|  |
| --- |
| TRACK Team Project Report |
| Distributed Database Query Engine Service (DDQES) |
| Frederic Colin [2012400565] Henry Loharja [2012280102] Ng Yi Ying [2012280098] |

Table of Content

[Introduction 3](#_Toc358801600)

[Project Scope 4](#_Toc358801601)

[System Environments 5](#_Toc358801602)

[Web User Interface 6](#_Toc358801603)

[2-Level High Availability (HA) Web Services 7](#_Toc358801604)

[High Availability Web Service (HAWS) – Level 1 (L1) 8](#_Toc358801605)

[High Availability Web Service (HAWS) – Level 2 (L2) 8](#_Toc358801606)

[High Availability Plan 9](#_Toc358801607)

[Database System and Engine 9](#_Toc358801608)

[Global Table Structure 9](#_Toc358801609)

[Table Fragmentation 10](#_Toc358801610)

[Database System and Engines Communications 11](#_Toc358801611)

[Local Systems 11](#_Toc358801612)

[Master System 11](#_Toc358801613)

[Query Processing 11](#_Toc358801614)

[Communication 12](#_Toc358801615)

[Master Site Strategy and Dictionary Storage 12](#_Toc358801616)

[Summary 13](#_Toc358801617)

[Project Implementation 13](#_Toc358801618)

[Web User Interface 13](#_Toc358801619)

[2 –Level High Availability Web Service 15](#_Toc358801620)

[High Availability Web Service – Level 1 (HAWS-L1) 16](#_Toc358801621)

[Data Decomposition 16](#_Toc358801622)

[Query Tree Optimization and Optimized Strategy Plan Generation 17](#_Toc358801623)

[High Availability Web Service – Level 2 (HAWS-L2) 17](#_Toc358801624)

[Hibernate Data Retrievals and Communication 17](#_Toc358801625)

[Packages and Classes 17](#_Toc358801626)

[Project Timelines 17](#_Toc358801627)

[Workload Matrix 18](#_Toc358801628)

[Project Challenges 19](#_Toc358801629)

[Future Enhancement 20](#_Toc358801630)

[Conclusion 20](#_Toc358801631)

[Appendices 22](#_Toc358801632)

[Appendix I – Meeting Minutes 1 22](#_Toc358801633)

[Appendix II – Meeting Minutes 2 23](#_Toc358801634)

[Appendix III – Meeting Minutes 3 24](#_Toc358801635)

[Appendix IV – Meeting Minutes 4 25](#_Toc358801636)

[Appendix V – Meeting Minutes 5 26](#_Toc358801637)

[Appendix VI – Meeting Minutes 6 27](#_Toc358801638)

[Appendix VII – Meeting Minutes 7 28](#_Toc358801639)

[Appendix VIII – Meeting Minutes 8 29](#_Toc358801640)

# Introduction

Distributed Database Query Engine Service (DDQES) is a project system which is assigned to promote a better and deep insight on the knowledge of distributed database system through a hands-on software design and implementation experiment. In term of scope, the project system is covering a distributed system environment which involved four distributed sites and four database engines. In short, the entire distributed environment is designed to be capable of handling distributed query processing that involved distributed query requests, query analysis for decomposition and localization. With the concern of data communication cost, query optimization is within the considerations of the project scope for effective database retrievals and optimized returned result set. In addition, to ease crossed platforms data communication; DDQES is developed with the functionality of being published as a web service to support remote service call from all available sites for further processing, which is described with Web Service Description Language (WSDL).

Simply to add in additional challenging aspect into the project system, DDQES is implemented in such a way which is capable in handling data communication among four physical computer systems to simulate real world distributed system environment; which overall uses three heterogeneous database engines; including MySQL, SQLite and PostGreSQL. High availability is in fact a practical concern in highly distributed environment; to cater for such a need a high-availability plan was generated where instead of relying on single centralized entry point for query processing; the number of web services increases with the number of sites. DDQES is providing its service through a two-level distributed web services; first level Distributed Query Processing Application (DQPA) which process user queries to generate an optimized query processing strategy while the second level executes the data retrievals. Hence, failure of one web service may not be visible to the frontend user as the query processing will be still working backend, provided the offline data is not possible to be read and display.

The entire project is handled by a group of three team members (with TRACK as the team name), namely Frederic Colin, Henry Loharja and Ng Yi Ying as the team leader. Throughout the entire final documenting report, there are mainly five main sections which covered: 1. Project introduction, 2. Project design, 3. Project implementation and testing; 4. Project timelines and workloads distributions and 5. Future enhancements. The structures of documentation are presenting the team’s compact efforts starting from the analysis and design of DDQES application to the final implementations with detailed considerations of communication efforts, costs and interoperability in processing users’ SQL queries which can be submitted through web interface or directly to the published web services. In the following section, the project descriptions which including the project scopes are further elaborated.

## Project Scope

Distributed Database Query Engine Service (DDQES) as mentioned is designed to handle multiple remote calls from involved sites in the predefined system network architecture; accomplishing by utilizing the published web services. It is within the project scope to include four physical sites which involved four distributed databases which consisted of four relational tables: Publisher, Book, Customer and Orders. To also consider the interfacing aspect of the application to accept user queries, web user interface is also included as part of the project scope. The final submitting DDQES is equipped to provide:

* Web User Interface: Upon publishing, the user can access the web-based user interface which allows the submission of SQL query request to the application for further processing. Upon completion of execution, the result and analysis data will be presented on the web user interface for user viewing.
* Structure Query Language (SQL) Support: Capable to handle SELECT statements which allows the users to express simple predicates conditions connected by logical operator, such as AND, NOT and OR.
* High Availability (HA) Web Service - Level 1 (L1): Provide web-based access point to conduct query request processing from various requesting sites through remote call. Mainly to promote a high availability rate, the HA Web Service-L1 is scoped to have the capability of redirecting user request to another online HA Web Service-L1 (based on the service dictionary) to carry out its responsible tasks if the local web service is down. Upon the HA Web Service - L1 is being called by the site, an instance of Distributed Query Processing Application (DQPA) is created to analyze the submitted query, further generate query retrieval plan after query decomposition and finally data location.
* High Availability (HA) Web Service - Level 2 (L2): Closely related to HA Web Service -L1 where it is responsible of executing optimized data retrieval strategy by requesting and gathering intended data; further merging the result set for return to request site. [Can further mentioned the name of this part after integration] Specifically mentioned on the result of the query, the output will include the display of the following:
  + The size of query result set
  + The optimized query tree
  + The time cost of query
  + The communication cost of query
* Query Optimization: To reduce the data communication cost, various optimizations are expected to be included in the project design and implementations, including:
  + Query tree optimization and result set merging
  + Query tree reduction by fragmentation

In term of project deliverables, Distributed Database Query Engine Service (DDQES) is delivered as expected with the mentioned scope and functionalities, together with this entire final report and system operation manual in the final submission. In the following section, the system environment for DDQES is documented which included the details of database system and engines, global tables structure and DDQES development and implementation environment.

# System Environments

Under this section, it aimed to document the descriptions of relevant system environment of Distributed Database Query Engine Service (DDQES). Mainly to ease the integration effort that supports crossed-platform development of having multiple developers (three team members) for the project, Java is the identified programming language for building DDQES. Another reason for making such a decision on the selected programming language is due to its capability to create web service through the use of JAX-WS Web Service which can closely work with Java Server Page (JSP) web interfaces which fulfilled the project development and implementation needs. Hence, it is reasonable to visualize that all four sites within the DDQES are running the same Java application instance on their physical location for handling all necessary operations.

To simulate the real world environment and to own a closer estimations for the communication cost, four physical personal computers will be used to each represents a site rather than using a single virtual computer for demonstration. Besides, three heterogeneous database systems are being used where two sites may have MySQL while the rest will be running SQLite and PostgreSQL as their supporting database. Regarding the data communication protocol, HTTP communication is being used which relies on the personal computer open ports. For further elaborations, the overall Distributed Database Query Engine Service (DDQES) architecture paradigm can be viewed as the following:

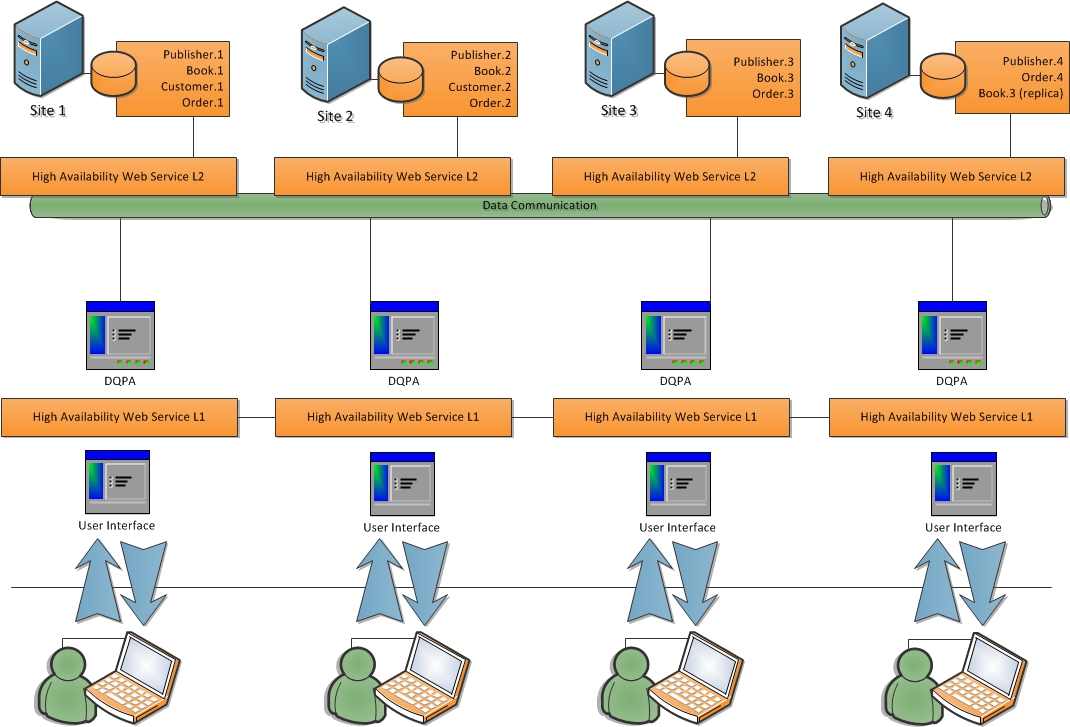


Figure 1 Distributed Database Query Engine Service (DDQES) Architecture Paradigm

Based on the above illustration, it can be further explained as the users who have submitted the SQL queries to Distributed Database Query Engine System (DDQES) own no direct knowledge regarding the backend operations and data communications; as well the physical storage location of the intended data. The users simply need to submit their query requests to the published web interface which is reachable through an URL. The 2-Level High Availability (HA) Web Service is responsible of further process and branching of proceeding operations which will be discussed in the coming section [Section: Web Services].

## Web User Interface

The designed web interface serves as the main communicating interface between Distributed Database Query Engine System (DDQES) and the user; as the users are expected to submit their query requests through the web interface while the actual processing of handling the requests will be branched out after receiving the user requests. In fact, every site may publish a Java Server Page (JSP) which hosted with a different URL address. In another way of saying, there are mainly four accessible webpages which can be used by the users for submitting their intended queries. Temporary failure of any site may not affect the operation of DDQES as a whole where the user can simply use another site for the queries.

Upon submitting the SQL queries to the web interface, the backend execution will begin and finally returned with a result which will be presented in a user-friendly way. To be more specific, once the users have submitted the SQL query to the web interface; an immediate check will be carried out to identify the on-line High Availability (HA) Web Service – Level 1 [Section: 2-Level High Availability (HA) Web Services] which is capable of handling the next step operation. Once identified, a further request will be submitted to the target High Availability (HA) Web Service – Level 1 which will further create an instance of Distributed Query Processing Application (DQPA) that will take care of the entire distributed query processing and finally return with a corresponding result set. In short, the hosting web user interface is merely an interface which allows the users to submit query request and view the result of response from DDQES. In the following part, the core backend processing elements are covered, which is the 2-Level High Availability (HA) Web Services.

## 2-Level High Availability (HA) Web Services

Narrow down to the details of High Availability (HA) Web Service, there are mainly two levels (HA Web Service - Level 1 and HA Web Service – Level 2) of web services to support the backend executions to respond to user requests. As mentioned in the previous section, the four Java Server Page (JSP) webpages which are hosted individually in the four sites; serves as an interface to accept user requests also will immediately identify the on-line hosting High Availability (HA) Web Service – Level 1 (L1) which will be in-charge of handling the entire execution to finally obtain an intact query result set; further return to be formatted and presented to the user through the web user interface.

Before proceeding to the operation flow behind, it is worth-nothing that each of the four sites in DDQES is hosting their own 2-Level High Availability (HA) Web Services. It will of course be the best case where the local 2-Level HA Web Service is on-line and once the users have submitted the queries; directly the local web service will be appointed to perform immediate execution. Anyhow, if the local 2-Level HA Web Service is offline for reasons, the entire designed architecture owns no issue to process the user queries too. Hence, it is able to promote a high-availability web service providing rates as the probability of having all four web services down is relatively low. In the next part, details of High Availability Web Service – Level 1 (HAWS-L1) and High Availability Web Service – Level 2 (HAWS-L2) are covered.

### High Availability Web Service (HAWS) – Level 1 (L1)

Once the web services are published, they are ready to receive any request. Under the first level which is HAWS-L1, an instance of Distributed Query Processing Application (DQPA) will be created which is responsible of further operation branching and integration to return back to caller with results upon finishing the execution. Once DQPA is created, user submitted SQL query will be decomposed into sub-queries and further optimized execution strategy plan will be created by considering the following:

* The current on-line HAWS-L2
* The size of query result set
* The time cost of query
* The communication cost of the query

In fact, the optimized execution strategy plan contains information of which site is the best site which will incur less communication cost throughout the data retrieval processes. A detailed content of optimized strategy plan can be viewed in Section: TBA. So, after the best and optimized execution plan is generated, the plan will be further submitted to High Availability Web Service – L2 (HAWS – L2) of the targeted best site to execute the identified data retrieval, communication and integration plan. In fact, the communication between HAWS-L1 and HAWS-L2 is through another web service request; where HAWS-L1 send a request to identified best site with the optimized execution plan as parameter and expecting a response from HAWS-L2 with result.

### High Availability Web Service (HAWS) – Level 2 (L2)

Usually, requests are submitted through HAWS-L1. Upon receiving request with parameters, HAWS-L2 will execute the data retrievals and data integration which involved union and join operations for merging the result sets. There are multiple operations exist in HAWS-L2 which can be utilized to fulfil the operation needs, mainly:

* The main handler which executes the receiving plan and generate respond
* Data retrieval which obtains data from intended site: Since each site contains their own 2-Level High Availability Web Service which also included HAWS-L2, submitting a web service request to the intended site can receive a respond with intended data. Although the concept is such, the actual supporting data retrieval operation happened in each site relies on the database system communication will be discussed in Section: Database System and Engines Communication.
* Consolidate and merging result set: Perform union or join based on the optimized execution plan.

A detailed content of the communicating request and respond data can be viewed in Section: TBA.

### High Availability Plan

Up to current point, it can be seen that the use of web service made the entire DDQES solution loose coupling and high cohesion which is relatively a good achievement; at the same times promoting high availability rates. Failure of any single web service at any level will not bring any visible defect to the user, provided certain intended data may not be retrievals. However, of course if all the hosting 2-Level High Availability Web Services are all crashing; there will be no solution yet to cater for this worst case.

## Database System and Engine

For times it is relatively hard to standardize all the distributed systems to be equipped the same database systems and/or engines for data storage and handling; the project is designed to handle the communication of multiple heterogeneous database systems, including PostgreSQL, MySQL and SQLite; where each database system was handled equally by each team member. It is believed that such implementation may increase the practicality of the entire project. The following is the details of installed database system according to sites:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Site ID** | 1 | 2 | 3 | 4 |
| **Installed Database System** | MySQL | SQLite | PostgreSQL | MySQL |

Table 1 Installed Database System Engine based on Site ID

## Global Table Structure

In details, each database system in each site is based on the following table structure as mentioned previously; including Publisher, Book, Customer and Orders table:

|  |
| --- |
| Table Structure |
| Publisher (pid integer PRIMARY KEY, pname char(100), pnation char(3))  Book (bid integer PRIMARY KEY, btitle char(100), bauthors char(200), bpid integer, bcopies integer)  Customer (cid integer PRIMARY KEY, cname char (25), crank integer)  Orders (ocid integer, obid integer, oquantity integer) |

Table 2 Global Table Structure of Publisher, Book, Customer and Orders

## Table Fragmentation

As can be noticed from the Distributed Database Query Engine Service (DDQES) architecture paradigm [Figure 1], the number of tables and ID are different from one site to another. It is in fact drawn based on the fragmentation rule which aim to optimize data communication cost and performance. The fragmentation details can be summarized as below, grouped by table name:

* Publisher

|  |  |  |
| --- | --- | --- |
| Fragmentation | Fragmentation Condition | Site |
| Publisher.1 | pid < 104000 AND pnation=’PRC’ | Database 1 [Site 1] |
| Publisher.2 | pid < 104000 AND pnation=’USA’ | Database 2 [Site 2] |
| Publisher.3 | pid >= 104000 AND pnation=’PRC’ | Database 3 [Site 3] |
| Publisher.4 | pid >= 104000 AND pnation=’USA’ | Database 4 [Site 4] |

Table 3 Fragmentation of Publisher Table

* Book

|  |  |  |
| --- | --- | --- |
| Fragmentation | Fragmentation Condition | Site |
| Book.1 | bid < 205000 | Database 1 [Site 1] |
| Book.2 | bid >= 205000 AND bid < 210000 | Database 2 [Site 2] |
| Book.3 | bid >= 210000 | Database 3, 4 (R) [Site 3, 4] |

Table 4 Fragmentation of Book Table

* Customer

|  |  |  |
| --- | --- | --- |
| Fragmentation | Fragmentation Condition | Site |
| Customer.1 | (cid, cname) | Database 1 [Site 1] |
| Customer.2 | (cid, crank) | Database 2 [Site 2] |

Table 5 Fragmentation of Customer Table

* Orders

|  |  |  |
| --- | --- | --- |
| Fragmentation | Fragmentation Condition | Site |
| Order.1 | ocid < 307000 and obid < 215000 | Database 1 [Site 1] |
| Order.2 | ocid < 307000 and obid >= 215000 | Database 2 [Site 2] |
| Order.3 | ocid >= 307000 and obid < 215000 | Database 3 [Site 3] |
| Order.4 | ocid >= 307000 and obid >= 215000 | Database 4 [Site 4] |

Table 6 Fragmentation of Orders Table

On top of the basic database configurations to support the implementation of Distributed Database Query Engine Service (DDQES), Hibernate Framework will be used which is an object relational mapping library for mapping an object-oriented domain model to a traditional relational database. In the following section, detailed discussion on the framework will be described.

# Database System and Engines Communications

The database system is composed of 2 different levels that are the local systems and the master system.

## Local Systems

The management of the local database systems is regulated by Hybernate. Hybernate is a Java library which provides an Object-Relational Mapping (ORM). This allows us to define the local tables with an XML file and class mapping. Then, during the execution, it can generate objects according to the XML description by using a DOM4J session. So that the sub-queries processed locally are handled with a generic code.

## Master System

[Check if this is still necessary]

Located at the master site, it correspond to the web-service that will be in charge to split a request, transmit the sub-requests to the local systems, make the unions, joins and semi-joins of the result-sets, and finally return the final result-set to the user.

## Query Processing

Here is an example step by step of the processing of a query:

1. User input its query by using the web interface or web-service.
2. The master site generates the query tree.
3. From the query tree, the master site creates 1 sub-query per site involved in the query.
4. Each sub-query is sent to the corresponding site.
5. Each local site processes its own sub-query by using the Hybernate framework.
6. Each local result-set is sent to the master site.
7. By referring to the query tree, the master site realizes unions, joins and semi-joins to get the final result-set.
8. The final result-set is sent to the user through the web-service.

## Communication

The communication between the different sites is done by sending packets over the network.

These packets contain the following information:

1. A fixed hexadecimal code to ensure that the packet is sent by the system.
2. A packet ID
3. A command ID to specify the type of message:
   1. QUERY (0 x 1)
   2. SUB-QUERY (0 x 2)
   3. ANSWER (0 x 3)
   4. RESULT (0 x 4)
   5. PING (0 x 5)
   6. PONG (0 x 6)
4. Query (XML formatted)
5. A timestamp for further optimization such as avoiding request collision.
6. A checksum to make sure the packet has not been corruption during the transfer.

## Master Site Strategy and Dictionary Storage

The master site is the site where most important processes such as the web service for interface, query distribution, and query optimization will take place. This master site will be the central “brain” for the system to operate and so this master site will be very important.

One of the main concerns is when the master site is not available because of several possibility reasons. Since it is very important then it is a must to elaborate some strategies to anticipate the possible conditions. One of the strategies is to actually make every site to have a potential to be master site. In another words, every site can be a master site even though at one time, there will be only one site which will act as the master site. With this strategy, when the current master site is not available, another site will be ready to replace as the master site so it will not disturb the service to the user. A mechanism similar to “ping” will be used to check and make sure that the current appointed master site is working.

The concept here is that it’s like a domain name and hosting service. We have a domain of www.abc.com for example which is an access point for the user to submit query. If the current master site is down (failed to ping it); then another site will take over to host the www.abc.com. This will be the main strategy which our team will implement. There is possibility when we actually working in building the system there will be several changes to this strategy.

Another important issue regarding the master site in the system is the dictionary. The dictionary can be considered as one form of the meta data for all the data which will be used in the system. This dictionary will be an irreplaceable component for the master site to do the processing especially for the query processing. There are many strategies regarding how to store the dictionary. In our implementation, we will store dictionary on all the sites. There are several considerations for implementing this strategy; one of the most important is that by storing it on all sites the dictionary will always be available to access as long as one of the sites is working.

## Summary

Various efforts were placed by the team members to finally come out with a detailed and effective design of system architecture for Distributed Database Query Engine System (DDQES), starting from the web user interface, including every detail of communication point between the application and different level of High Availability Web Service in performing targeted operations; until finally generate returning result set as well as present it with a user-friendly web user interface. The main highlight for such a design is due to its nature of providing high availability rates and user unawareness towards the backend architecture and intended data location. In the next section, detailed elaborations of project implementations are included.

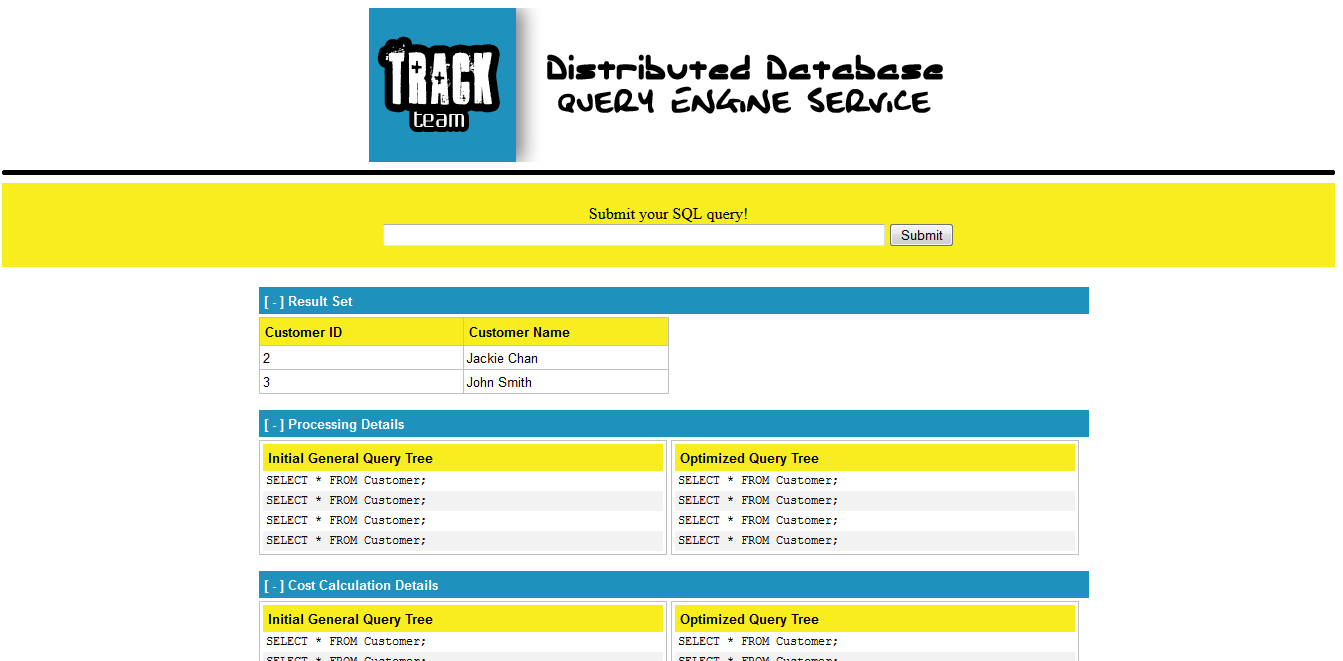
# Project Implementation

## Web User Interface

Web interface serves as the main communicating interface between Distributed Database Query Engine System (DDQES) and the users. Therefore, the main consideration here is its user-friendliness where the users should not be required to invest any learning effort to use it. Hence, learnability and simplicity are the main design objectives. By considering the user tasks while using the web interface, it can be listed as two simple tasks:

* Submit SQL queries
* View the result

To satisfy the above needs, the following interface is created by maintaining the user attention to the content while leaving sufficient white space to avoid the users from feeling overwhelmed towards the contents. The following is the print screen of the designed user interface to satisfy the mentioned tasks:



1

2

3

4

5

6

7

8

9

10

Figure 2 Web Interface of Distributed Database Query Engine Service (DDQES)

Based on the above figure, the functionalities are labeled with individual number and its corresponding explanation can found as follow:

1. Reload the current page
2. Enter SQL query, such as SELECT \* FROM customer
3. Submit SQL query by sending request to backend system for further executions
4. Error message. For the case where the user is entering an invalid SQL query, an error message will be shown up with red in font. An example of invalid SQL is “SELECT \* FROM people” as there is no such table named people in the database.
5. Clickable hide/show result set bar. By default, labeled element 6 will be shown. By clicking on this bar, the entire section of element 6 will be hidden. For the case where element 6 is already being hidden, another click will show the content again.
6. The result set of the entered SQL query
7. Clickable hide/show processing details bar. By default, labeled element 8 will be shown. By clicking on this bar, the entire section of element 8 will be hidden. For the case where element 8 is already being hidden, another click will show the content.
8. Processing details which having the default initial general query tree on the left while the optimized query tree on the right.
9. Clickable hide/show cost calculation details bar. By default, labeled element 10 will be shown. By clicking on this bar, the entire section of element 10 will be hidden. For the case where element 10 is already being hidden, another click will show the content again.
10. Cost calculation details which having the communication cost of the default initial general query tree on the left while having the communication cost of the default optimized query tree on the right.

As the entire web interface may need to be accessible by the users online, Java Server Page (JSP) is created as it can be hosted online and can be accessible through a corresponding URL. To style the entire web interface, a Cascading Style Sheet (CSS) is written while a JavaScript is linked within the webpage to support to show and hide functionalities.

Due to its nature as a JSP, application codes can be written to support any necessary feature. Therefore, it is holding two functions behind its default functionality of displaying the web interface:

* Validate user submitting SQL query: A validation check happened once the user submitted the request by clicking on “Submit” button. The check is taken place by pre-running it with the database to check its error. If there is an error, it will be shown in element 4 as shown in Figure 2 with red font. For the case where there is no error for the submitted SQL query, it will proceed with execution.
* Stand as an entry point for further result execution: At this step, it simply means the submitted SQL query is valid. Then, check on the on-line web service is carried out which will determine the later optimized strategy plan generation. And, all the backend execution begins from this point.

## 2 –Level High Availability Web Service

According to the identified architecture design, there are two levels of High Availability Web Services; each handling different tasks namely generation of optimized query tree and strategy while the other execute the generated plan to gather, merge and return with result set. These two web services are described using Web Service Description Language (WSDL) which including the description on available functions and data communication as request parameter(s) and respond.

## High Availability Web Service – Level 1 (HAWS-L1)

### Data Decomposition

Distributed Database Query Engine System (DDQES) is within a distributed environment and therefore query may need to be further decomposed into sub-queries as data is not being stored in only one site. Query decomposition allows the query to be reconstructed after identifying the best strategy plan to be executed; further submitting to corresponding sites to retrieve intended data which will be discussed in Section: High Availability Web Service – Level 2 (HAWS-L2). However, it is normal that user submitting SQL queries may involve multiple relational tables. For instance, “SELECT Customer.cname, Orders.oquantity FROM Customer, Orders WHERE Customer.cid = Orders.ocid” is involving the data retrievals from two tables, which are Customer and Orders. In addition, another concern is regarding the flexibility of SQL syntax. For example, “SELECT cname, oquantity FROM Customer, Orders WHERE Customer.cid = Orders.ocid”, “SELECT Customer.cname, Orders.oquantity FROM Customer, Orders WHERE Customer.cid = Orders.ocid” and “SELECT cname, oquantity FROM Orders INNER JOIN Customer ON Orders.ocid = Customer.cid” are all syntactically correct as well as carrying the same meaning. Since it is rather challenging to implement the feature to cover all the SQL possibilities, the scope is limited to support the syntax without JOIN keywords as follow:

SELECT cname, oquantity FROM Customer, Orders WHERE Customer.cid = Orders.ocid

SELECT Customer.cname, Orders.oquantity FROM Customer, Orders WHERE Customer.cid = Orders.ocid

SELECT cname, oquantity FROM Orders INNER JOIN Customer ON Orders.ocid = Customer.cid

So, the data decomposition feature is coded in such a way that it will be able to identify each selected column and predicates belonging table. For example, if “SELECT cname, oquantity FROM customer, orders WHERE customer.cid = orders.ocid” is entered, the following is the decomposed results:

|  |
| --- |
| **Table**  Table: orders, Alias: orders  Table: customer, Alias: customer  **Select statement**  Table: orders, Column: orders.oquantity, Alias: oquantity  Table: customer, Column: customer.cname, Alias: cname  **Where clause**  **Join condition**  Join tables: customer,orders, customer.cid = orders.ocid |
| SELECT orders.obid, orders.oquantity, orders.ocid FROM orders;  SELECT customer.cname, customer.cid FROM customer; |

Referring to the above table, the first section is referring to the decomposition of the entire SQL query to its finest granularity while the second section is the reconstruction of the decomposed elements to be grouped by relation table which can be used to support the further decomposition of optimized query tree.

### Query Tree Optimization and Optimized Strategy Plan Generation

Discuss on how the optimized query tree is created and optimized strategy plan is generated, what are the considerations? What is the content of optimized strategy plan and in which format? What is the next step? How the information is communicated?

## High Availability Web Service – Level 2 (HAWS-L2)

Upon receiving the optimized strategy plan, how the execution is carried out, through web service? Did you check the site before retrieving? How do you handle if it is offline? What is the communication format, such as submitting parameter to web service, or? [As detail as possible]

### Hibernate Data Retrievals and Communication

What is the procedure and logic of retrieving the data? What is the format?

## Packages and Classes

[We will write this if we have enough time]

# Project Timelines

Since the project was initiated, the team was having a weekly meeting to plan and follow up the project progression [Appendix I - VIII]. Good project management starts with proper scheduling of a project which may increase the project success rate. As indicated the deadline submission of the project is mid of June, below is the created milestones for the team proper progression and monitoring after the last group discussion. It is glad that these milestones were closely followed throughout the entire project progression. While the person responsible simply means the person who owns a higher knowledge towards one particular subject, the entire team will be paying full attention and coordination to contribute to the project success:

|  |  |  |
| --- | --- | --- |
| Milestone Title | Date | Person Responsible |
| Mid-Point Project Review  Finalize Launcher and Hibernate Connectivity | 29th April, 2013 | Ying  Fred |
| Perform Query and Data Communication Cost  Finalize Optimization Strategy Plan | 6th May, 2013 | Fred, Henry, Ying  Ying |
| Database Building and Hibernate Linkage  Final Documentation Template Creation | 13th May, 2013 | Fred  Ying |
| Individual Application Development (Java) | 20th May, 2013 | Henry |
| Application and Connection Integration | 27th May, 2013 | Fred, Henry, Ying |
| Application Wrap and Documentation | 3rd June, 2013 | Fred, Henry, Ying |
| Final Review and Presentation Wrap Up | 10th June, 2013 | Henry |

Table 7 Table of Project Timelines

## Workload Matrix

Distributed Database Query Engine System (DDQES) is a successful project due to a close cooperation among team members and also a fair distribution of tasks. In actual speaking, the entire project was analyzed, designed, developed and tested together. To be more précised into every individual task, it can be summarized as the following table:

|  |  |  |
| --- | --- | --- |
| Task Name |  | Person Responsible |
| Distributed Database Query Engine System Development   * Web User Interface * 2-Level High Availability Web Service   + High Availability Web Service – Level 1 (HAWS-L1)     - Query Decomposition     - Optimized Strategy Plan     - HAWS-L1 Web Service Conversion   + High Availability Web Service – Level 2 (HAWS-L2)     - Optimized Strategy Plan Execution     - Hibernate Database Communication     - HAWS-L2 Web Service Conversion * Integration * Database System   + Hibernate Database System Connection Handler     - MySQL     - SQLite     - PostgreSQL   + Database Structure Creation   + Database Test Data Creation | | Ying  Ying  Fred, Henry  Henry  Ying  Henry  Fred  Fred  Fred  Fred, Henry  Fred  Fred  Fred  Ying  Henry  Ying |
| Final Documentation   * Introduction * Project Scope * System Environment * 2-Level High Availability (HA) Web Services   + High Availability Web Service – Level 1 (HAWS-L1)   + High Availability Web Service – Level 2 (HAWS-L2) * Database System and Engine * Global Table Structure * Table Fragmentation * Database System and Engine Communications * Local Systems * Master System * Query Processing * Communication * Master Site Strategy and Dictionary Storage * Summary * Project Implementation * Web User Interface * 2 –Level High Availability Web Service   + High Availability Web Service – Level 1 (HAWS-L1) * Data Decomposition * Query Tree Optimization and Optimized Strategy Plan Generation   + High Availability Web Service – Level 2 (HAWS-L2) * Hibernate Data Retrievals and Communication * Packages and Classes * Project Timelines * Workload Matrix * Project Challenges * Future Enhancement * Conclusion * Meeting Minutes * Final Integration | | Ying  Ying  Ying, Fred, Henry  Ying  Ying  Ying  Ying  Ying  Ying  Ying  Ying  Fred  Fred  Fred  Fred  Henry  Ying  Ying  Ying  Ying  Ying  Ying  Henry  Fred  Fred  Ying  Ying  Ying, Fred, Henry  Ying  Ying  Ying  Ying |
| Final Presentation Slide  Presentation Content  Presentation Template | | Ying, Fred, Henry  Ying, Fred, Henry  Ying |
| System Operation Specification   * Introduction * Project Development Environment * Installation and Configuration   + Required Hardware   + Required Software   + Launching Application   + Help   + Support Contacts | |  |
| Mid-Term Presentation Slide  Presentation Content  Presentation Template | | Ying, Fred, Henry  Ying, Fred, Henry  Ying |

Table 8 Workload Matrix

# Project Challenges

As the entire Distributed Database Query Engine System (DDQES) is completed and finally ready for submission, there are quite a lot of efforts being placed behind to make the entire project a success one. Hence, it worth highlighting the challenging parts of the project:

* 2-Level High Availability Web Service

Before DDQES project, there is none of the team members own any knowledge regarding the implementation of web service and Web Service Description Language (WSDL). Therefore, converting from the typical object oriented programming model to web service which support data communication by merely sending request and respond is a new experience and yet a very challenging point. There was a steep learning curve implementing it.

* Query Decomposition

As there is no open source SQL parser to be used for query decomposition, the entire query decomposition algorithm is written by the team with a huge variety of considerations regarding the possible user input. Continuously testing and experiments were carried out before it was finalize with the current result.

* Optimized Query Tree Generation

Henry’s

* Heterogeneous Database System Communication

Even though Hibernate framework is being used to reduce the difficulties in handling heterogeneous database environment, it is still challenging to perform the configuration and data retrieval processes due to the difference in nature of different database system.

## Future Enhancement

Though the Distributed Database Query Engine System (DDQES) project is rather complete for its submission version, there are still several improvements and enhancements can be done to make it even better. In fact, it can be improved in various different aspects, including the server configurations and SQL decomposition.

* Server Configuration for URL Redirect

For current moment, if one particular hosting web interface is down; the user may need to manually enter another site URL to submit the intended SQL queries. However, if proper configuration is done beforehand; if the visiting URL is down; the server should be automatically redirecting the users to another site seamlessly.

* SQL Syntax Decomposition

The current version of DDQES is capable of handling SELECT statements with simple predicates conditions that are connected by logical AND, NOT and OR. Even though the decomposition can handle Having, Group By and Order By clauses, there are still no implementation in place to consider these situation while generating the optimized strategy plan. Therefore, Having, Group By and Order By are being included for optimization; the result will be even better.

# Conclusion

Though TRACK team had decided to adopt the default scenario for implementing Distributed Database System Query Engine Service (DDQES), the following are the strengths and/or challenging points which worth a note:

* Four physical computers: The team is increasing the project difficulty level by pushing the DDQES environment to be run from four different physical computers rather than running on a single computer with four virtual sites.
* Heterogeneous database engines: Instead of putting all four databases into a single computer which most likely to be the same database system, the team is challenging the scope by using heterogeneous database system; including MySQL, PostgreSQL and SQLite by using Hibernate framework library.
* High Availability (HA) Web Service: To reduce the limitation of the nature of distributed database system, the publishing web service is aimed to provide a high availability rates which is capable of service recovery to reduce the possibility of having an offline service.
* Loose coupling and high cohesion web services: By publishing all the web services it makes the data communication and process executions to be more flexible as well as having a higher availability rates as a whole. Therefore, it is considered as a breakthrough and valuable point of the project.

It is considered as a valuable experience to understand the overall distributed database environment through this entire project. The well-integrated ad highly cooperative TRACK team members had made the project a successful; not to overlook the knowledge sharing sessions among the team members. Lastly, it is reasonable to conclude the entire DDQES project a successful product with full considerations from various aspects, including initiation, analysis, design and development.

# Appendices

## Appendix I – Meeting Minutes 1

Date : 11th March, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Identify team leader
* Identify as to create new scenario or use the default one
* Identify development environment
  + Number of sites
    - Using same application or different sites uses different application environment, such as site A for finance and site B for warehousing
  + Programming language
  + Data communication and transmission medium
  + Database management system
* Gather all team members’ student ID
* Desirable project task distributions and responsibilities
* Identify schedule for proper progression tracking of the project
* Next step to proceed after discussion

**Meeting Minutes**:

* The student ID of the team members were noted to be filled up to the discussion board in the website and Yi Ying was selected as the team leader.
* The team decided to use back the default scenario and add features to the existing one after the basic requirements are met.
* The following are the discussion results regarding the development environment
  + Number of sites: The team decided to stick back to the default 4 sites and expand it if there’s a time.
  + Programming language: The team decided to start with the same programming language first and JAVA is preferred.
  + Data communication and transmission medium: It is preferable to use socket and TCP/IP or UDP which able to expose the data through web services.
  + Database Management System: The team had shortlisted Hybernate and MySQL.
* The team decided to work together for all tasks and will distribute other unknown tasks in future for better project management.
* Every Monday will be the weekly meeting for the team and during the meeting, the team members will review the progression since last week, discuss on the tasks for this week and how additional working hours are required for this week.
* Fred suggested storing the scenario condition into an XML, further creating an application to generate and launch the sites by referencing the XML “dictionary”.

**To-do List**:

* Submit group to discussion board in <http://learn.tsinghua.edu.cn>
* The student ID for team members are as follow:
  + Frederic Colin – 2012400565
  + Henry Loharja - 2012280102
  + Ng Yi Ying – 2012280098 [Team Leader]
* Register for GitHub and download and install Git
* Create a scenario for the assignment by specifying the limitation and general picture of the entire project
* Create a prototype launcher based on the XML
* (Optional) Draw the relationship among sites and possible working/communicating logic

## Appendix II – Meeting Minutes 2

Date : 18th March, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Brief discussion on last week lecturing materials
* Review on the draft of the created first scenario
  + Architecture and drew diagram
  + XML for environment variables
  + Scalability
  + Potential modifications
* Review on the created launcher prototype
* Shortlisting database systems to further prototyping
* Distribution of tasks
* Scheduling and milestones for better project monitoring
* Next step to proceed after discussion

**Meeting Minutes**:

* A brief review of last week lecturing was carried out by the team.
* The initial draft of the case scenario was finalized to be just fine for the starting and will definitely enhance the model when the project goes on. The potential solution is to make each and every site a master site of its own so that when one of the sites was down, the other may still progress while the other will perform the recovery.
* Fred and Henry were agreed to check out on Hybernate and its connectivity to first launcher prototyping while Yi Ying will be working on MySQL to further explore the database connectivity.
* After checking the schedule for mid-term presentation, it was agreed by the team that there will be sufficient time to prepare and work on this project to achieve desire goals. For now, weekly meeting and report are agreed to be kept going as more prototyping can increase the completeness of the project.

**To-do List**:

* Fred and Henry: Try out launcher prototype and test out the connectivity between the application and Hybernate.
* Yi Ying: Try out the connectivity between the application and MySQL and try to expand the launcher prototyping.

## Appendix III – Meeting Minutes 3

Date : 25th March, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Brief discussion on last week lecturing materials
* Review on the draft and progression of the created second scenario
  + Connection from Launcher to MySQL
  + Connection from Launcher to Hibernate
* Discussion on the concept of web services (Since one of the requirements expects the distributed database query engine is capsuled into a web service) and WSDL
* Review on the required content of mid-term design report and presentation
  + Design of the distributed database query engine
  + Timeline for project work plan
* Next step to proceed after discussion

**Special Notes:**

* Mid-term presentation will be on 26 April, 2013

**Meeting Minutes**:

* A short discussion on the homework assignment was carried by linking to the project needs in the future
* The use and publish of service to the web may need further verification with the teaching assistant before proceeding
* It is agreed to ready the launcher before next coming meeting for evaluation and preparation for later mid-point presentation

**To-do List**:

* Meet up the teaching assistant for concept verification
* Build up the launcher connectivity to MySQL, Hibernate and Postgre

## Appendix IV – Meeting Minutes 4

Date : 19th April, 2013

Time : 15:00 pm – 18:00 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Review on the draft and progression of the created second scenario
  + Connection from Launcher to MySQL
  + Connection from Launcher to Hibernate
* Review on the required content of mid-term design report and presentation
  + Design of the distributed database query engine
  + Timeline for project work plan
* Preparation of midterm presentation and report writing section distribution
* Next step to proceed after discussion

**Meeting Minutes**:

* A short discussion on last class of distributed database system.
* The current progression of the Launcher was reviewed by specifying existing limitation of not yet including Hibernate as part of the prototype.
* However, the connection from Launcher to database system is relatively stable for now and should plot over to include Hibernate right after midterm presentation.
* The discussion of the entire scenario was carried out to double confirm the discussed details in the past month.
* The content of the midterm report was discussed and section writing was distributed and agreed to complete before Thursday for integration.
* The demonstration of Launcher was agreed not to push up to the midterm presentation since it is not quite complete yet.
* Presentation slide will be prepared by Yi Ying and content will be added by Fred and Henry.
* Another meeting was expected to be carried out before the midterm presentation for rehearsal of having better presentation performance.

**To-do List**:

* Write on midterm report
* Create midterm presentation slides and content

## Appendix V – Meeting Minutes 5

Date : 29th April, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Brief discussion on last week presentation lesson and feedbacks
* Review on the draft and progression of the created second scenario, finalizing:
  + Connection from Launcher to MySQL, MySQLite and PostgreSQL through Hibernate
* Discussion on the immediate step in implementing next steps as of prioritizing which tasks first:
  + Stable programming solution
  + Cost estimation calculation
* Task distribution for the following tasks
* Next step to proceed after discussion

**Meeting Minutes**:

* The team agreed to finalize and come out with a working solution in next week meeting for all the mentioned three database connections:
  + MySQL
  + SQLite
  + PostgreSQL
* It was agreed to work on the programming part first as the stability of a working solution should be in place and optimization plan can come in later throughout the process
* Fred was assigned to handle connectivity to MySQL, Henry assigned to PostgreSQL connectivity while Yi Ying was assigned to handle connection to SQLite.
* To test out the latest Hibernate Framework, version 4.1.12 was suggested and accepted to be used for the building.

**To-do List**:

* Finalize and develop final version of Launcher to Hibernate and to corresponding database system

## Appendix VI – Meeting Minutes 6

Date : 8th May, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Brief discussion on last week assigned work progression
* Review on the progression of the created second scenario, finalizing:
  + Connection from Launcher to MySQL, MySQLite and PostgreSQL through Hibernate
* Discussion on the next application development plan and achievements
* Task distribution for the following tasks
* Next step to proceed after discussion

**Meeting Minutes**:

* There were issues encountered while finalizing the Hibernate connection from the Launcher application upon adopting the latest Hibernate library files:
  + SQLite connection was not well-established from Hibernate
    - Solved by moving the configuration file to source folder
  + PostgreSQL connection was not well-established yet from Hibernate due to lack of tutorial resources
  + Issues of upgrading as the preferred feature in version 3+ Hibernate Framework was wiped off from the latest Hibernate Framework
* It was agreed to work to downgrade the Hibernate Framework down back to the latest version of 3+ due to the shortcoming of latest Hibernate Framework. The downgrading version is 3.6.10 which can be downloaded from

<http://sourceforge.net/projects/hibernate/files/hibernate3/3.6.10.Final/>

* To quickly pull the project on-track, Fred was assigned to work on exploration of the intended features which will add value to the entire application project, tested and conduct knowledge transfer.
* Henry was assigned to handle the connection between Hibernate and PostgreSQL with Hibernate Framework 3.6.10.
* Yi Ying was assigned to handle the structure of proper programming file management format and the online socket connection between computers.

**To-do List**:

* Downgrade Hibernate Framework back to 3.6.10 which supporting XML mapping and format identification.
* Connection between Hibernate and PostgrSQL
* File management structure and programming naming convention
* Socket connection between two computers

## Appendix VII – Meeting Minutes 7

Date : 20th May, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Review on the progression of the created second scenario, finalizing:
  + Connection from Launcher to MySQL, MySQLite and PostgreSQL through Hibernate
* Discussion on the current progression and achievement since last meeting
* Finalizations on the modification of version of Hibernate
* Task distribution for the tasks in development
* Next step to proceed after discussion

**Meeting Minutes**:

* The demo of the socket network communication was brought up for team sharing knowledge; the use of networking class was presented by Yi Ying
* It was finalized by the group where it’s no longer necessary to downgrade the version of Hibernate as it is already possible to achieve the intended functionality.
* Connection to Hibernate and PostgreSQL was successful; same goes to MySQL and SQLite.
* Discussion on the remaining task and schedule was carried out.
* The tasks were split among team members and each of the member will be handling each part as follow:
  + Yi Ying: Interface, web services and query decomposition
  + Henry: Based on the decomposed query, generate optimized query tree
  + Fred: Capture optimized query from Henry’s part and perform data retrievals within site and send it back to calling site
* Discussion in term of overall working plan of the entire application, query decomposition, calculating and identifying optimized query were carried out.
* It was agreed by the team members as all the works will be at least drafted with confidence before coming next meeting.

**To-do List**:

* Core application development
  + Web user interface, web services, query decomposition, query tree optimization, data communication and retrievals
* Final report structure identification
* Final presentation slide identification

## Appendix VIII – Meeting Minutes 8

Date : 20th May, 2013

Time : 10:00 am – 12:30 pm

Venue : 8-304

Attendees : Frederic Colin, Henry Loharja, Ng Yi Ying

**Intended Discussions**:

* Review on the progression of the created second scenario, finalizing:
  + Connection from Launcher to MySQL, MySQLite and PostgreSQL through Hibernate
* Discussion on the current progression and achievement since last meeting
* Finalizations on the modification of version of Hibernate
* Task distribution for the tasks in development
* Next step to proceed after discussion

**Meeting Minutes**:

* The demo of the socket network communication was brought up for team sharing knowledge; the use of networking class was presented by Yi Ying
* It was finalized by the group where it’s no longer necessary to downgrade the version of Hibernate as it is already possible to achieve the intended functionality.
* Connection to Hibernate and PostgreSQL was successful; same goes to MySQL and SQLite.
* Discussion on the remaining task and schedule was carried out.
* The tasks were split among team members and each of the member will be handling each part as follow:
  + Yi Ying: Interface, web services and query decomposition
  + Henry: Based on the decomposed query, generate optimized query tree
  + Fred: Capture optimized query from Henry’s part and perform data retrievals within site and send it back to calling site
* Discussion in term of overall working plan of the entire application, query decomposition, calculating and identifying optimized query were carried out.
* It was agreed by the team members as all the works will be at least drafted with confidence before coming next meeting.

**To-do List**:

* Core application development
  + Web user interface, web services, query decomposition, query tree optimization, data communication and retrievals
* Final report structure identification
* Final presentation slide identification