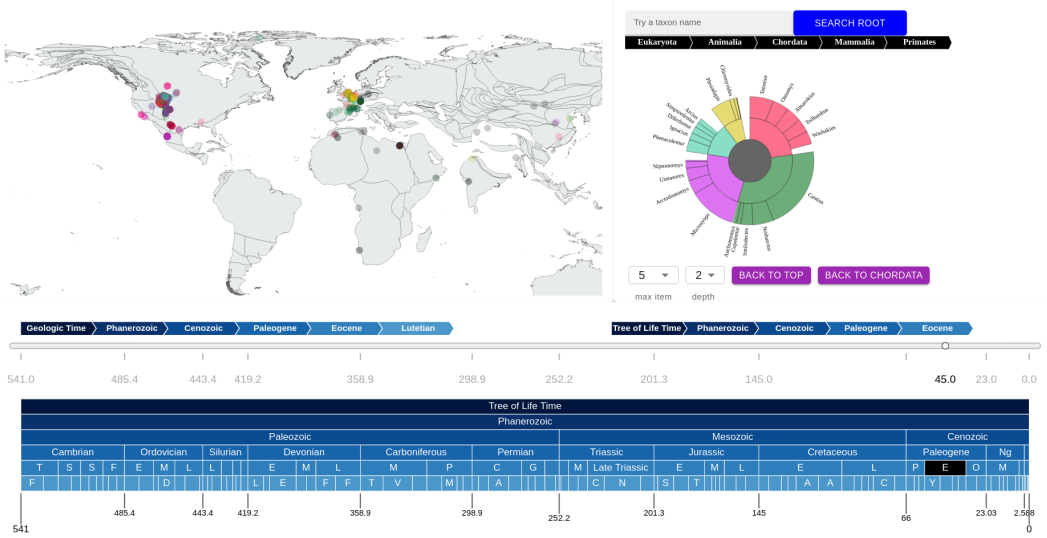
# A visualization of biodiversity through time




Master Thesis
January 19, 2022

by Wuxing Dong, 18-946-038

Supervisors:
Prof. Dr. Renato Pajarola
Gaudenz Halter

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

# Abstract

# Contents

# 1 Introduction

## 1.1 Effective visualization of data

In the time of the information age, the Internet has become a primary source to educate ourselves about the world. It is therefore a great opportunity for the scientific community to take advantage of the web and create high quality products to communicate their findings and ideas to the non-experts and general public. Textbook-styled verbal explanations can be rigorous, however, they easily become tedious and are already made available by school and college education. Videos are based on visual learning, quick to absorb, yet there is no engagement with the viewers. An interactive data visualization can provide an excellent way for users to understand scientific concepts. New knowledge is conveyed by direct visual impact that easily draw attention, and insights can be formed by actively exploring the data and observing the response of the presentation due to the users' own actions. This learn-by-doing strategy can consolidate the new knowledge.

Although graphics plays a major role to deliver scientific messages, there is a lack of formal theory on producing good graphics, and domain experts tend to rely on practical experiences that are guided by certain design principles. Increasing amount of studies have focused on concluding such principles. Chen and colleagues have summarized that three main aspects to pay attention to in a visualization project are its contents, the relevance in the context of the larger whole as well as the forms and aesthetics (construction). Similarly, McCandless argued that a good visualization requires integrity of information, interestingness of story (concept), usefulness of goal and beauty of the visual form.

Before embarking on the production of a visualization, one should note the characteristics of human visual perception to improve the design strategies in practice. Cognitive psychological studies aimed at examining human visual attention by exposing users to certain visual processing tasks. The results of those studies suggested some guidelines for more effective visualization. When the visual features of a certain type of data is expected or previously known, the accuracy and speed of visual perception is drastically improved. Therefore, a visual feature should be assigned to the same data dimension, or type of data. It is also important to note that attention is limited and selective, therefore the amount and diversity of information should be kept within the limits of cognitive load. When many categories of data are present, compromises are needed to retain graphics complexity and only demonstrate a few number of categories at a time.

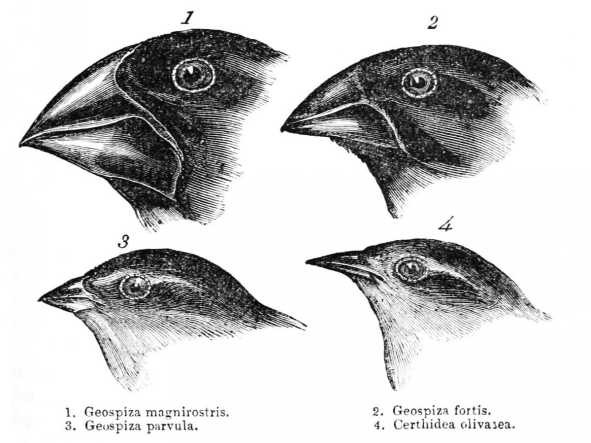
Through early years of research on how images are analyzed visually, it has been found that detailed vision is only achievable by actively focusing on small portion of the visual field. However, a few visual features, such as colors and curvatures, are subjected to preattentive processing, which can be effortlessly and rapidly detected by low-level visual processes. Colors is thus a powerful method in graphics. It can encode extra layers of information, and serves a leading factor to make a visualization memorable. Three major schemes for using colors are sequential, diverging and qualitative. A sequential scheme relates increasing value of the data to the intensity of the color, while using only one or two hues. In diverging scheme, two extremes are shown by two sequential schemes, and the middle or neutral value corresponds to a white or neutral color. The qualitative scheme uses a variety of distinguishable colors to represent data belonging to distinct categories. It is a good practice to choose color schemes that are distinguishable enough, such that minimal information is lost when converted into a gray scale. In addition, when a range of colors are used together, reducing the opacity of certain or all colors avoids overwhelming visual impact to the viewers.

Design principles have also been examined and summarized by different studies. The design process should be guided by the core messages to deliver. It is important to first prioritize what information to show, envision the final result, and then choose appropriate geometries that fulfill the objectives. Geometries are the forms in which data is illustrated, and the majority of them can be categorized into amounts, compositions, distributions or relationships. Often, different geometries can be considered on the same data, however, the most efficient geometry should be of high data-ink ratio, which is the ratio between the amount of ink allocated on the actual data to the total amount of ink used in a figure. While a high data-ink ratio indicates high efficiency, it is also important for a visualization to be simple, self-explanatory and straight to the points, such that the viewers are not overwhelmed with too much information.

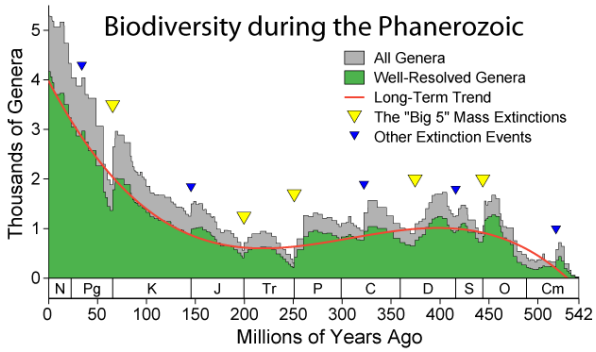
## 1.2 Background on biodiversity and tectonic change

The data this thesis chooses to visualize is biodiversity through earth history. Considering various pressing environmental issues nowadays, it is important to raise the awareness of the general public on how different forms of life came to exist and change during the course of earth history. Climate change has become a non-negligible factor influencing on our daily life and the future of human society. Recently, the world sees increasing amount of extreme weather conditions, causing significant losses in human lives and properties. The ecosystem is also under threat of a mass extinction event, after which 90% of all species are expected to disappear, and its leading cause is considered to be human activity. By understanding the history of life through simple and interesting visualizations, viewers could appreciate the inseparable relationship with our environment, and be more promoted to act in a more environmentally-friendly and sustainable manner.

### 1.2.1 Biodiversity as a result of evolution




(a) John Gould's sketches of the various beak shapes of the Galapagos finches as a result of evolution.




(b) Biodiversity during Phanerozoic experiences non linear change with mass extinction events.

Current views on the process of species formation is based on modern evolution theory originated from Darwin's work. To summarize briefly, all forms of life are argued to share a common ancestor, which diversified over the course of evolution to form the current tree of life. An organism's genomic information is encoded in DNA or RNA, which is replicated over generations. However, recombination and random mutation events occur during the replication process, resulting in genetic variations, which in turn causes difference in phenotype, the observable traits of organisms. Individuals with traits that are more suitable to their particular environment are more likely to survive and pass their genes to their offspring, comparing to those that are poorly adapted to the environment. This process is termed natural selection. A classic example of evolution is the adaptive radiation of finches on the Galapagos islands. The beak shapes of the finches on different islands diversified to specialize

on obtaining the dominant type of food on different islands. Over generations, species evolve as a consequence of the change in the prevalence of advantageous genes. Finally, a new species is formed during the process of speciation. It occurs when enough genetic variations are accumulated between the original population and the descendant population, such that organisms from the two groups are not capable of producing fertile offspring, namely, the two groups have reproductive isolation.

It is believed that life emerged from 3.5 billion years ago. The past 540 million years ago (the Phanerozoic Eon) sees rapid development and diversification of life. Throughout evolutionary history, there are records of drastic changes in biodiversity occurring in a relatively short period of time. Radiations are rapid increase in biodiversity. An example is the Cambrian explosion, during which most phylums of multi-cellular organisms appeared. On the contrary, biodiversity experiences rapid decline during mass extinction periods. There is evidence of five mass extinction periods, one of which occurred in Cretaceous and Tertiary, when the dinasours were exterminated and mammals thrived due to their selective advantage given by their fur in the cooling environments. These periods of radiation or extinction are caused by special events, plate motions, as well as change in climate, such as temperature and oxygen level.

### 1.2.2 Tectonic shift as a driver of biodiversity change

The earth crust is composed of a number of tectonic plates undergoing constant changes. The hear generated from the interior of the planet causes the movement of the plates, as a result, earth appears very differently throughout millions of years of history. The plate motion results in new landscapes and new continents, affecting the inhabitants of lives. For instance, the current continents are thought to belong to one super continent called Pangaea, and gradually drifted away from each other since 175 million years ago.

As the continents separated, populations from the same species are geographically isolated into their particular environments that is created from the plate motion. Since organisms are under constant selective pressure from the environment, new species are formed in the newly created environments. An example is the distribution of ratite as a result of tectonic movement.

## 1.3 Source of data

*The fossil*
The biodiversity of ancient time periods is represented by fossil records. For this purpose, we use paleodb, which contains 1.3 million occurrences of fossil records. Each record has its unique ID, identified taxonomy, estimated maximal and minimal time (in million years) to today, and their coordinates.

*The tree*
All identified species are classified in taxonomy, following the hierarchy of domain, kingdom, phylum, class, order, family, genus and species. Therefore, it can be naturally organized into a tree structure. Conveniently, for each fossil record in PBDB dataset, the names on each rank of the taxonomy is available. The highest taxonomy rank in the PBDB dataset is kingdom, namely the kingdom of Animalia (animal) and Plantae (plants). Since the two kingdoms are under the domain of Eukaryota, a rooted tree of life can be generated from the taxonomy information of all fossil record.

*The map*
The geological information is provided by Matthew and colleagues, who traced the coastlines of the continents from 410 million years ago until the current world.

## 1.4 Aim of the thesis

This thesis aims at building a web application that visualizes the change in biodiversity through the course of earth history. In the final product, users can view a customizable tree of life during a selected time period under a selected taxonomy. In the meantime, the fossil records belonging to the given tree of life is shown on the map, whose coastlines correspond to that of the selected epoch in ancient time.

## 1.5 Thesis structure

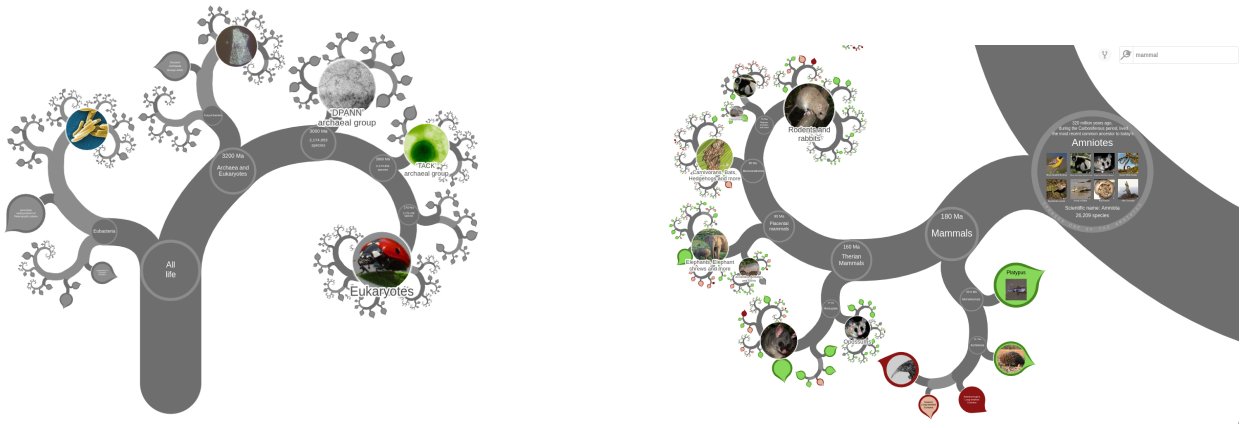
Chapter one introduces the topic of data visualization and domain specific background information, and it claims the goal of the thesis. Chapter two analyzes some related work, indicating what could be learned and improved for the visualization to be built. Chapter three outlines the problems to be solved at each stage for building the web application. Chapter four discuses the technical solutions for data processing and web development stage, respectively. In chapter five, the implementation methods for data processing, backend and frontend are listed. Chapter six reports the result of the product. Chapter seven concludes the project and suggests future work.

# 2 Related work

There already exists an abundance of visualizations on the Internet regarding to tree of life as well as plate motion. This section selects several related work and discusses their inspirations for designing our own visualization on this topic.

## 2.1 Tree-based visualizations on biodiversity

OneZoom (https://www.onezoom.org/, Fig.2.1) visualizes all known living things in a fractal tree, and it is targeted for both professionals and the masses. Users can zoom into or out of different levels of the tree by scrolling, and can search for a particular location in the tree by its name.




(a) The overview of OneZoom's tree from root.

(b) The view on OneZoom's tree when zoomed onto mammals.

Figure 2.1: Tree of life implementation in OneZoom.

The fractal design gives the tree a nice appearance, with leaves and nodes resembling the leaves and branches of an actual tree. General information such as pictures or earliest time of a node is also well packed in the interactive image. There is a reset button to return to the root, and a drop down list that provides links to the nodes along the path to root from the current position. These functions are convenient for navigation and similar functionality should be implemented in this thesis. It is claimed that all categorized organisms, which has a total amount of over two million, are included in the tree. Each species in the tree structure are incorporated with a phylogenetic classification, indicating that for a particular species represented by a leaf to reach the root, it visits all of the nodes representing each of its ancestor throughout the evolutionary history. Although how a certain organism evolved from the common ancestor of all life can be rigorously inspected, from the user experience's point of view, the tree is extremely deep. Apart from manually searching for a name, navigating to a desired location in this deep tree is mainly achieved by scrolling and dragging the item of interest to the center of the screen. This makes it difficult for viewers to relocate to a different section of the tree, or to gain an overview of where the current node or leaf sits in the entire tree of life. To reach the root (2.1(a)) from mammal (2.1(b)) while viewing its path, users need to scroll through all upper levels of the similar-looking spiral structures of the tree, which appears to be confusing. A similar tree of life is Lifemap (http://lifemap.univ-lyon1.fr/explore.html), which ameliorates the navigation difficulty by highlighting the ancestry path from root to an entry (Fig2.2). However, due to the

sheer depth of the tree, users still need to travel deep into the tree by scrolling. Rather than demonstrating the exact ancestral paths, the purpose of the tree for this project is to organize each species such that biodiversity can be viewed in an intuitive way. As a consequence, all entries can be organized in only the main taxonomic ranks, limiting the tree's depth to be eight and thus simplifying the navigation process. In addition, to avoid scrolling and dragging, the tree can take a fixed position, while where a node or leaf locates can be shown in a clickable breadcrumb structure that provides an overview of the whole tree.

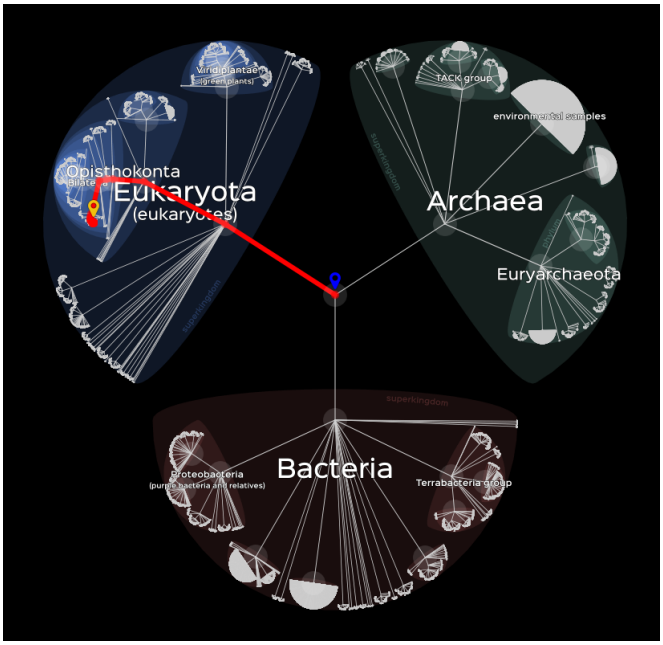
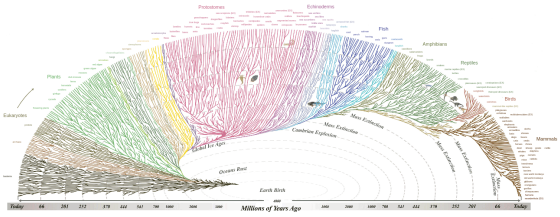
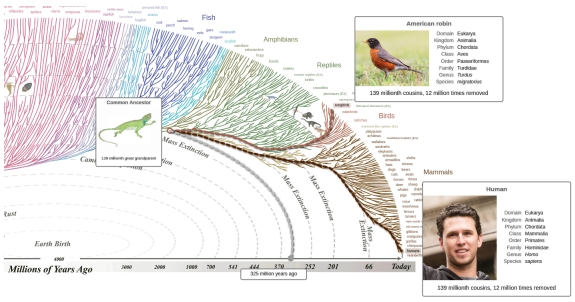



Figure 2.2: Lifemap enables a "path from root" functionality that highlights the ancestry path from *Homo sapiens* to the root.

A much simplified version of tree of life is Evogeneao (https://www.evogeneao.com/en/explore/tree-of-life-explorer), which is intended for school education. As shown in Fig2.3, all species are grouped into clearly colored groups, which gives the viewers a direct and memorable visual impact. A similar design could be adapted in this project. With the tree of life arranged in a half circle that is labeled with time and a few iconic events, users could get a view of how life radiated from the common ancestor into great diversity, and how this process is affected by special events on the planet. Since only a small subset of all organisms are shown, the tree is small and can be entirely shown, thus always giving the audience an overview. However, this web application provides little possibility of exploration due to its limited options for interaction, as the main functionality is to show the common ancestor of two selected items (Fig2.3(b)).



(a) The overview of Evogeneao's tree.




(b) The ancestral relationship between human and songbird.

Figure 2.3: Tree of life implementation in Evogeneao.

Finally, there are a wide variety of tree of life visualized for computational biologists. Such visualizations provide rich functionalities for bioinformatic analysis that are well beyond the need of general public and often take efforts to use. An example is iTOL (https://itol.embl.de/itol.cgi,Fig2.4). The design of arranging all leaves on the edge of a circle and all nodes in its interior utilizes space effectively, therefore can be adapted in our implementation.

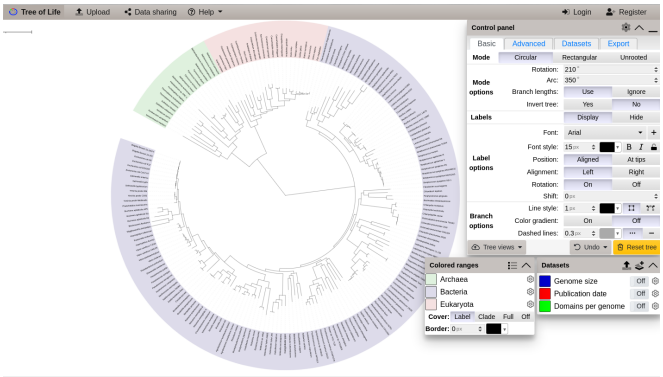
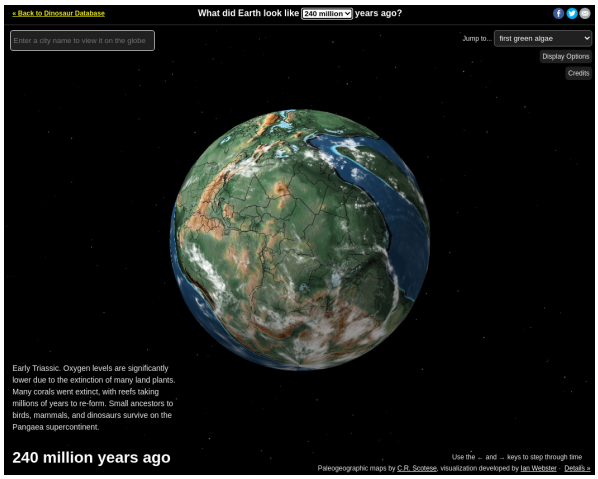
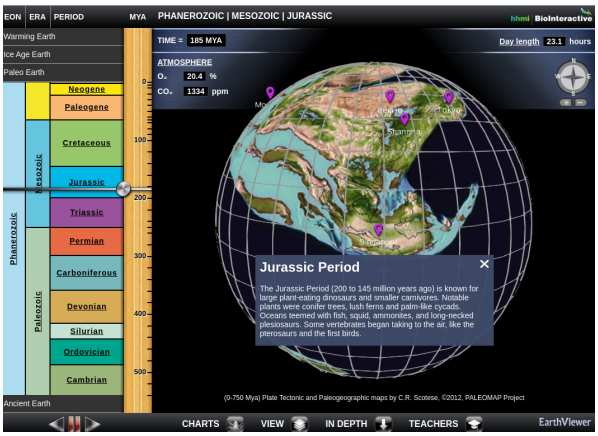



Figure 2.4: Tree of life in iTOL.

## 2.2 Visualizations on tectonic shift




(a) Tectonic shift visualized in Ancient Earth.




(b) Tectonic shift visualized in Earth Viewer.

Figure 2.5: Example visualizations on tectonic shift.

Current project involves visualizing plate motions in various time period, since biodiversity information is shown on the world map during the select time period of interest. Two typical tectonic shift visualizations are Ancient Earth (https://dinosaurpictures.org/ancient-earth#240, Fig2.5(a)) and Earth Viewer (https://www.biointeractive.org/classroom-resources/earthviewer, Fig2.5(b)). Both of them show the coastlines of the continents at a select time point on a three-dimensional globe. Ancient Earth allows users to pick a particular time point from a list of options and search for the ancient location of a city. Earth Viewer provides more detailed information on geological time, and selectable time points are incremented at every five million years on the scroll bar. Time periods are organized in a hierarchical structure with names, which can be adapted in this project for easy navigation. When clicking on a time period, instead of directing the user to the selection, a pop-up window containing a brief introduction of the time period appears, which is counter-intuitive. An improvement is to show the earth at the middle of the selected time period upon clicking.

# 3 Problem statement

The data visualization project is divided into several sub-problems as follows.

- **Data processing**
  - Process the PBDB dataset to extract fossil records
  - Extract a tree of life from the taxonomy information of the fossil record
  - Trace the coordinates of each fossil record to ancient time
- **Database setup**: store the fossil data, coordinate data and the taxonomy data into a database
- **Back-end building**: enable data queries from the database upon users' request from the front-end
- **Front-end building**: visualize interactive components for users to explore biodiversity through deep time
  - A time component that lists ancient epochs that users can select
  - A tree component that shows the tree of life constructed by the fossil records found in the selected time period, under the selected taxonomy.
  - A map component that shows the shape of the continents in the selected time period, and the locations of the fossil records that constitutes the tree of life in the above component.

# 4 Technical solution

## 4.1 Overview

This section explains the technical solutions to the problems stated in chapter 3. To summarize, the project is divided into data processing in section 4.2, database setup in section 4.3, back-end development in section 4.4 and front-end development in section 4.5, and the workflow of the project follows the same order.

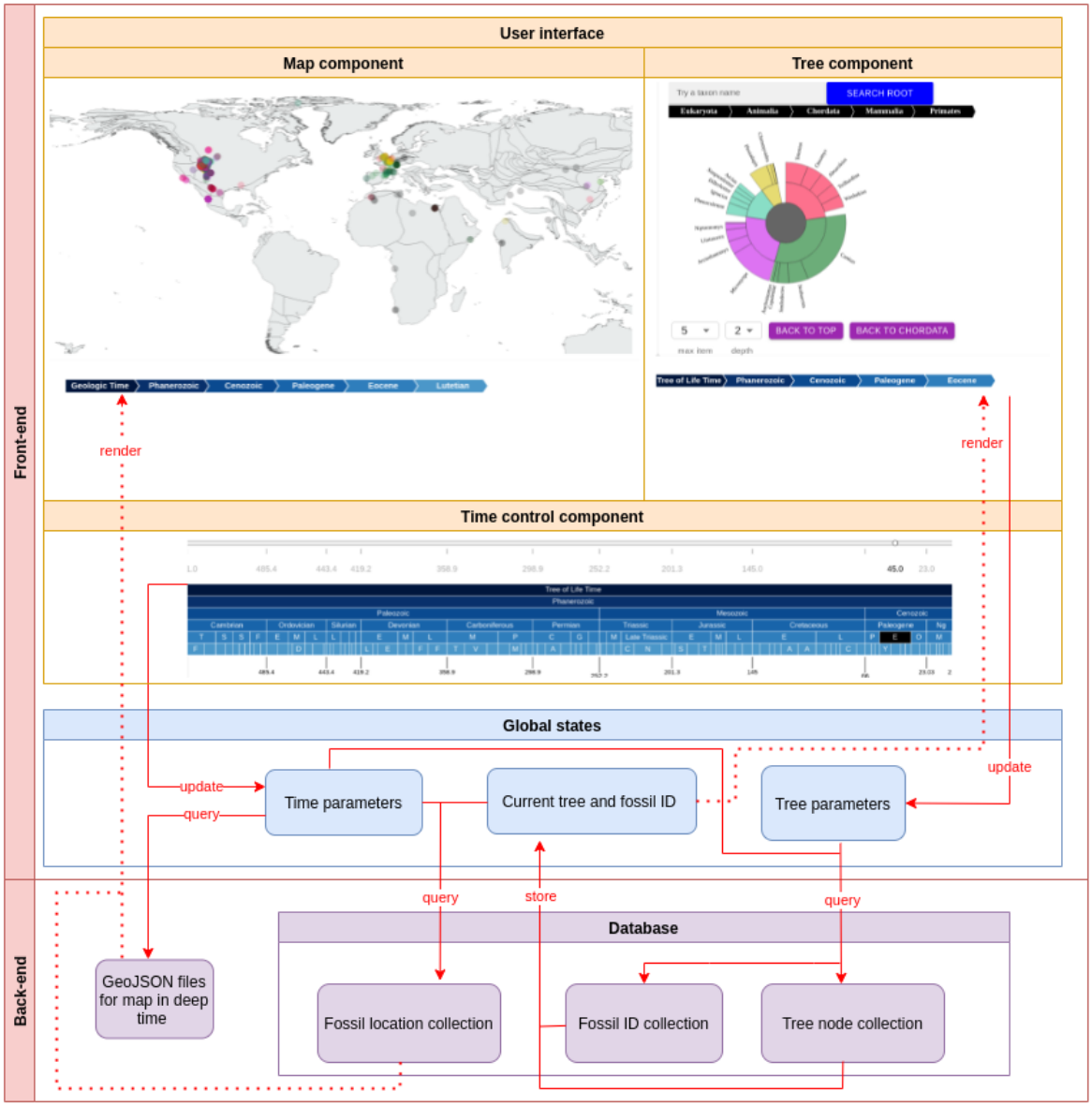



Figure 4.1: An overview of the architecture solution for the project. The user interface consists of a map, a tree and a time table. The global states store tree and time parameters that are customized by users, and the current tree nodes attached with fossil ID sent from database. The back-end resolves queries on tree, fossil locations and map, before sending data back to front-end, where the data is visualized.

Diagram 4.1 proposes a structure of the designed visualization system. The front-end of the web application has a user interface and global states section that stores variables accessible to the components of the user interface. The back-end stores data either in database or as static files, and it responds to the data requests by the front-end, by sending data back to global states section for storage, or directly to the user interface for visualization. Map, tree and time control are the three components that users can view and interact with. The time control panel allows users to select a time period to either view the tree of life, or track the fossil location change due to plate motion. The *time parameters* from global states section listen to user-induced changes in the time control panel, and triggers request of map data, tree data (together with tree parameters stored in global states) or fossil location data (together with fossil ID information stored in global states). The map component receives 1, coastline boundary data in GeoJSON format and 2, fossil location data in database at back-end, to render world map and fossils in a selected time period. The tree component shows the tree of life stored in global states. When users customize the tree view, the tree parameters are updated, and new data on tree nodes with attached fossil IDs is queried from the database. The data is returned to the tree and fossil ID storage in global states, which in turn sends only the tree information to the tree component to be rendered as the new tree.

## 4.2 Data processing

### 4.2.1 Raw data parsing

The data source for fossil records and the tree of life is provided by Paleobiology Database (PBDB). The raw data consists of two csv files that contain 1240202 animal fossil records and 105918 plant fossil records, respectively. Since the two files share the same data attributes, the data is concatenated into one file. Only the attributes that are relevant to this project are parsed, which are listed below.

Extracted attributes in the PBDB dataset:

- *occurrence_no*: a unique ID code for a fossil record
- *accepted_name*: the accepted name of a fossil record
- *accepted_rank*: the taxonomic rank in which the accepted_name belongs to
- *phylum*: the phylum name of the fossil record
- *class*: the class name of the fossil record
- *order*: the order name of the fossil record
- *family*: the family name of the fossil record
- *genus*: the genus name of the fossil record
- *max_ma*: the oldest estimated time of the fossil record, in million years
- *min_ma*: the earliest estimated time of the fossil record, in million years
- *lng*: the longitude of the fossil record's found location
- *lat*: the latitude of the fossil record's found location

In addition, we append a new attribute to the dataset to keep note of which one of the two files a fossil record is originated from:

- *kingdom*: the kingdom name of the fossil record

The kingdom of Animalia and Plantae is assigned to each fossil record, depending on whether it is from the animal or plant file, respectively.

By observing the available attributes in the raw data, we can determine how it can be used for an interactive visualization. The biodiversity of a given time period is represented by 1, a dynamic tree of life formed by the fossil records during that time period, and 2, the locations of the fossils on a world map of that time period. The *occurrence_no* uniquely identifies a fossil record. The *max_ma* and *min_ma* together describe during which time period(s) the organism was alive. The *lat* and *lng* are required to calculate the location of a record during an ancient time period. Most importantly, a tree of life is needed to host each fossil record on a specific node. To generate a tree structure, information on parent-child relationship is essential. This relationship is represented in the names on each of the taxonomic ranks, in the descending order of *domain*, *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, *species*. The raw data lacks information on *domain* and *species*. However, since all records are either animals or plants, they all belong to the common domain of Eukaryota. When the value of *accepted_rank* is species, the species name is found in *accepted_name*. Therefore, the raw data has sufficient information to construct a tree of life with the eight above-mentioned taxonomic ranks as its levels.

The datasets on fossils and tree are created from this parsed raw data. In the next steps, we first prepare the dataset that contains all fossil points, since the tree dataset is created based on it.

### 4.2.2 Fossil point dataset preparation

Since the coordinates of a fossil record depend on the time due to tectonic motion, two datasets are created in regarding to fossil data. This section produces the first dataset *fossil point*, which concerns the taxonomy and the range of age for each record, and sub-section 4.2.4 produces *fossil location*, the second dataset that stores the coordinates of each fossil record on various time periods in history.

The purpose of *fossil point* dataset is to accurately describe a given fossil record in relation to the tree of life, without considering its location at a given time period. The parsed raw data already provides *occurrence_no*, a unique ID code, as well as *max_ma* and *max_ma*, which indicate its range of age in million years. However, one more attribute is required to describe its taxonomy, such that it can be attached to a particular node in the tree of life to be constructed in the next step. This attribute is essentially the unique identifier of a node in the tree of life, and will also serve as the link between the fossil point dataset and the tree dataset. Although it is natural to consider *accepted_name* and *accepted_rank* in the parsed raw data for this purpose, they cannot be used for the following reasons. Firstly, *accepted_name* and *accepted_rank* does not correlate well with the names on the taxonomic ranks, as some *accepted_rank* does not belong to any of the eight ranks that will be used in the tree of life. As a consequence, some fossil records cannot be assigned to the tree, since the *accepted_name* values corresponding to the *accepted_rank* will not be found in the tree. Secondly, the *accepted_rank* and *accepted_name* attributes are not unique. For instance, the genus "Banksia" is both an animal taxonomy and a plant taxonomy. In such cases, fossil records can be associated to a incorrect branch if only these two pieces of information is used. Thirdly, many names in the taxonomic ranks are missing. This leads to ambiguity when attaching a particular fossil to the tree of life, when there exists a rank above the *accepted_rank* with missing names. An example is shown below.

| occurrence_no | accepted_name | accepted_rank | kingdom | phylum | class | order | family | genus | ... |
|---|---|---|---|---|---|---|---|---|---|
| 3285 | Tetradium | genus | Plantae | Rhodophyta | NaN | NaN | NaN | Tetradium | ... |

Table 4.1: An example row of parsed raw data with missing information.

With only the *accepted_name* "Tetradium" on the *accepted_rank* "genus", we are uncertain about which class, order or family to associate this fossil record to under the phylum of "Rhodophyta". Extra knowledge in taxonomy is needed to retrieve the missing names on the path from "Rhodophyta" to "Tetradium" and this challenge is beyond the scope of the current project. The current solution only attaches the fossil record "3285" to the phylum

of "Rhodophyta" and ignores its taxonomy below the phylum level.

To solve these three issues and provide a method for uniquely identifying a fossil record to the tree without ambiguity, the attributes *path_from_root* and *rank* are introduced.

- *path_from_root*: a comma separated string indicating the path from the root Eukaryota to the name on lowest possible taxonomic rank with no missing names in between
- *rank*: the lowest taxonomic rank of *path_from_root*

Since a path begins from the common root Eukaryota and terminates when reaching the lowest rank level "species", or encountering a missing name, the example in Table 4.1 has "Eukaryota, Plantae, Rhodophyta" as *path_from_root* and "phylum" as *rank*. All names in *path_from_root* are guaranteed to be present in the tree, since the names are taken from the same taxonomy information that is used to construct the tree. The path from the common root Eukaryota describes the exact location of a node in the tree, which serves as the node's unique identifier. In addition, the fossils associated to the subtree under a certain taxonomy can be easily obtained by string matching this path using regular expression.

To calculate *path_from_root* and *rank*, the following attributes are used: *domain*, *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, *accepted_name*, *accepted_rank*. Instead of iterating over each row in the parsed raw data, we first aggregate the rows when those values are identical and then perform the iteration, before joining the result with the original data (similar to an "inner join" operation in SQL), in order to avoid repetitive computation.

At each iteration process on the aggregated data, we initialize the *path_from_root* as "Eukaryota" and visit each taxonomic rank from *kingdom* to *genus*, while appending a comma and the name on each rank to the end of the string and updating the current rank. The iteration terminates at the first missing name, when both *path_from_root* and *rank* are those of the missing name's parent. After visiting *genus*, the iteration finishes when the *accepted_rank* is "genus". Otherwise, the *accepted_rank* is "species" or "subspecies". Under this circumstance, we append the value of *accepted_name* to *path_from_root*, update *rank* to be the value of *accepted_rank*, and terminate the iteration. The name in the path after genus can thus be either a species or a subspecies, which is acceptable, since a subspecies name already includes the species name (*i.e.*, the species name of the subspecies "Argopecten circularis impostor" is "Argopecten circularis").

It is important to note that the columns *path_from_root* and *rank* are precursors to the tree dataset, since they represent all nodes with at least one fossil record attached. The unique values of the two columns are used for further processing in sub-section 4.2.3. In the meantime, the aggregated data with *path_from_root* appended are joined with parsed raw data, such that every fossil record has a path. Finally, the *fossil point* dataset is generated, by extracting the following attributes:

- *occurrence_no*
- *path_from_root*
- *max_ma*
- *min_ma*

### 4.2.3 Tree dataset preparation

The purpose of the *tree* dataset is to describe every node on the tree of life that can also be easily searched. The following attributes are computed:

- *path_from_root*: a comma separated string indicating the path from the root Eukaryota to the name on lowest possible taxonomic rank with no missing names in between
- *parent*: the *path_from_root* of its parent
- *name*: the name of the node
- *is_leaf*: a boolean value that indicates if the node is a leaf node
- *max_ma*: the earliest age in million years of the fossils in the subtree of this node
- *min_ma*: the latest age in million years of the fossils in the subtree of this node

For each node, we need a unique identifier, which is provided by *path_from_root* as discussed in the above sub-section 4.2.2. The last *name* of the path is also extracted as an attribute, enabling a more convenient node search when users type a name in the search bar. The node's parent attribute allows searching for all children of a given node. The *is_leaf* attribute indicates directly whether or not a node has children without searching for them. Lastly, *max_ma* and *min_ma* provide a quick way to tell if a node is present during a given time period without searching for fossils attached to its subtree.

To prepare this dataset, we start from the precursor dataset from 4.2.2, which has two columns: *path_from_root* and *rank*. As a reminder, the *path_from_root* are the unique identifiers of all nodes in the tree with at least one fossil record attached. However, there are more nodes to include in the tree. For each entry in the tree represented by its *path_from_root*, we must also include each of the upstream ancestor in the tree. For instance, when the node "a,b,c,d" exists, its ancestors "a,b,c", "a,b" and "a" should also be included in the tree. Therefore, the first step is to add the missing nodes. We iterate every *path_from_root* to add all ancestors along each path to the data. Given a path, the *path_from_root* values of all its ancestors are simply strings sliced from the beginning of the path, until each of its commas. The *rank* value of an ancestor can also be computed by counting the number of commas in the path, as a "domain" has no comma in its path, and a "genus" has six commas. After the iteration, only the unique *path_from_root* and the corresponding *rank* values are kept to avoid duplicates.

The *parent* and *name* values can be easily extracted from a *path_from_root*. The common root "Eukaryota" has no parent, and any other node's parent is represented by the substring of its *path_from_root* sliced until the last comma. Similarly, the *name* of a node is the sub-string beginning from the first character after the last comma in its path. Next, we visit each *path_from_root* once again. If the path is in the set of all *parent* values, then this node is not a leaf node, and its *is_leaf* value is set to be "False", otherwise "True".

Finally, we calculate the *max_ma* and *min_ma* values for each node, using the processed fossil data from 4.2.2, which now contains the path name and the age range information for each fossil record. We first create a reference dataset, by grouping the fossil data by the same *path_from_root* on two attributes: 1, maximum of *max_ma* values among the fossils with identical path and 2, minimum of *min_ma* values. Next, we iterate through this reference dataset. For each path, we extract all its ancestors using the same string splitting method as mentioned above, visit those ancestors in the tree dataset, and 1, update its *max_ma* only if it is less than the current *max_ma* of the path under iteration and 2, update its *min_ma* only if it is greater than the current *min_ma* of the path under iteration.

### 4.2.4 Geospatial transformation of map and fossil data

Apart from the tree, a major component of the visualization is a world map responsive to time selection, on which the locations of the fossils are also shown. Therefore, given a geological timescale, we need to calculate the world maps in all time periods. For each fossil point, its coordinates during those time periods are also calculated and stored in the *fossil_location* dataset.

The timescale data is provided by PBDB. Similar to the tree of life, it is also hierarchical, with five layers in total. The highest layer Phanerozoic ranges from 541 million years ago to present time, and the intervals on the fifth layer often spans only several million years. Nevertheless, plates move constantly, such that the location of a given point at the end of a time interval varies from the location at the start of the interval, even if this interval is in the fifth layer of the timescale. Our solution is to represent the geology of a time period using the middle value of a certain fifth-layer interval. When the user select a time interval in the fifth layer, the coordinates are transformed to that of the interval's middle time point rounded to an integer. When a higher-layer time interval is selected, we first find the fifth-layer in which the middle value of the selected interval falls into, before transforming the coordinates to the integer time point in the middle of that fifth-layer interval.

Both the global plate boundaries of modern day and the model to transform a modern location to an ancient one are provided by Matthew *et al.* [MMZ+16]. In this paper, they proposed a high-resolution model on the movement of plates from late Paleozoic (410 million years ago) to modern day. The boundaries of plates that form the coastlines of a world map, as well as the fossil points belonging to their respective plates, can thus be transformed to a desired time point no earlier than 410 million years ago. In the parsed raw PBDB data from sub-section 4.2.1, we extract the *lat* and *lng* attributes as the modern coordinate of a fossil point, and *occurrence_no* as the ID for each fossil point. For each time point, the model transforms both the modern plate boundaries and the modern locations of each fossil record to their ancient coordinates. Finally, the map data is stored as GeoJSON files, and the ancient locations of each fossil point is stored in the *fossil_location* dataset with the following attributes:

- *occurrence_no*: the ID of a fossil record
- *mya*: the time point as an integer in million years
- *lat*: the latitude of the fossil record in the corresponding time point
- *lng*: the longitude of the fossil record in the corresponding time point

## 4.3 Database setup

The fossil and tree information are stored in a database, from which requested data is sent for visualization according to queries generated from user interactions. The three prepared datasets are uploaded to the database, with slight renaming and reorganizing in the following manner:

- *fossilLocation*: It stores the *fossil location* dataset, which contains the locations at different time points for each fossil record. It has the following fields: *id* (*occurrence_no* from *fossil location*), *mya* (*mya* from *fossil location*), *coordinates* (an array of [*lat*,*lng*] from *fossil location*).
- *fossilPoint*: It stores the *fossil point* dataset, which contains taxonomy and range of age for each fossil record. It has the following fields: *id* (*occurrence_no* from *fossil point*), *maxma* (*max_ma* from *fossil point*), *minma* (*min_ma* from *fossil point*), *pathFromRoot* (*path_from_root* from *fossil point*).
- *TreeNode*: It stores the *tree* dataset, which contains each node in the tree of life. It has the following fields: *pathFromRoot* (*path_from_root* from *tree*), *maxma* (*max_ma* from *tree*), *minma* (*min_ma* from *tree*), *parent* (*parent* from *tree*), *name* (*name* from *tree*), *isLeaf* (*is_leaf* from *tree*).

An entity relationship is shown in Figure 4.2. *FossilLocation* is linked to *FossilPoint* via *id*, the unique identifier for a fossil record. An instance of *FossilPoint* can relate to many instances of *FossilLocation*, because a fossil point has many locations in various time points. An instance of *FossilLocation* relates to one and only one fossil record in *FossilPoint*. An instance of *FossilPoint* is uniquely identified by its *id*. *FossilPoint* and *TreeNode* are linked via *pathFromRoot*. One fossil record is associated to one and only one node in *TreeNode*, however, a tree node can have zero or many fossil records. Indeed, a tree node of a high taxonomic rank has no fossil attached to

it, when all fossils under this taxonomy are identified with a lower rank, thus are attached to its descendants. For instance, the root node "Eukaryota" has no fossil attachment, since for any given fossil, we have at least identified it as an animal or a plant. Finally, an instance of *TreeNode* represents a certain node in tree of life, therefore is uniquely identified by its *pathFromRoot*.

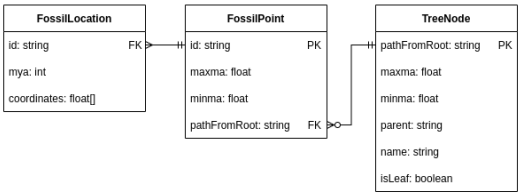



Figure 4.2: An entity relationship diagram of the database.

## 4.4 Back-end query processing

The back-end is responsible for processing the data queries requested by user actions, and sending the result to the front-end. How the queries are resolved is addressed in this sub-section. There are two main queries to carry out: 1, given a time range and tree parameters, obtain a tree of life with fossil IDs attached to each node, and 2, obtain the locations of the fossil records given their IDs and a time point.

The first query returns a fossil-attached tree under a requested root node. The following parameters are passed to the database:

- *pathFromRoot*: the unique identifier of the root node
- *name*: the root node's name
- *maxma*: the start of the selected time period
- *minma*: the end of the selected time period
- *maxElement*: the maximum number of siblings to retrieve
- *depth*: the number of tree layers to retrieve

The resulted tree should be rooted by the requested node, with a depth no greater than the parameter *depth*. The number of children under each node is no greater than number of *maxElement*. The IDs of fossils that exist during the time period is also attached to each node. The returned tree has the following attributes:

- *pathFromRoot*: the unique identifier of the node
- *name*: the node's name
- *parent*: the *pathFromRoot* value of its parent
- *isLeaf*: a boolean value indicating if the node is a leaf node in the returned tree
- *fossils*: the IDs of fossil records existing during the requested time period that either are attached to this node (if the node is not a lead node) or belong to the node's subtree (if the node is a lead node)

To retrieve the tree, we first find the requested root node by searching for its name in *TreeNode*. The *name* is passed from user input. Since the first letter of all names in *TreeNode* are capitalized, we capitalize the first letter in *name* when it is not, before performing the search. The *parent* value of the root node is set to null, since although it may have a parent in the complete tree of life, it is the root with respect to the tree to be returned.

The fossil IDs attached to the root node is also appended. We then recursively search for its children and append the nodes to the result, until reaching the requested depth, or encountering nodes with no children left. When processing the children of a certain node during the recursion, we check the *isLeaf* value in the following way. If this node has no children, then it is certain a leaf. However, if the node is on the maximum depth, then it is a leaf node with respect to the tree under construction, whether or not it has children in the complete tree of life. When it is a leaf node, we search for the IDs of all fossils belonging to its subtree. This can be easily performed with string matching the *pathFromRoot* of the fossil records using regular expression, since all desired fossils' paths beigin with the *pathFromRoot* of the current node. When the node is not a leaf node, we only attach fossils that are strictly associated with this node, by exact string matching of *pathFromRoot* between this node and the fossil records. To improve efficiency, the sibling nodes are filtered before continuing to the next recursion level. A count value is assigned to each sibling node, which is the number of fossil records in *FossilPoint* belonging to its subtree that is present during the time period. Then the siblings are ranked in descending order, and the top *maxElement* number of siblings are kept.

The second query receives a time point of interest and an array of nodes in the current tree, which is a result of the first query. However, only the relevant fields are passed:

- *name*: the node's name
- *fossils*: the IDs of fossils attached to the node (if it is a lead node) or its subtree (if it is not a leaf node)

From *FossilLocation*, we retrieve the fossils' coordinates by their IDs and time point, and organize the result in the following manner:

- *id*: the fossil record's ID
- *coordinate*: the fossil location at the time point, in the form of [latitude, longitude]
- *name*: the name to which the fossil record is associated in the current tree
- *mya*: the requested time point

## 4.5 Front-end development

This section introduces the design solution for the front-end, which includes a global states section that locally stores variables to which different components in the user interface have access, and the user interface where viewers interactively explore the visualized data. The user interface is in turn divided into a time control component, a map component and a tree component.

### 4.5.1 The global states

The global states serve as the intermediate agent between the user interface and the back-end, and between the different components of the user interface. As shown in Figure 4.1, the information on time parameters, tree parameters and current tree with attached fossils are stored in global states.

*Time parameters*. Time parameters include a time period and a time point. Upon a time-related request from the user interface, certain values of time parameters are updated, and new data queries are triggered. When only the time point is changed, new fossil location data is queried using time point and the fossil IDs stored in *Current tree and fossil ID*, and new map data on the requested time point is loaded. When time period value is changed, data on a new tree and fossil IDs that are attached to the tree's nodes are queried, before storing the new result by updating *Current tree and fossil ID*.

*Tree parameters*. Tree parameters keep the record of how the tree is customized by the users. It includes the requested depth, the maximal number of children to retrieve under a given node, and the name of the node to search for. When any of the parameters are changed as a result of user actions, combining with the time period information in *time parameters*, a new data query is sent to compute a customized tree of interest formed by fossils found in the time period, and both the tree nodes and the attached fossil IDs are stored in *Current tree and fossil ID*. Lastly, in the user interface, when a user hover on a node in the tree graph, or a fossil point on the map, the identity of the node is recorded as a parameter, in order for both tree and map to be aware of, which is required for the communication between the two components.

*Current tree and fossil ID*. This global state variable stores information on current tree nodes with fossil IDs attached. It mainly serves two purposes. When users are interested in viewing how the same fossil points move as a result of plate movement through time, they change the time point value in *time parameters*, triggering a data query for new fossil locations with the pre-stored fossil ID values. Since under this circumstance, the fossil IDs remain constant, it is more efficient to store them locally. In addition, the colors of nodes shown in the tree component is consistent with the colors of their attached fossil points shown in the map component. As a result, upon receiving new tree and fossil ID data from the database, instead of sending directly to the tree component in the user interface for rendering, we compute the colors before storing in *Current tree and fossil ID*. Thereby, the color values are calculated for only once, and are shared by both map and tree component when they render fossil points and tree respectively. The coloring of the tree node is determined as follows. The current root node is in black. Its descendants are assigned with one of the five distinctive colors. The assignment initiates from the children of the root node, by visiting each node in a breath-first-search manner. If all five colors are used upon encountering a node, if the node is the children of the current root, *i.e.*, we are still on the second layer of depth in the tree, then the last of the five color is assigned to this node; otherwise, we assign the same color as its parent, which must already have one of the five colors, as the parent has been visited and assigned with a color in the breath-first-search process.

### 4.5.2 The user interface: time control component

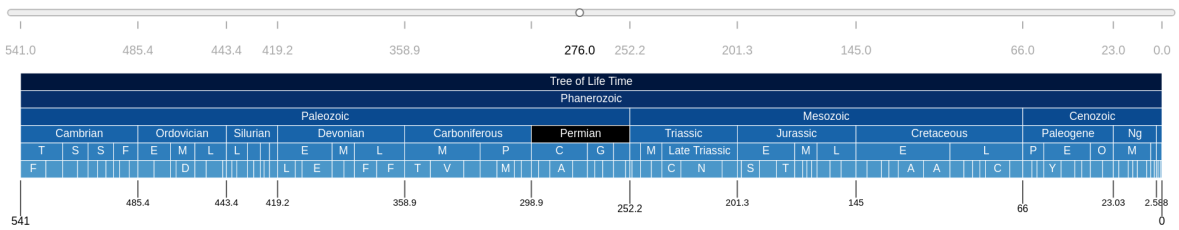
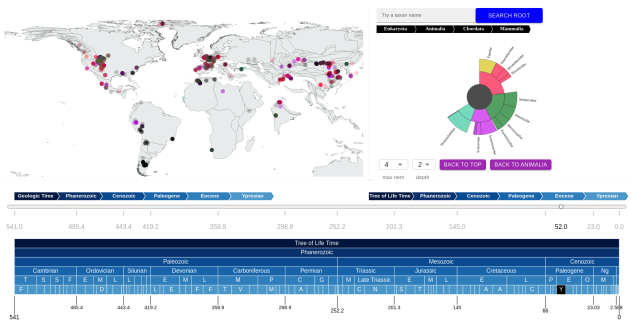


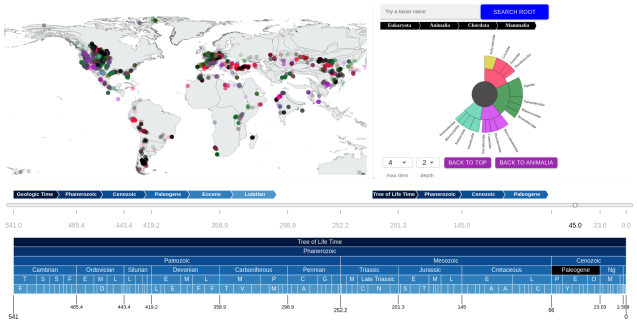

Figure 4.3: An overview of the time control component. A timescale table displays all time periods of the Phanerozoic that are organized hierarchically in five layers. When selecting a time period, *i.e.* "Permian", its cell is marked in black. Some tick values are shown below the table, marking the start and end years of all time periods on the third layer, in the unit of million years. There is a scroll bar on the top that allows users to drag, in order to view the world map and the fossil locations of a later time than the selected time period.

The time control component shown in Figure 4.3 sits at the bottom of the user interface. It allows users to control the map and tree components by either clicking in the timescale table, or dragging the scroll bar. The first functionality is shown in Figure 4.4. Users can pick a time period of interest to view the tree and fossil points during that period by clicking a cell in the time table. Only a time period on or below the third layer can be selected, in order to avoid requesting fossil data spanning hundreds of millions of years at a time, which may exceed the size of the heap memory. When a time period is selected, a time point at which the world map and the fossil locations will be queried is also calculated. This time point is intended for representing the geology during the time period. Due to the hierarchical nature of timescale, a certain time point must belong to a time period in

the fifth layer. We intend to show the name of this time period, since it is of the highest detail. When a fifth-layer time period is selected, the time point is simply the middle value between the start and end year, rounded up to an integer (Figure 4.4(a)). Otherwise, the middle value of the time period is calculated, and we search for the fifth-layer time period where this calculated middle value falls into. Then, the middle integer value of that fifth-layer time period is used as the time point (Figure 4.4(b)). The second functionality is shown in Figure 4.5. Users can pick a time point that is later than the selected time period to view how fossil locations are shifted with the continents, by dragging the scroll bar on the top. Those time points are the middle values of all fifth-layer time periods later than the current time point, rounded up to integer.



(a) The time period "Phanerozoic -> Cenozoic -> Paleogene -> Eocene -> Ypresian" is selected. Since this is a time period in the lowest layer, the timelines below both map and tree show the name of this time period. The map in fact shows the continents and fossil locations at the middle point of this period.



(b) The time period "Phanerozoic -> Cenozoic -> Paleogene" is selected. This time period is not in the lowest layer, but its middle value falls into "Phanerozoic -> Cenozoic -> Paleogene -> Eocene -> Lutetian", a time period on the lowest fifth layer. Therefore, in the map component, Lutetian is the time period that represents for the selected Paleogene, and its name is shown beneath the map on the left. We then take the middle value of Lutetian as the time point at which the continents were shaped and the fossils were located.

Figure 4.4: The timescale functionality of the time control panel. Users can select a period in the timescale to view the map and tree during this time period. The tree of life is always in accordance to the selected time period, whereas when the time period is not on the lowest layer in the timescale, the map component picks the middle value of a time period on the lowest layer in which the middle value of the select time period falls.

The entire Phanerozoic spans over 500 million years, whereas certain time periods, especially those of lower layers, are only hundredth of that length. Without dynamic scaling of the time table, it is difficult to have a clear view or an easy accessibility of the shorter time periods. A solution is demonstrated in Figure 4.6. When hovering on a time period on the third degree or lower, a zooming effect takes place, where all time periods from the third layer and below are elongated, such that the third layer cell occupies 90% length of its immediate ancestor. As a result, labels indicating the start and end year of the shortest time periods on the fifth layer can also be shown. To further highlight the time period, when hovering on a time period, the opacity of all other time periods that are not in the path to the common ancestor "Phanerozoic" are decreased, and a small pop-up note also shows its name as a path from the common ancestor.

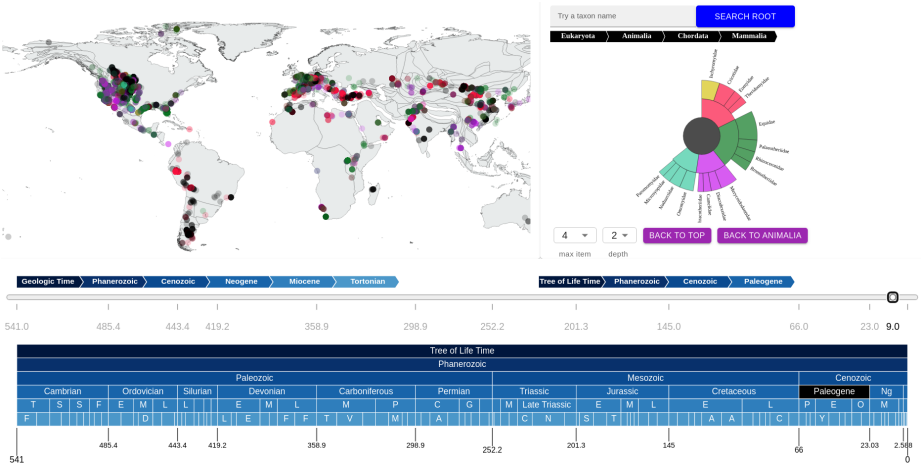



Figure 4.5: The scroll bar functionality of the time control panel. Users can drag the scroll bar to a later time point than the currently selected time period, in order to track how the continents and the fossil points move due to plate motion. From the time table and the breadcrumb beneath the tree, we can see that the tree of life and the attached fossils are still with respect to the selected "Phanerozoic -> Cenozoic -> Paleogene". However, the number of the scroll bar and the breadcrumb beneath the map imply that the scroll bar has brought the world map to a more recent time point at 9 million years ago, which is during "Phanerozoic -> Cenozoic -> Neogene -> Miocene -> Tortonian".

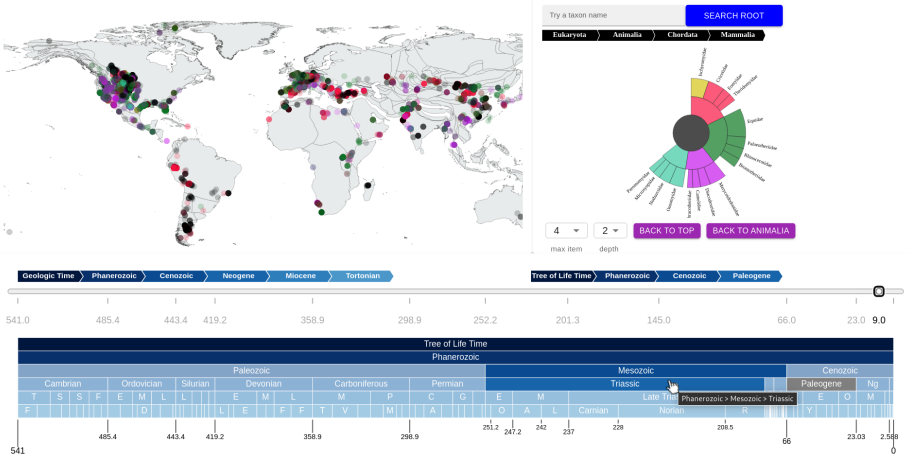



Figure 4.6: The zooming and highlighting effect of the timescale upon hovering. When users hover on a particular time period, a pop-up note with its name shows next to the pointer, and all other time periods that are not in its path to Phanerozoic are decreased in opacity. The selected "Triassic" belongs to a period on the third layer or below, as a result, a zooming effect takes place. All time periods at or below "Triassic" are proportionally elongated such that "Triassic" takes up 90% the length of its parent "Mesozoic". Due to this expansion, the tick values can now show all start and end years on the lowest layers.

### 4.5.3 The user interface: tree component


Figure 4.7: The tree component of the user interface.

The tree component is on the right of the user interface. It shows the current tree of life, while providing options to customize the tree. A search bar is located on the top of the tree component, where users can search for a name. Below the name, a line of breadcrumb indicates the location of the current node in the complete tree of life, by showing all of its ancestors organized in a path from the common root. Each ancestor can be visited by clicking. Since the ancestor nodes are upstream of the current node, they are colored in black, to distinguish from the descendants of the node, which are in various colors. The tree is rendered as a sunburst graph from the global state variable *Current tree and fossils* addressed in sub-section 4.5.1, which contains information on a node's name, color and fossils.

To demonstrate the prevalence of a certain taxonomy during the selected time period, the angle of each node's arc reflects the number of fossils attached to its subtree. The current node under inspection is the root of the tree being displayed, as a consequence, it is a full circle, since it includes all fossils in its subtree. The arc angle of sibling nodes is proportional to the ratio between the number of fossils in their respective subtree, and the number of fossils included in their parent's subtree. To calculate the number of fossils under a given node's subtree, we

recursively sum up the fossil counts from leaf nodes to the common root node, by the number of fossils attached to each node. Note that when performing the sum up, if the node is a leaf node, we sum up by all fossils under its subtree, however, when the node is not a leaf node, we sum up by the fossils that are attached to it. This is because if we still sum up by the fossils belonging to its subtree, we are double counting the fossils, as each of its children is already summed up by the fossils under their respective subtree. *Current tree and fossils* conveniently stores fossils attached to node for non-leaf nodes, and fossils associated to the node's subtree for leaf nodes. Therefore, when performing the sum up at the front-end, no further processing is needed to distinguish if a fossil count is with respect to only the node, or its subtree.

To highlight a certain node of interest, we provide the following visual aids. First, a second line of breadcrumb indicates its position from the current root node. Second, similar to the time control component highlight effect, the opacity of other nodes are decreased, and a pop-up note informs the user on the node's name. When a node's name is too long or a node's arc is too narrow to display its name on the tree, the pop-up note and the dynamic breadcrumb provide an opportunity to view the name upon hovering. Finally, the hovering effect is communicated to the map component, such that only the fossil points attached to the hovered node are shown and in full opacity, to inform the users where their interested taxonomy inhabited on the map.

To enable users to customize and navigate the tree, we designed a small control panel below the tree graph. The *max item* adjusts the maximal number of elements to display under each node, and the depth adjusts the tree's depth. If the current node is not the common root "Eukaryota", the *Back to top* button directs the user to it. Clicking on the "Eukaryota" in the breadcrumb above the tree graph provides the same functionality. If the distance from the current root to the common root is greater than the current *depth*, a second button redirects the users to upstream of the current node that is it's *depth*'th ancestor.

Finally, the tree's time period is reflected below the control panel. This is to remind the viewers that the tree of life is subjected to this particular time period, since the map they are viewing might be of a different time period.

### 4.5.4 The user interface: map component

The map component is on the left of the user interface. It shows the world map with fossil points at the select time point. The geological data on coastlines are provided from static files in the back-end, and the fossil coordinates are rendered directly from database's *FossilLocation*, whereas their pre-computed colors are loaded from front-end in *Current tree and fossils* discussed in 4.5.1. The zooming and panning functionality enables users to view regions of interest in greater detail, and when hovering on a fossil point, the tree component shows which node it attaches to, by highlighting the node and providing a second breadcrumb that indicates its position, in a similar way as hovering on a node on the tree. Lastly, often fossil points are of very close proximity, or even have identical locations, leading to the fact that the points can easily be on top of each other. As a solution, we decreased the opacity of the fossil points in order to reflect all fossil records' presence — a point of higher visibility implies that there are multiple records found at or close to the location.

# 5 Implementation

## 5.1 Code structure

## 5.2 Data processing stacks

## 5.3 Web development stacks

# 6 Results

## 6.1 Data analysis

### 6.1.1 fossil data

### 6.1.2 tree data

## 6.2 Data visualization case studies

# 7 Conclusion

## 7.1 Limitations and Future Work

# Acknowledgements

# Bibliography

[MMZ+16] Kara J Matthews, Kayla T Maloney, Sabin Zahirovic, Simon E Williams, Maria Seton, and R Dietmar Mueller. Global plate boundary evolution and kinematics since the late paleozoic. *Global and Planetary Change*, 146:226–250, 2016.

# List of Figures

# List of Tables