database periodically

Data is continuously accumulated at the staging area, where it is processed and transformed into the statistics database schema structure. In the staging area the AIS data is decoded. At fixed intervals (e.g. 12:00 every night), data in the staging area is inserted into the database.

Before inserting data into the database, index updates are disabled. After the bulk loading has been completed, the indices are rebuilt. This is done in order to avoid index fragmentation and to ensure efficient insertion of data, since it is much more efficient to rebuild an index, than continuously maintaining it, when inserting millions of rows.

Data in the database can be queried and extracted whilst new data is loaded into the database, with the exception of data for the time-period that the bulk load is updating.

Furthermore, the bulk loader organizes the insertion of data into one time of the day, where it is unlikely that queries will be performed. This prevents insertion of data from reducing the performance of queries and vice versa.

4.2.2 DOWN-SAMPLING OF DATA

The DBLSS can be configured to down-sample data before it is inserted into the database. Figure 2 illustrates how down-sampling can be configured using the ASM tool.

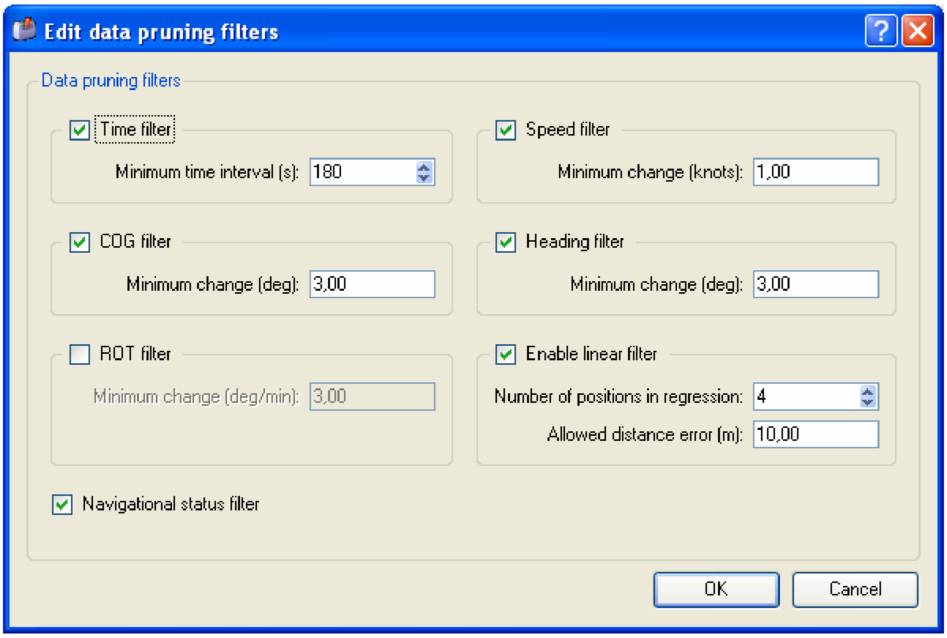


Figure 2: Configuring downsampling of data in the DBLSS-Record

Data can be down-sampled according to time (e.g. a minimum time interval between each position report). Furthermore, it is also possible to down-sample data using a more intelligent linear approximation scheme. The linear approximation is only used when inserting data into the statistical part of the database.

The "linear approximation" principle illustrated in Figure 3 is used for thinning/pruning the AIS data before inserting it in the statistical tables. The green samples (position reports) are those that will pass the filter and the gray are those that are removed by the filter. The newest approximation is shown as a thick line and the previous one is shown as a thin line. The individual steps in the example are:

1. When the ship first arrives in the coverage area of the system the first 4 samples are used for initialising the linear approximation algorithm. Thus, the first 4 samples will always pass the filter. The next 3 samples received in this example are located within a configurable maximum distance, $\Delta_{max\ dist}$, from the line and are therefore removed by the filter.

2. The next sample is too far away from the line and will therefore pass the filter.

3. A new line will be computed based on the last 4 samples (i.e. the last green one and the three gray ones).

4. The next sample again is too far away from the line

5. A new line is computed

6. The next sample is on the line and therefore will be removed by the filter.

The linear approximation filter is combined with a maximum interval ensuring that a sample is passed at least every T_{max} seconds.

Speed variations can also be used to trigger a passed sample, if the speed between the newest sample and the previous one differs by more than, $\Delta_{max\ speed}$, then the sample is passed. The filter will also react to variations in COG (Course Over Ground), Heading and ROT (Rate Of Trun) using the $\Delta_{max\_cog,}$ $\Delta_{max\_heading}$ and Δ_{max_rot} parameters.

The sample will also be included if the "Navigational Status" or if the "Reserved for regional applications" parameter of the Position report is changed.

Figure 4 and Figure 5 shows examples of the result of using the linear filter. In this example the total number of points from the ship is about 11.000 and the samples used is about 1100, so the samples have roughly been down sampled with a factor of 10. The example uses Δ_{max_dist} = 10 meters.

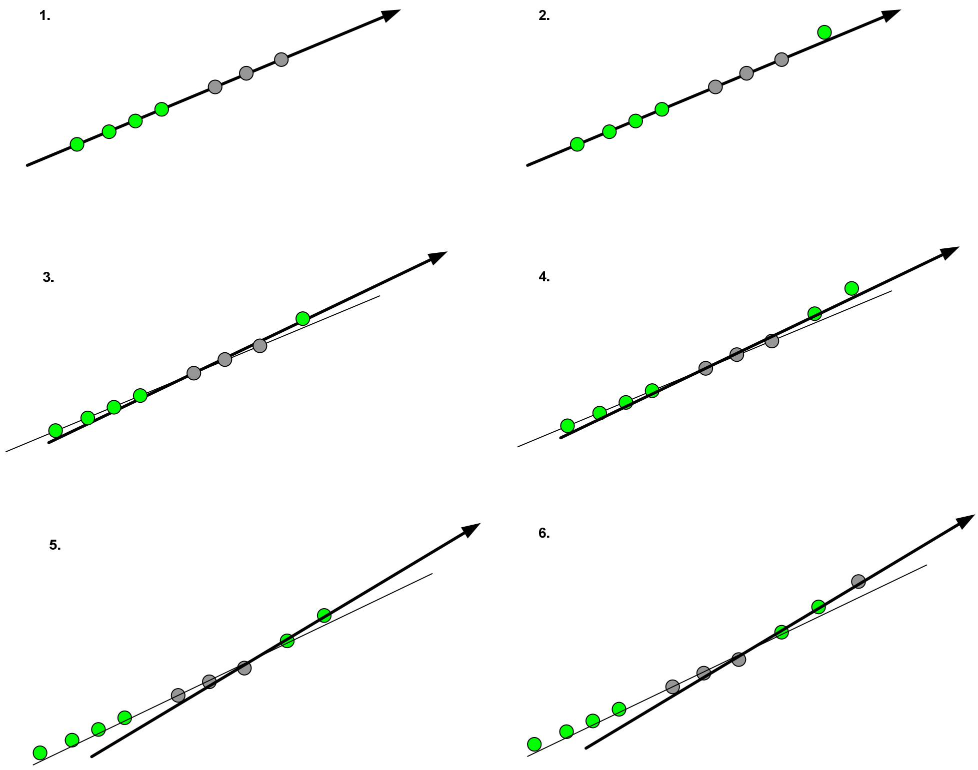


Figure 3: Illustration of the linear approximation principle

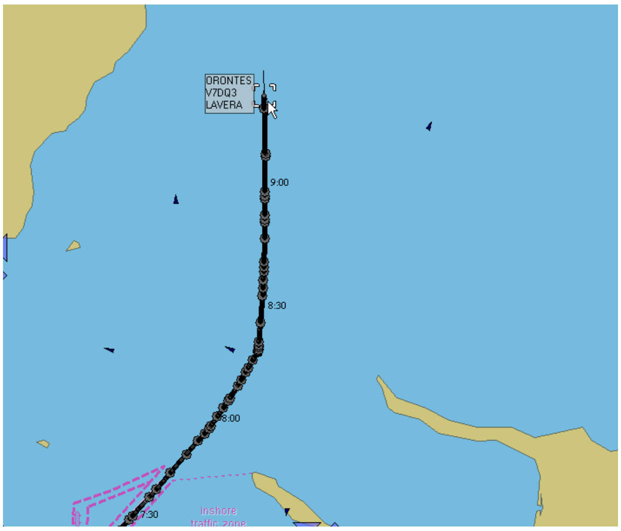


Figure 4 Example showing the linear predictor in action, accuracy 10 meters, the big dots have passed the filter

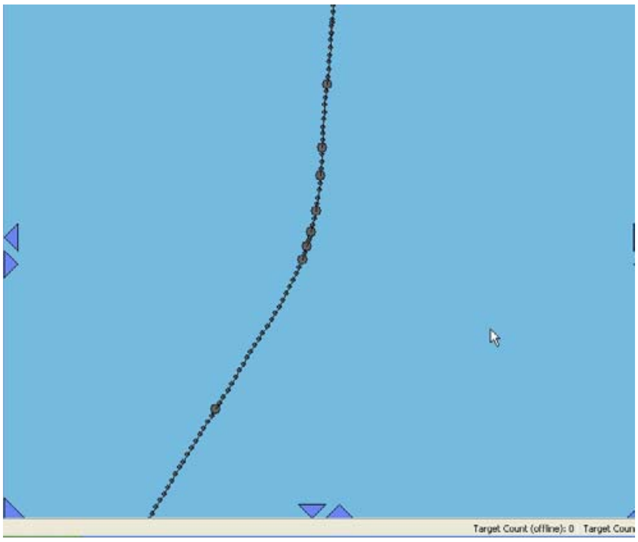


Figure 5: Zoom in on a curve, the big dots have passed the filter

.

4.2.3 DISK SPACE CONSUMPTION

Based on operation of the database in a lab environment, GateHouse has determined that a database footprint per message in the data model is on average 430 bytes per AIS message.

4.2.4 SPATIAL TEMPORAL INDEXING

The Oracle DBMS provides efficient data types to index spatial data. However, it does not provide efficient data types to support spatial temporal data.

Therefore a special index has been designed, which allows indexing spatial temporal data.

The index works by dividing the world into multiple small cells. Each of these cells is given a unique ID, using a “space filling curve”. This transforms the two-dimensionality of the geographical data into a one-dimensional problem.

This transforms the problem of querying for all ships within an area, within a given time range, to the problem of finding all position reports within a given time range AND within a given range of geographical IDs.

This allows the usage of the very efficient B-Tree indexes, which have been designed for supporting range queries.

4.2.5 SPATIAL TEMPORAL QUERYING

The usage of a specially designed index, adds to the complexity of issuing queries, as Oracle does not provide the same functionality for this index, as it does for its own spatial index.

In order to check for instance if a ship has crossed a given passage line, the query needs to be reduced to only considering the relevant geographical cells. Therefore a filter function is introduced, which from a passage line calculates the different cells needed to answer the query using information about how far a ship can sail given a particular down sampling scheme.

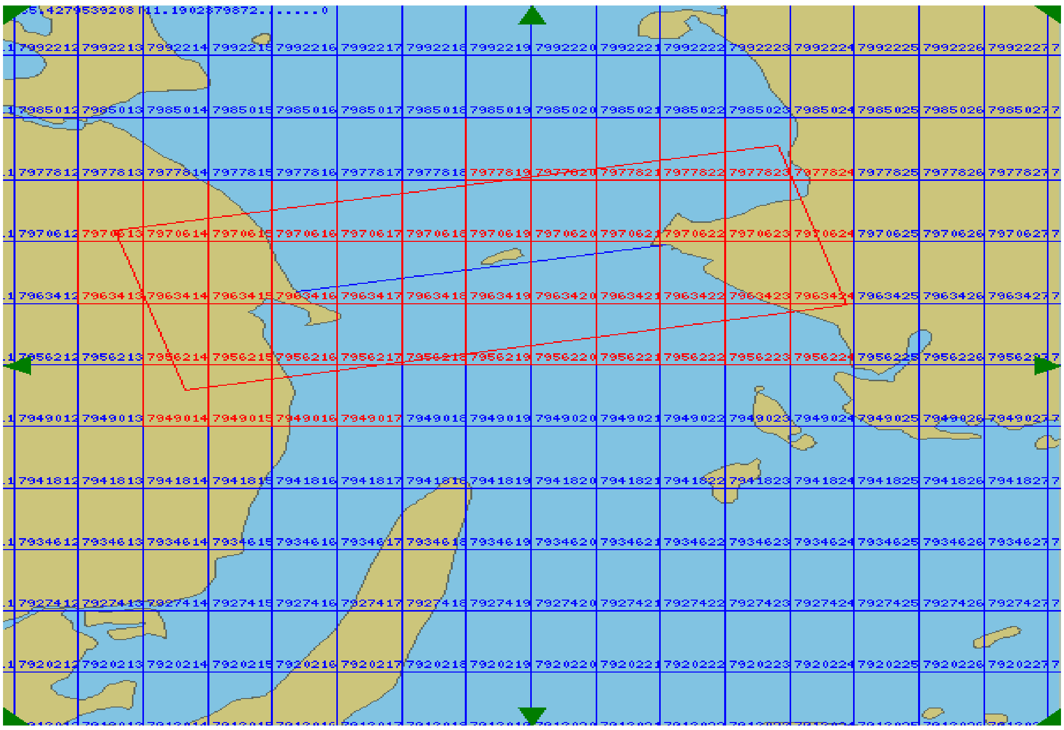


Figure 6: Querying algorithm, using spatial temporal index, using a threshold of four nautical miles.

The figure above illustrates the geographical cells needed to answer a query for a particular passage line, assuming that there can be no more than four nautical miles between each position report.

In the following it is described which steps are taken in the database architecture to ensure that scalability is ensured when the number of rows in the database grows. These are:

- Data partitioning
- Spatial Temporal Indexing
- Bulk loading + index maintenance

4.2.6 DATA PARTITIONING

The Oracle DBMS provides the functionality of transparent data partitioning. This allows breaking up a database table along with its indices in many small parts – e.g. AIS data for each separate day is stored separately in different locations on the disc. This has the advantage that indices become less fragmented, and easier to maintain. At the same time queries can be issued transparently and the Oracle DBMS can access multiple partitions concurrently, optimizing I/O usage.

4.2.7 DATA OUTPUT FORMATS

The density plot output format is:

- PDF file
- PNG file
- Comma Separated Values (.csv file)
- Zipped ESRI shape file

The two latter formats are used for processing in other systems. An example of an .csv file is shown below.

start_latitude	start_longitude	end_latitude	end_longitude	value
553.899	772.215	553.905	772.278	1
553.899	772.278	553.905	772.342	1
553.899	772.342	553.905	772.405	1
553.899	772.405	553.905	772.468	1
553.905	771.772	553.911	771.835	1
553.905	771.835	553.911	771.899	1
553.905	771.899	553.911	771.962	1
553.905	771.962	553.911	772.025	1
553.905	772.025	553.911	772.089	1
553.905	772.089	553.911	772.152	1
553.905	772.152	553.911	772.215	1
553.905	772.215	553.911	772.278	1

Tabel 1: Example of output file formatted as a Comma Separated Value file (.csv)