5-2018

Database Design and Optimization for Telemetric Aquatic Species-Tracking Systems

Bijay Regmi
University of New Orleans

Database Design and Optimization for Telemetric Aquatic Species-Tracking Systems

An Honors Thesis Submitted to the Faculty of

The Department of Computer Science of

The University of New Orleans

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science,

With University High Honors

and Honors in Computer Science

Department of Computer Science

The University of New Orleans

By

Bijay Regmi

May 2018

Acknowledgment

I would like to thank Dr. Mahdi Abdelguerfi for providing me with the opportunity to work on this project under his guidance as well as providing me with the proper resources and equipment to enable this project. I would also like to thank Nathan Cooper and Dustin Peabody for their help in the development of the project and completion of this thesis. Finally, I would like to thank Dr. N. Adlai Depano and UNO Honors Department for helping me to complete the thesis.

Contents

Table of Figures

Abstract

Tracking an individual species has always been a challenge for scientists, especially when one has to make sure to not change its natural movement pattern. When the number of individuals being tracked is increased and water is added to the equation, the task becomes next to impossible. But thanks to technologies and tracking methods like telemetry, the task of tracking any species without affecting the natural movement pattern has not only become a reality but easily accessible to scientists. Underwater acoustic telemetry has become a standard tool for fisheries biologist to study the movement pattern of the fish (Heupel). This project develops a minimalistic database designed to meet the needs of the telemetry systems. The database is optimized for storing a large number of datasets generated by the telemetry system and also for the most common queries run against the system.

Keywords: Acoustic, Satellite, Telemetry, Fish, Aquatic, Tracking, PostgreSQL, Database

Background & Summary

Environmental changes affect the distribution and movement of marine species at different spatiotemporal scales (Jaine, Schlaff). However, our ability to predict species' responses to these changes relies on accurate records of animal movement patterns. Therefore, long-term monitoring of animal movement is paramount for predicting the behavioral responses under changing environmental conditions (Hoenner).

Tracking species in natural form is a challenging task. The vastness, complexity, and opacity of aquatic environment have historically impeded the efforts to study their movement patterns. However, technological advances like telemetric systems have not only enabled biologists to track the movement of different species in space and time, but also to document the state of the ambient conditions surrounding the species (Hussey).

The Louisiana Department of Wildlife and Fisheries in collaboration with Louisiana State University at Lake Calcasieu and the University of New Orleans made use of the telemetric system to gather large quantities of acoustic data on fish at Bayou St. John (Louisana Department of Wildlife and Fisheries). *Telemetry,* a web application that developed as a successor of this study, provided a tool to visualize the movement of the fish (Bajwa). However, the database design used in the *Telemetry* web application contained some flaws which resulted in some underdeveloped simulations. The database schema of the *Telemetry* web application is not general enough. It associates an acoustic receiver with a location and also an acoustic transmitter with a fish. However, in the real world the receivers may be transferred from one location to another, and, similarly, the acoustic transmitters may be attached to different fish across different spans of time. Failure to address these real-world situations by the *Telemetry* web application resulted in its data and therefore simulations sometimes being erroneous.

This project builds on the *Telemetry* web application and aims to fix the shortcomings in its database design. Furthermore, this project aims to generalize the database design such that it can be used with any application that uses a telemetry system to track species. This new design can be used for tracking any other species with minimal changes to the database schema.

Telemetry

Telemetry is an automated process of taking measurements and transferring the data to a receiver, usually located far away from the site of measurement (NASA report). Because radio waves do not propagate effectively through an aquatic environment, aquatic telemetry is rooted in two principal approaches: acoustic (Donaldson) and satellite telemetry (Hazen).

An acoustic telemetry system consists of two main components: transmitters and receivers. Transmitters are electronic tags that broadcast a series of sound waves to the surroundings. These transmitters can be either implanted in the fish or attached to the surface of the fish. Receivers are small data logging computers, usually fixed in a certain location, that listens to the pings from the transmitters and logs the information. The diagram below shows the component of the acoustic telemetry systems.

Figure 1 Diagram showing components of acoustic telemetry systems (Acoustic Telemetry).

In satellite telemetry, a transmitter is either attached or implanted to a fish and the satellite transmitter sends the data to the land-based receiver via satellite. The setup of a satellite telemetry system is shown in the following figure.

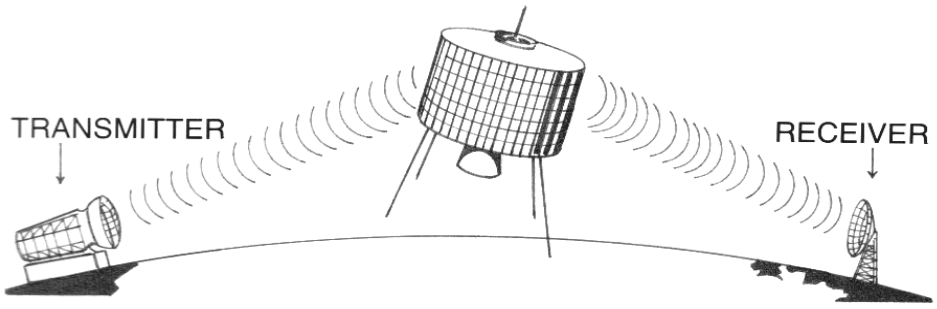

Figure 2 Figure demonstrating the setup of satellite telemetry system.

Over the past decade, the number of aquatic telemetric studies have increased six-fold, spanning all continents and biomes (Hussey). The following figure shows the global distribution of aquatic telemetry studies.

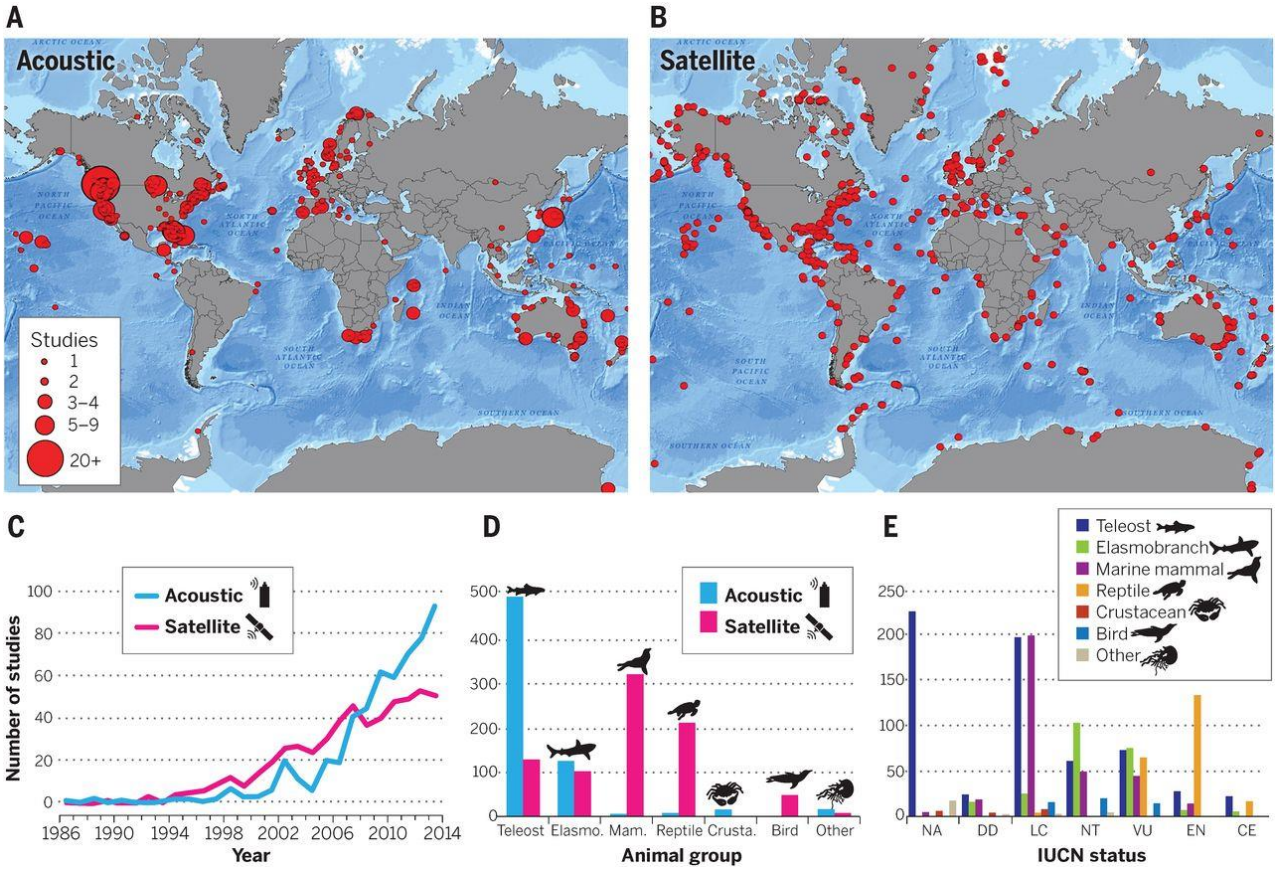

Figure 3 Distribution of aquatic telemetry studies. (A) Global distribution of acoustic telemetry studies. (B) Global distribution of satellite telemetry studies. (A) Global distribution of acoustic telemetry studies. (B) Global distribution of satellite telemetry studies (C) Graph showing the increase in aquatic telemetry studies since 1986. (D) Graph showing the distribution of aquatic telemetry studies according to major aquatic animal groups [Elasmo., elasmobranch; Mam., marine mammals (including polar bears); Crusta., crustacean; Bird, flightless marine birds only]. (E) Number of acoustic and satellite telemetry studies per major animal group by International Union for Conservation of Nature (IUCN) threat categorization [NA, not assessed; DD, data deficient; LC, least concern; NT, near threatened; VU, vulnerable; EN, endangered; CE, critically endangered](Hussey).

Laying the Foundation

We can clearly see that the number of aquatic telemetry studies made has been increasing rapidly over time. Due to continuous development in technology and better accessibility of these technologies, these numbers are expected to rise.

Therefore, this project aims to develop a minimalistic database schema that suits most of the telemetry systems. The aim here is to be as generic and modular as possible so that more studies will be able to adapt this database schema.

The Challenges

There are many challenges that need to be addressed as we develop a database design for telemetry systems. Some of the key challenges that need to be addressed are as follows:

Scalability

The amount of data that a telemetry system generates is colossal. A constant influx of detections means that the database is ever-growing. If the number of species being tracked is increased by a handful, then the size of data generated will increase exponentially. Therefore, the database should be able to scale efficiently for a large amount of data.

Efficient Querying

Using the database for research studies means that the researchers are going to be querying the database frequently for different kinds of information. Therefore, the database should be able to handle the most common queries efficiently. Most of the time, the researchers are interested in the information about the detection of species. Hence, the database should be designed such that it can quickly retrieve the species identifier, detection date, and detection location of any species efficiently.

Data Transfer

It is often the case that the data are served via web requests in the majority of applications. In most of the cases, data resides in a server and is served upon request from the client. Therefore, it will be hugely beneficial if the database facilitates easy and fast data transfer across the web.

The Solution

The database is designed to address the above mentioned key issues. The database is designed in PostgreSQL because it is free and capable of handling high volumes of activity and data (PostgreSQL 9.4.7 Documentation). This will make the schema scalable for high-volume of data.

Efficient querying is achieved primarily by pruning the data that does not add any effective informational value.

Similarly, to make the data transfer quicker the result after all the optimizations is stored as JSON objects in the materialized views. Since JSON is a widely used data format for transport across network, the final results are stored as JSON objects.

The key issue with the existing database model is that a receiver was coupled with a location and a transmitter was coupled with a fish. However, in the real world the transmitters can be implanted to a new aquatic species and the receiver can be transferred to a new location. The coupling did not provide an efficient way to address the reuse or relocation of the transmitters and the receivers. The new database design solves this issue by decoupling the fish with transmitters and receivers with locations.

Database Design

The database holds two types of information: the system metadata and the detections. The metadata represents the data pertaining to the receivers, transmitters, users, receiver installations, transmitter installations and the locations. The metadata gives the information about the status of the user, transmitters, and receivers. There are multiple validations imposed at the database level for the metadata information so that the applications using this database schema will not have to

worry about enforcing those validations at the application level. Since invalid metadata will result in multiple detection values to be faulty, these kinds of validations and restrictions are deemed essential.

The detection data contains the information about the detections themselves. A detection log consists of a receiver ID, transmitter ID and the timestamp of the detections. While storing the raw detections, very few restrictions are imposed on the data. This is to ensure that the detection data are not lost from the system even if the data seemed to be invalid upon initial inspection. Ensuring the validity of the detection is itself a complex problem, and the parameters to determine the validity varies with the area, species being monitored and different other factors. Hence, it seemed plausible not to undertake the task of detection validation and leave the burden to the application that uses this database design.

The database schema for this project is given below. Some of the key changes and techniques used in this database design is described in the sections below.

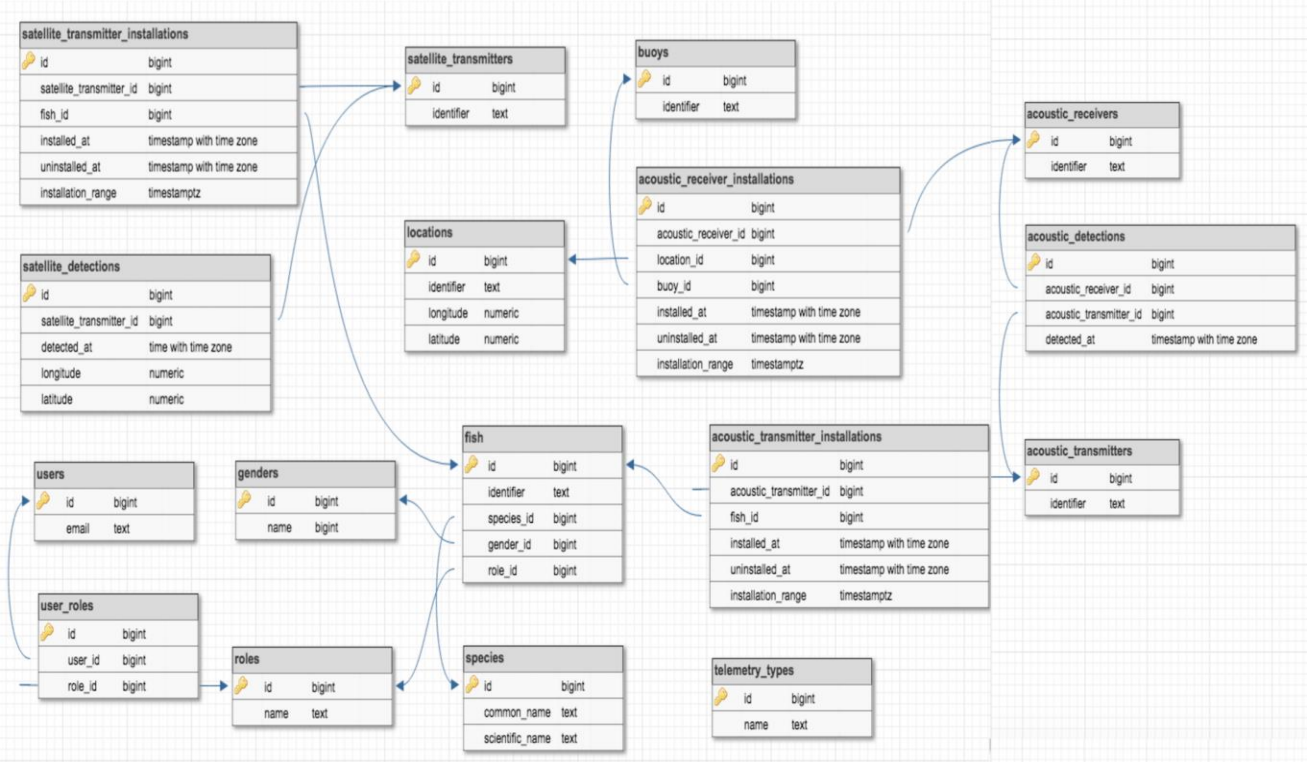

Figure 4 Database ER Diagram

Decoupling

One of the key change in this new database design is the decoupling of fish with transmitters, and receivers with locations. The information pertaining to fish and locations is stored in the table *fish* and *locations* respectively. The information pertaining to the acoustic and satellite transmitters is stored in the tables *acoustic_transmitters* and *satellite_transmitters* respectively. Similarly, the information about the acoustic receiver is stored in the table *acoustic_receivers*. The association between fish and transmitters is represented by transmitter installations (*acoustic_transmitter_installations* and *satellite_transmitter_installations*) and the association between the locations and the receivers is represented by the receiver installations (*acoustic_receiver_installations*). In this new design, the coupling between fish and transmitters, and between receivers and locations is not permanent. The coupling is over a specific temporal

region and hence this new database design addresses the issue of reuse and relocation of the transmitters and receivers.

Data Vs Information

As mentioned, a detection is simply a log of where the species was located at a given time. A satellite transmitter continuously logs its location and sends the log to the receiver via satellite. Since the fish are constantly moving and the location is logged precisely, all the data of satellite detection convey some useful information.

On the other hand, not all acoustic detection data contains the same informational value. The acoustic transmitter just broadcasts the signal and the receiver upon receiving the signal logs the entry. From the acoustic detection, we will only know that the species was at the vicinity of the receiver at the logged time, not the exact location. An acoustic transmitter can have multiple simultaneous detections logged with the same acoustic receiver. The information that all those detections provide is that the fish was in the vicinity of the receiver from the time the first detection was logged to the time the last detection was logged. If we only had the first detection and last detection from the series of detections where no detections were registered in between from other receivers, then these two detections will also provide the same information (i.e. the fish was in the vicinity of the receiver from the time the first detection was logged to the time the last detection was logged). Hence, the detections that were logged between the first and the last detections do not have any informational value.

We pruned the data in a similar fashion without losing any informational value from the original data set. After pruning the data, we ended up with relatively smaller data set. The

reduced data set helped in making the execution of queries much faster. Pruning also reduced the amount of data required to send over the network.

Materialized Views

Materialized views are stored result of the query. They are similar to views, but, unlike views, the result persists in a table-like format (PostgreSQL 10.3 Documentation).

As we know, the size of the data in the telemetry is colossal and querying against such huge dataset can be costly. As mentioned, we can effectively prune some acoustic detection data without losing any informational value. Also, as seen from the schema diagram, the detection tables inherently do not contain information about the species being detected or the location of the detection. To find the complete information about the detection we need to do some joining and filtering.

To store the reduced data set obtained after pruning the data and to store different intermediate results obtained after joining and filtering, materialized views are used in this project. Materialized views are the meat and potatoes of this revision effort. All the optimizations are done using materialized views.

Below is the diagram that illustrates the materialized views used in this project.

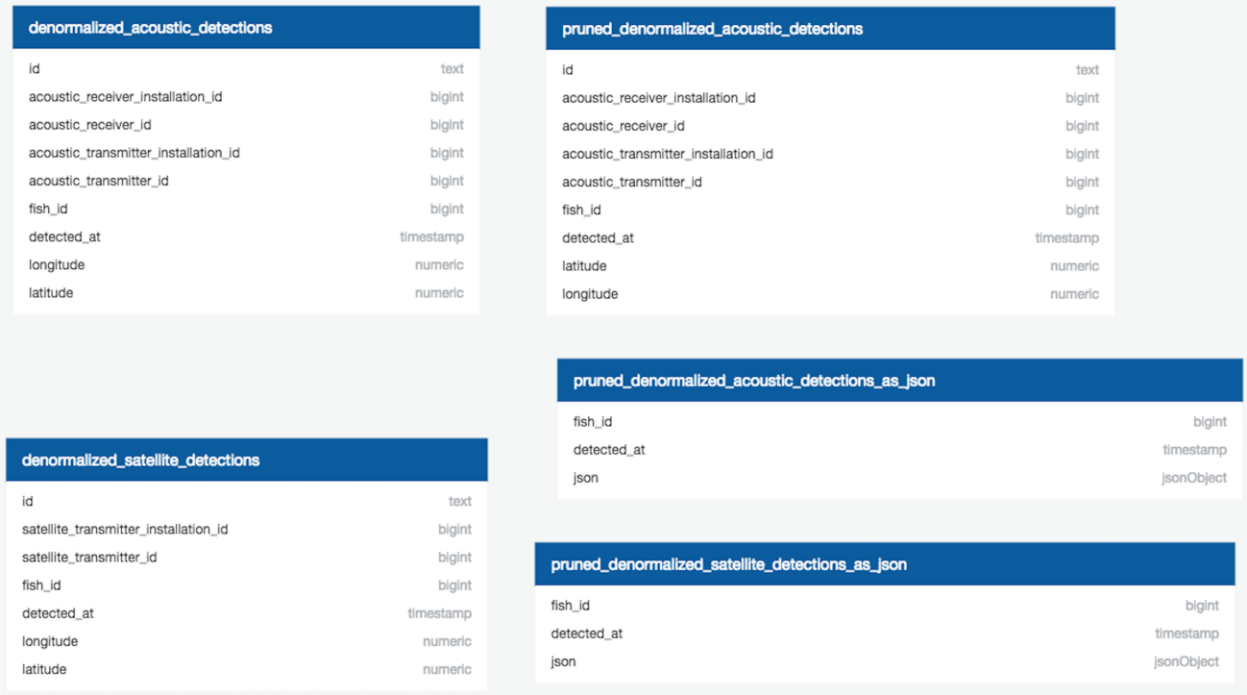

Figure 5 Diagram showing all the materialized views used in the project

Extraction of Information

In research studies, researchers want the information about the detection of the fish. The fish detection should contain the information of which fish was detected, where it was detected and when it was detected. As we know that the raw detection inherently does not contain all this information. This information should be extracted through series of joins and filters.

The following sections describe the process of extraction of required information from the original data.

Denormalization

The satellite detection table consists of columns *satellite_transmitter_id*, *detected_at*, *latitude*, and *longitude*. Just by looking at this data we can know the time and position of detection but we cannot know what was detected. Similarly, acoustic detection table consists of columns *acoustic_receiver_id*, *acoustic_transmitter_id*, and *detected_at*. From this data, we can

know the time of detection but we will not know what was detected and where was it detected. Hence, both *satellite_detection* table and *acoustic_detection* table on their own do not provide us with all necessary information needed. To get the information of the species being detected and the time and location of detection we need to denormalize the detections table by joining it with the tables that contain that specific information.

For denormalization of the *satellite_detections* table, it is joined with the *satellite_transmitter_installations* table. On the other hand, the denormalization of *acoustic_detections* table is done by joining it with *acoustic_receiver_installations*, *acoustic_transmitter_installations* and *locations* tables. The denormalized satellite detections and acoustic detections are stored in materialized views *denormalized_satellite_detections* and *denormalized_acoustic_detections* respectively.

Figure 6 shows the code snippet that performs the denormalization of the detections.

The ID of the denormalized table is obtained by prefixing the ID of the detection table with '1-' in case of acoustic detection and '2-' in case of satellite detection ((1) and (2) in figure 6). The prefixing is done to maintain the uniqueness of the ID across both tables.

Some join criteria are employed during this denormalization process. The join criteria is that the detection time has to fall inside the receiver installation range (range between receiver installation date and receiver uninstallation date) (see (3) in figure 6) and also inside the transmitter installation range (range between transmitter installation date and transmitter uninstallation date) (see (4) in figure 6). In the case of satellite detections, there is no receiver installation range and hence the detection has to fall inside the transmitter installation range (see (5) in figure 6).

```sql
CREATE MATERIALIZED VIEW denormalized_acoustic_detections AS (
  SELECT
    ('1-' || acoustic_detections.id::text) AS id,
    acoustic_receiver_installations.id AS acoustic_receiver_installation_id,
    acoustic_receiver_installations.acoustic_receiver_id AS acoustic_receiver_id,
    acoustic_transmitter_installations.id AS acoustic_transmitter_installation_id,
    acoustic_transmitter_installations.acoustic_transmitter_id AS acoustic_transmitter_id,
    acoustic_transmitter_installations.fish_id AS fish_id,
    acoustic_detections.detected_at AS detected_at,
    locations.latitude AS latitude,
    locations.longitude AS longitude
  FROM
    acoustic_detections
    INNER JOIN acoustic_receiver_installations
      ON acoustic_receiver_installations.acoustic_receiver_id = acoustic_detections.acoustic_receiver_id
      AND acoustic_receiver_installations.installation_range @> acoustic_detections.detected_at
    INNER JOIN locations
      ON acoustic_receiver_installations.location_id = locations.id
    INNER JOIN acoustic_transmitter_installations
      ON acoustic_transmitter_installations.acoustic_transmitter_id = acoustic_detections.acoustic_transmitter_id
      AND acoustic_transmitter_installations.installation_range @> acoustic_detections.detected_at
) WITH DATA;

CREATE INDEX idx_denormalized_acoustic_detections_fish_id ON denormalized_acoustic_detections (fish_id);
CREATE INDEX idx_denormalized_acoustic_detections_detected_at ON denormalized_acoustic_detections (detected_at);


CREATE MATERIALIZED VIEW denormalized_satellite_detections AS (
  SELECT
    ('2-' || satellite_detections.id::text) AS id,
    satellite_transmitter_installations.id AS satellite_transmitter_installation_id,
    satellite_transmitter_installations.satellite_transmitter_id AS satellite_transmitter_id,
    satellite_transmitter_installations.fish_id AS fish_id,
    satellite_detections.detected_at AS detected_at,
    satellite_detections.latitude AS latitude,
    satellite_detections.longitude AS longitude
  FROM
    satellite_detections
    INNER JOIN satellite_transmitter_installations
      ON satellite_transmitter_installations.satellite_transmitter_id = satellite_detections.satellite_transmitter_id
      AND satellite_transmitter_installations.installation_range @> satellite_detections.detected_at
) WITH DATA;

CREATE INDEX idx_denormalized_satellite_detections_fish_id ON denormalized_satellite_detections (fish_id);
CREATE INDEX idx_denormalized_satellite_detections_detected_at ON denormalized_satellite_detections (detected_at);
```

Figure 6 Code snippet showing denormalization of acoustic and satellite detections. (1,2) Code snippet showing the prefixing of IDs with 1 and 2 respectively. (3,4) Code snippet showing the constraints applied for denormalizing acoustic detections. (5) Code snippet showing the constraints applied for denormalizing the satellite detections.

If the uninstallation date is not set then we assume the range stretches from installation date to infinity. The range of valid detection date range is illustrated in the following diagram.

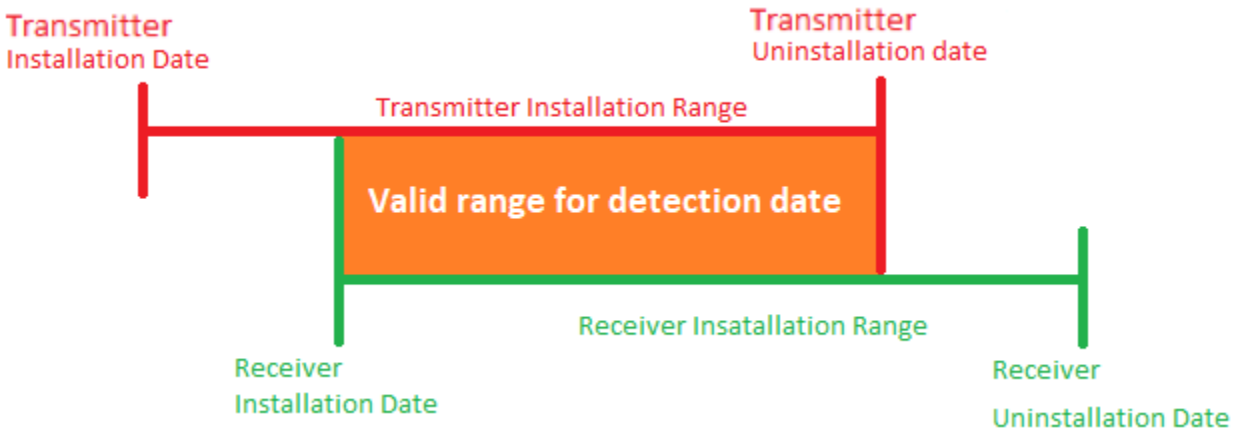

Figure 7 Diagram showing the valid detection range

Pruning

As discussed earlier, not all data of acoustic detections have the same informational value. We can prune some of the acoustic data without losing information. The detections that do not add any effective informational value are the detections that fall in between the first and last detections of a series of consecutive detections. In other words, if a series of detections are logged between a receiver and a transmitter with no detections from other receivers in between, then the detections that fall in between the first and the last detection of this series of detections can be safely pruned. This will not alter the informational content of the original dataset.

Pruning will result in a smaller dataset in the materialized views and running a query against the smaller dataset will be a lot faster. Because of the benefits of pruning, the original data is pruned and the result of the pruning is stored in the materialized view *pruned_denormalized_acoustic_detection*. The SQL statement for pruning the data is given below.

Figure 8 Code snippet for pruning the acoustic detection data. (1) Code snippet showing the selection of all rows from the table denormalized_acoustic_detections. (2) Code snippet for the partitioning of the table based on fish_id. (3) Code snippet showing the usage of lead and lag functions of PostgreSQL

The process of pruning involves many steps. First, all the columns from the table *denormalized_acoustic_detections* are selected (see (1) in figure 8). Next, the resulting data is partitioned according to *fish_id* and ordered in ascending order of detection time in order to create a windowed view (see (2) in figure 8). This will ensure that we have a windowed view of each fish that is arranged according to the detection time. Then the *lead* and *lag* function of PostgreSQL (see (3) in figure 8) are used to find the *acoustic_receiver_installation_id* of the previous and the next detection in the same window (if there does not exist any previous or next detections within the same window then the *previous_acoustic_receiver_installation_id* or *next_acoustic_receiver_installation_id* results in NULL accordingly).

From this windowed view, we will then select the *denormalized_acoustic_detections* entries whose *previous_acoustic_receiver_installation_id* is NULL or does not match the current

acoustic_receiver_installation_id. A similar technique is used for the *next_acoustic_receiver_installation_id*. This ensures that we will prune all the detection that falls between the first and last detection from the series of detections occurred between an acoustic receiver and an acoustic transmitter. The process of pruning is explained with an example below.

Let us assume that we have the following associations between acoustic receiver installations, acoustic receivers, acoustic transmitter installations and fish.

acoustic_receiver_installation_id	*acoustic_receiver_id*
10	1
15	2
20	3

fish_id	*acoustic_transmitter_id*
N 111	10
222	20
333	30

ow let's assume that we have following detections as per the association described above:

acoustic_transmitter_id	*acoustic_receiver_id*	*detected_at*
10	1	March 10, 2018, 08:00 AM
10	1	March 10, 2018, 10:00 PM
20	3	March 10, 2018, 12:00 PM
10	1	March 11, 2018, 01:00 AM
10	1	March 10, 2018, 05:00 PM
30	2	March 15, 2018, 03:00 PM
10	1	March 14, 2018, 02:38 PM

The following table shows the detection stated above with the associated *fish_id*. This information along with other information is obtained after denormalization of the detections table:

fish_id	....	*acoustic_receiver_installation_id*	*detected_at*
111	….	10	March 10, 2018, 08:00 AM
111	….	10	March 10, 2018, 10:00 PM
222	….	20	March 10, 2018, 12:00 PM
111	….	10	March 11, 2018, 01:00 AM
111	….	10	March 10, 2018, 05:00 PM
333	….	15	March 15, 2018, 03:00 PM
111	….	10	March 14, 2018, 02:38 PM

With the partition with respect to *fish_id* and ordered according to detection time, the table looks as below:

fish_id	....	*acoustic_receiver_installation_id*	*detected_at*
111	….	10	March 10, 2018, 08:00 AM
111	….	10	March 10, 2018, 05:00 PM
111	….	10	March 10, 2018, 10:00 PM
111	….	10	March 11, 2018, 01:00 AM
111	….	10	March 14, 2018, 02:38 PM
222	….	20	March 10, 2018, 12:00 PM
333	….	15	March 15, 2018, 03:00 PM

Now the *lead* and *lag* function of PostgreSQL will find the *previous_receiver_installation_id* and *next_receiver_installation_id*. If there does not exist any previous or next acoustic receiver installation ID within the window then the installation ID is set to NULL. After the previous and next acoustic receiver installation ID is populated, the table looks as follows:

fish_id	…	*acoustic_receiver_installation id*	*detected_at*	*previous_receiver_installation id*	*next_receiver_installation_ id*
111	…	10	3/10/18, 8:00AM	null	10
111	…	10	3/10/18, 5:00PM	10	10
111	…	10	3/10/18, 10:00PM	10	10
111	…	10	3/11/18, 1:00AM	10	10
111	…	10	3/14/18, 2:38PM	10	null
222	…	20	3/15/18, 3:00PM	NULL	NULL
333	…	15	3/10/18, 8:00AM	NULL	NULL

Now we can apply the pruning condition and remove the unnecessary rows. For the rows to be removed, its previous or next receiver installation ID should either be NULL or not match its own receiver installation ID. With that constraint, the middle three rows from the first window are eliminated. After pruning, the table looks as below:

fish_id	…	*acoustic_receiver_installation*	*detected_at*	*previous_receiver_installation*	*next_receiver_installation_*
111	…	10	3/10/18,	NULL	10

			8:00AM		
111	…	10	3/14/18, 2:38 PM	10	NULL
222	…	20	3/15/18, 3:00PM	NULL	NULL
333	…	15	3/10/18, 8:00AM	NULL	NULL

Initially, we had 7 rows and after pruning, we ended up with just 4 rows. Nevertheless, we still retain the same information (i.e. fish with id 111 lingered around receiver with ID 1 from March 10, 2018, 8:00 AM to March 14, 2018, 2:38 PM) even though we filtered out some of the detections.

Truncation of Metadata

A detection has to provide the information about the fish that was detected, the location of the detection and the time of detection. The *pruned_denormalized_acoustic_detections* table contains extra metadata information like *acoustic_transmitter_installation_id*, etc. Hence, we need to trim down the metadata and extract only the needed information i.e. species ID, detection time and detection location. For that purpose, the union of tables *pruned_denormalized_acoustic_detections* and *denormalized_satellite_detections* is taken and the information regarding fish, location and the time of detection is extracted. This extracted information is stored in the view *pruned_denormalized_detections*. The final result is saved in a view so that the queries can be run against this view to get the information about any type of detection.

Optimization for Data Transfer

The detection data resides on a server and is served upon request from the application. To serve the data to the application, we need to transfer the data across the network. Most modern

web frameworks use JSON data structure to transfer the data across the network. However, building a JSON object can be an expensive task. Even if the database query is fast, if the transfer of data takes a much longer time, the overall performance of the system is decreased. Hence, to make the transfer of the data across fast, storing the data as JSON objects seemed plausible. Although using this approach we would essentially duplicate data, and incur a storage cost, the benefit of storing the data in JSON format significantly outweighs the cost associated with it. Storage is relatively cheap and the end user will value the speed of the application. Hence, the detection data is stored in JSON format in the materialized views *pruned_denormalized_acoustic_detections_as_json* and *pruned_denormalized_satellite_detections_as_json*. These materialized views contain *fish_id, detected_at* and *json* columns where *json* column holds the detected time and detected location as a JSON object.

To get detection data from both acoustic and satellite system, the union of *pruned_denormalized_acoustic_detections_as_json* and *pruned_denormalized_satellite_detections_as_json* is taken and stored in a view *pruned_denormalized_detections_as_json.*

Future Works

In order to improve the application and provide the users with better tools, a minimalistic web application that simulates the fish movement using this database design will be built. We have already implemented the basic administration views and maps for the project. Also, we have designed and implemented the Application Programming Interface (API) to get the data

from this database design. Now, we will have to run the simulation in the front-end using the data obtained from the database.

Also, to get the better understanding of the efficiency of this database schema, we will benchmark the performance of this database design against different situations. Also, we will compare the performance of this database schema with the previous implementation.

Another area where the database can be improved is the interpolation method used in the tracking algorithm. In the current database design, if the species falls in the overlapped region of two receivers, then both receiver will hear the ping broadcasted by the transmitter. This will cause both receivers to log the detections and it seems like the species is moving back and forth between the two receivers really quickly. In reality, the species was just lying in the region where more than one receiver can hear its broadcast. Methods exist to fix this issue in the future using triangulation methods wherever possible to infer the actual location of the fish. Also, we plan on developing means to detect detection logs from dead fish and filter out such detections.

Conclusion

This project fixes the problem in the database design of *Telemetry* web application and provides a minimalistic database that can be used for any aquatic telemetry system. Designed in PostgreSQL, the database schema has no cost associated with it and is capable of handling a large amount of activity and data. Moreover, the database is optimized for the most frequently used queries and also for data transfer across the network.

Sources

"Acoustic Tagging." *Fish Louisiana*. Louisiana Department of Wildlife and Fisheries, 20 May 2015. Web. 11 August. 2015. <http://www.fishla.org/fisheries-management/fish-tagging-programs/acoustic- tagging/>.

"Acoustic Telemetry." *Great Lake Acoustic Telemetry Observation System.* Web. 02 April, 2018. <https://glatos.glos.us/acoustic>

Bajwa, P. S. 2016.*Visualizing Aquatic Species Movement with Spatiotemporal Data from Acoustic and Satellite Transmitters.* Undergraduate Honors Thesis, University of New Orleans, 2016. Web. April 05, 2018.

Donaldson, M. R. et al., Making connections in aquatic ecosystems with acoustic telemetry monitoring. Front. Ecol. Environ 12, 565–573 (2014). doi: 10.1890/130283

Hazen, E. L. et al., Ontogeny in marine tagging and tracking science: Technologies and data gaps. Mar. Ecol. Prog. Ser. 457, 221–240 (2012). doi: 10.3354/meps09857

Heupel, M. & Webber, D. Trends in acoustic tracking: where are the fish going and how will we follow them? Advances in Fish Tagging and Marking Technology. American Fisheries Society Symposium 76, 219–231 (2012).

Hoenner, X. et al. "Australia's Continental Scale Acoustic Tracking Database and its Automated Quality Control Process." Jan 30, 2018. *Scientific Data Vol 5.* Web. March 20, 2018

Hussey, N. E. et al. Aquatic animal telemetry: A panoramic window into the underwater world. Science 348, 1255642 (2015).

Jaine, F. R. A. et al. When giants turn up: sighting trends, environmental influences and habitat use of the manta ray Manta alfredi at a coral reef. PLOS ONE 7, e46170 (2012).

"PostgreSQL 9.4.7 Documentation." *PostgreSQL: Documentation: 9.4: PostgreSQL 9.4.7 Documentation*. The PostgreSQL Global Development Group. Web. 11 Mar 2018. <http://www.postgresql.org/docs/9.4/static/>

"PostgreSQL 10.3 Documentation." *PostgreSQL: Documentation: 10.3: PostgreSQL 10.3 Documentation*. The PostgreSQL Global Development Group. Web. 11 Mar 2018. <https://www.postgresql.org/docs/10/static/rules-materializedviews.html>

Schlaff, A. M., Heupel, M. R. & Simpfendorfer, C. A. Influence of environmental factors on shark and ray movement, behaviour and habitat use: a review. Reviews in Fish Biology and Fisheries 24, 1089–1103 (2014)

"Telemetry: Summary of concept and rationale." *NASA report.*1987. Web. March 25, 2018.