Chapter 8

Data Model Design

In order for me to design a complete database model for each of the technologies, an initial investigation into the specific dataset values was required discussed in section 8.1. Once this was complete I followed the same database design process for each indexing solution which is discussed in section 8.2.

8.1 Dataset Values

The EMAGE data which was supplied was made up of 4 tab separated files. Each file contained information pertaining a certain aspect of the dataset; Annotations, Publications, Submissions and Results. On initial review each of the files contained similar, repetitive metadata values. Thus creating a level of noise which would not be add anything to the project in terms of analysis and evaluation. Therefore I undertook an initial cleansing of the data before implementing the database design.

This cleansing process consisted of loading the data into Google Refine (GR) and manually manipulating the data using the filtering and editing tools the package provides. Using this tool allowed me to identify any erroneous rows of data which would affect the integrity of the dataset. In each file there was at most 5 rows of data which were either blank or inconsistent with the convention of the rest of the file. I decided to remove these rows as there inclusion in the file was unnecessary.

Another irregularity found in the *Publications* data file was the characters used in the title and author fields. There were over 600 rows of data which contained a non-ascii character. These rows would be rejected when importing into the databases as by default I decided to

apply a UTF-8 character encoding to each indexing solution. Despite these values only contributing to around 5% of the file, the issue needed to be addressed. To do this I devised a regular expression which would identify and subsequently remove any of these characters.

Using a software tool such as GR enabled me to manipulate and cleanse the data in such a way that I would be able to load it successfully into the databases. Despite accomplishing the cleansing successfully, from the challenges I faced, I identified some problematic circumstances which, despite not being as prevalent using the EMAGE data, may affect other big data sets.

The maximum number of rows uploaded into GR was around 150,000. The EMAGE dataset is relatively small in comparison to large scale data collation, however the volume of data was a factor in my decision to use a software tool in an ETL workflow as opposed to rolling my own scripting solution. It is important when choosing a methodology or tool to enhance the veracity of a dataset that the volume of data is taken into consideration. GR is a Java application that utilises the Java Virtual Machine (JVM) and therefore it is integral to allocate enough memory to handle processing large files and thus avoid Java heap space errors. The GR developers suggest that a typical best practice is "start with no more than 50% of your available physical memory, since certain OS's will utilize up to 1 Gig or more of physical RAM." [**?**]. While using this software solution was sufficient for the data in this project, should the dataset be of a greater scale, a more robust and resilient system would need to be considered.

As discussed in section 1.1.1 a major challenge in data collection and manipulation is ensuring the veracity of your data. A leading contributor to this challenge is human error. It is a fact of life that humans are error prone and can often make mistakes, therefore where possible the minimal amount of manual handling of a dataset is key. An example of an issue which can arise from this may be as simple as date formatting changing over time. The data may initially be input in a UK standard date format of DD-MM-YYYY by one person and then stored in a US standard data format of MM-DD-YYYY by another person. A simple example, however one which can have serious repercussions on the validity of a dataset. The cleansing of the EMAGE dataset relied on my knowledge of the data and any obvious flaws such as blank values where a value was required. While the data provided was reliable and generally healthy; a richer more granular dataset may require a more rigorous method of validation. One way to do this would be to implement a software script which takes a subset of the data,

defines a format, and restructures the remaining data in the dataset accordingly.

8.2 Database Design

In order for me to develop multiple, reliable, data models which accurately represent the dataset, I created a database diagram for each indexing solution which effectively illustrates the relationship between the data entities. The order in which I decided to create the database model designs was based around my previous experience of using the database systems. This experience range was from a competent level to the complete unknown. As a result I decided the first data model I would develop would be for the relational database management system MySQL. The reason why I done this was because it is the database system which I have most experience in using and implementing. The next system I developed was MongoDB, followed by Neo4j and finally Apache Cassandra. Each implementation presented a different challenge all of which will be discussed below.

MySQL

As discussed in section 4.3.4 MySQL is a relational database management system which stores and represents structured data through entity tables and relationships. There are a number of variations in which the design of the MySQL database could be modelled for the EMAGE data. Figure 8.1 is an entity-relationship (ER) diagram which illustrates the implementation of my MySQL normalised database design. Normalisation in database design is a process by which an existing schema is modified to bring its component tables into compliance through a series of progressive normal forms. It aids in better, faster, stronger searches as it entails fewer entities to scan in comparison with the earlier searches based on mixed entities. Data integrity is improved through database normalisation as it splits all the data into individual entities yet building strong linkages with the related data. A description of the tables is below the diagram.

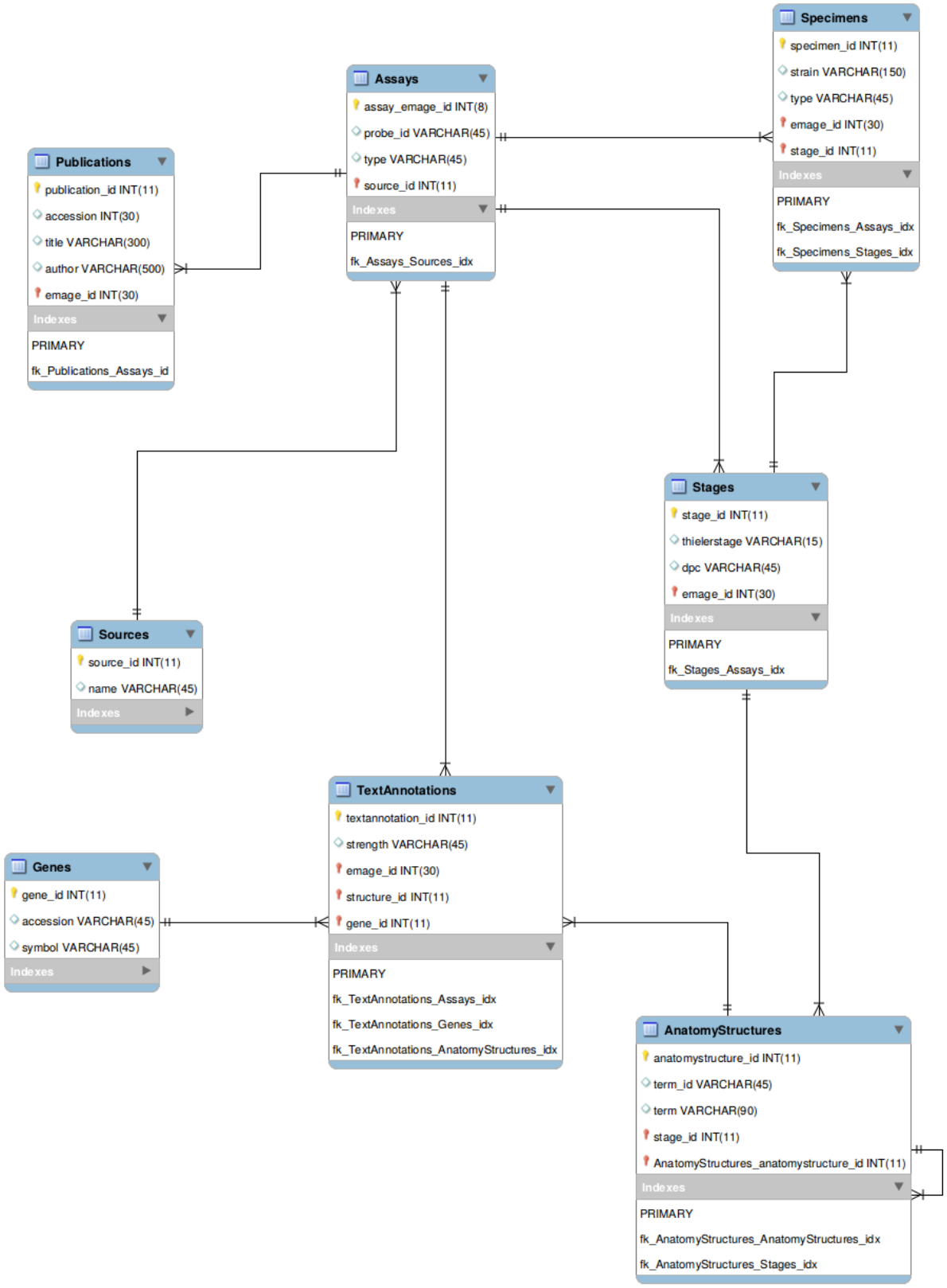

Figure 8.1: MySQL ER table diagram

- **AnatomyStructures** : This table contains the EMAPA ID, and the term which refers to the part of the anatomy.
  - Many to One relationship with the Stages table as one anatomy structure can have the same stage.
  - Many to One relationship with itself on the ID as one structure can have many parts.
- **Assays** : This table contains the EMAGE ID number, the ID of the probe and the type of assay. The *type* field refers to whether the assay is *in situ* or *part of*.
  - Many to One relationship with the Sources table. While one assay can only have one source, many assays can have the same source.
- **Genes** : This table contains the accession number and symbol of each gene.
  - The Genes table does not reference any other table.
- **Publications** : This table contains the accession, title and every author of each assay publication.
  - Many to One relationship with the Assays table as there can be many publications for one assay.
- **Sources** : This table contains the source of the assay.
  - The Sources table does not reference any other table.
- **Specimens** : This table contains the ID, strain and type of each specimen. The *type* field refers to whether the assay is a section, wholemount, sectioned wholemount or unknown.
  - One to One relationship with the Assays table as one assay has one specimen.
  - Many to One relationship with the Stages table as many specimens can have the same Stage.
- **Stages** : This table contains the theiler stage and number of days post conception (dpc) of each assay, specimen and anatomy.

  - Many to One relationship with the Assays table as one assay has can have multiple stages.

- **TextAnnotations** : This table contains the structure and strength of each assay.
  - Many to One relationship with the Assays table as one assay can have multiple text annotations.
  - Many to One relationship with the Genes table as one text annotation can have multiple genes.
  - One to One relationship with the AnatomyStructures table as one text annotation can have one structure.

MongoDB

As discussed in section 4.3.1, MongoDB is a homogeneous, schema-less, NoSQL document store database. There are no formal relations between the data which makes modelling the database a little more challenging; especially with data which is so closely bound as the EMAGE dataset.

Data in MongoDB sits in *collections*, a grouping of documents which are stored on a database. A collection exists within a single database and is the equivalent of an RDBMS table. Documents within a collection can have different fields and typically, all documents in a collection have a similar or related purpose.

The MongoDB implementation was the second prototype data model I created for this project. I followed the same structural process which I had undertaken for the previous MySQL data model. When creating a MongoDB data model there are a number of factors and considerations which need to be identified before starting the formal implementation. Firstly the biggest decision I deliberated over was how should the data be connected. As there were a few options I decided to explore all of them to fully comprehend the pros/cons of each.

My initial design was based around using multiple collections to store the various aspects of the data. The design followed the MySQL model, with were 4 collections; Assays, Text Annotations, Anatomy Structures and Genes. To connect the data and bind the values required an additional manually developed ID field for every document. While this was not a complex task, it was one which I felt was unnecessary and added extra unwanted noise to the data. Using this option would have also incurred more overhead when writing the queries for

the database as one would firstly have to connect the data (similar to a RDBMS ”join) and then include a further query.

After some deliberation I concluded that the best way to implement the data model would be to have all of the data in the one collection. Figure 8.2 illustrates an example document in the developed MongoDB data model. The diagram should be read as follows:

- **id** : This is the EMAGE id of each assay and is the value which binds all of the data together. The id value has been manually configured to correspond with the EMAGE value.
- **specimen** : The specimen value is an array of size 2 which holds data regarding the strain and type of the assay.
- **probe id** : This value is the id of the probe accession.
- **assay source** : The source in which the assay has been retrieved from.
- **assay type** : The type of assay which is being analysed. By type I am referring to whether the assay is *in situ* or otherwise.
- **stage** : An array containing the information regarding the stage of the assay; Theiler stage and DPC.
- **publication** : An array containing all publication information regarding that specific assay; id, title, author.
- **text annotation** : The text annotation array is the grouping of strength, anatomy structure and gene of an assay. An anatomy structure has a term id and the name of the term and a gene has the symbol and id.

```
{
  "_id":"6",
  "specimen":
   {
     "specimen_strain":"Swiss Webster",
     "specimen_type":"wholemount"
  },
  "probe_id":"MGI:1334951",
  "assay_source":"emage",
  "assay_type":"in situ",
  "stage":
   {
     "theiler_stage":"11",
     "User_Stage":"7.5 dpc"
  },
  "publication":
   {
     "publication_id" : "PMID:1409588",
     "publication_author" : "Tanaka A, Miyamoto K, Minamino N, Takeda M, Sato
B, Matsuo H, Matsumoto K",
     "publication_title" : "Cloning and characterization of an androgen-induced
growth factor essential for the androgen-dependent growth of mouse mammary
carcinoma cells."
  },
  "textannotation":
   [
     {
       "anatomystructure":
       {
         "term_id":"16107",
         "term":"allantois"
       },
       "strength":"strong",
       "gene":
       {
         "symbol":"Fgf8",
         "gene_id":"MGI:99604"
       }
     }
  ]
}
```


Figure 8.2: Example MongoDB document diagram