# Using Golang for implementation of a concurrent and distributed realtime processing system.

CHAI YING HUA

SESSION 2017/2018

FACULTY OF COMPUTING AND INFORMATICS
MULTIMEDIA UNIVERSITY
FEBRUARY 2018

# Using Golang for implementation of a concurrent and distributed realtime processing system.

BY

CHAI YING HUA

SESSION 2017/2018

THIS PROJECT REPORT IS PREPARED FOR

FACULTY OF COMPUTING AND INFORMATICS
MULTIMEDIA UNIVERSITY
IN PARTIAL FULFILLMENT
FOR
BACHELOR OF COMPUTER SCIENCE (HONS)
WITH SPECIALIZATION IN
SOFTWARE ENGINEERING
FACULTY OF COMPUTING AND INFORMATICS

MULTIMEDIA UNIVERSITY

FEBRUARY 2018

# Declaration

I hereby declare that the work in this thesis have been done by myself and no portion of the work contained in this thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Name: *Chai Ying Hua*
Student ID: 1141328508
Faculty of Computing & Informatics
Multimedia University
Date: $7^{th}$ February 2018

# Acknowledgements

The success and outcome of this project require tons of guidance and assistance from many people, and I am blessed and appreciate to have got this all along the completion of my project. My project would not be complete smoothly without their helping hands.

First and foremost, I would like to express sincere gratitude to my project supervisor, Mr Wan Ruslan Yusoff of Faculty of Computing Informatics at Multimedia University Cyberjaya for your unfailing support and assistance on my project. The door to Ruslan office and his mailbox was always open whenever I faced any impediment and trouble about my project or writing. He consistently and patiently steered me in the right direction whenever he thought I needed it. Also, he is eager to share his expertise and industrial experience in computing fields and provide encouragement and motivation on my project.

I owe gratitude to my parents for providing chances and opportunity for me to study in Multimedia University. They are caring, and concern about my academic and regularly provide support and attention on cultivating me to get myself prepare for an upcoming challenge.

Last but not least, I place on record, my sense of gratitude to my friend and classmate, who directly and indirectly unceasing encouragement and provide guidance till the completion of my project.

| | |
|---|---|
| **IBM** | International Business Machines |
| **GCP** | Google Cloud Platform |
| **AWS** | Amazon Web Services |
| **ICT** | Information and Communication Technology |
| **AMD** | Advanced Micro Devices |
| **GCC** | GNU Compiler Collection |
| **GCCGO** | Golang GNU Compiler Collection |
| **LEO** | Longitudinal Education Outcomes |
| **NSPL** | National Statistic Postcode Lookup |
| **OORDBMS** | Object-Oriented Relational Database Management System |
| **MMU** | Multimedia University |
| **FYP** | Final Year Project |
| **IDE** | Integrated Development Environment |
| **UK** | United Kingdom |
| **CTF** | Capture The Flag |
| **MVCC** | Multi-Version Concurrency Control |
| **TCP** | Transmission Control Protocol |
| **HTTP** | Hypertext Transfer Protocol |
| **OO** | Object-Oriented |
| **POC** | Proof Of Concept |
| **OS** | Operating System |
| **CSV** | Comma Separate Values |
| **GDB** | GNU Project Debugger |
| **GNU** | GNU's Not Unix! |
| **UNIX** | Uniplexed Information and Computing Services |
| **SQL** | Structured Query Language |
| **WIP** | Work In Progress |
| **DDL** | Data Definition Language |
| **DML** | Data Manipulation Language |
| **GUI** | Graphic User Interface |
| **LTS** | Long Term Support |
| **UK** | United Kingdom |
| **ORM** | Object Relational Mapping |
| **PL/pgSQL** | Procedural Language/ PostgreSQL Structure Query Language |

# Contents

# List of Tables

# List of Figures

# Listing

# Management Summary

The project focuses on a utilized concurrent programming language concepts and their expressive power on data processing with concurrent computing.

The research draws attention on implementation and utilization of Go and Rust programming language on data processing cycle with PostgreSQL database as data storage. These languages' paradigm, characteristic and focus are used in data preparation, data processing and data storage.

Big datasets are obtained from secondary sources with data collection and verify with data validation to inspect the quality and logical weakness in data contents. The raw datasets in CSV format will be backup and import into PostgreSQL database with data transformation. The defects discovered such as inconsistency, incorrect and duplication in large datasets are eliminated with data encoding and data cleaning. Ultimately, the unnormalized and unorganized data will be migrated into normalized table in new storage to establish excellent relational database management system freed from anomalies.

Several concurrent programming language based programs are developed to support data processing activities such as data transformation, data cleaning and data migration. The processing execution's performance of program developed from different concurrent programming language and programming style will be compared and discussed in detail.

PL/pgSQL scripts will be developed to create database entity's data structure, objects, schemas and perform data migration within PostgreSQL database. The lightweight scripts will execute multiple written query simultaneously to perform database creation, manipulation and control efficiently.

The project successfully prove concurrent programming has better performance and throughput on data processing compare to sequential programming. Data duplication, data inconsistencies and data incompleteness had successfully eliminated to establish high data quality. The capabilities and limitation of concurrent programming features on data processing are demonstrated and further discussed.

# Chapter 1

# Introduction

## 1.1 Introduction

In a globalization and modernization era, the volume and variety of big data continue to increase at an exponential rate. Cloud computing environment such as IBM, Microsoft Azure, GCP and Amazon AWS possess great shifts in modern ICT and robust architecture to perform large-scale and complex computing service for enterprise applications.[1] Chip makers AMD, IBM, Intel, and Sun rapidly building chips with energy-efficient multiple processing cores that improve overall performance by handling more work in parallel for server, desktops and laptops. [2] The performance and availability of system required to increase dramatically with the inclusion of multi-threading and multi-processing.

Software development activities are consistently working on improving efforts in development and deployment activities by solving issues, challenges and problem regarding concurrent and distributed computing. With the advent of client/server focus; massive cluster and networking technologies, the advancement of technology reveal problem and constraints on linguistic issues to the developer.[3] Availability of inexpensive hardware allow developer to exploit various possibilities in the construction of distributed system and multi-processors that were previously economically infeasible. [4]

Software application today is inherently and expanded into concurrent and distributed computing with real-time applications. [5]. However, majority of systems language not designed with concurrent and parallelization in mind and software users and load of request gradually increase.

Google created a new concurrent programming language, known as Go to rewrite their large production system to solve compile time and string processing by inventing a language that design for efficiency, simplicity and quick compilation without dependency checking. [6] At the same time, Mozilla Research invents a system concurrent programming language, known as Rust that emphasize security, safety and control with performance.

Go is free and open source programming language created by Google at 2007 and announce on 2009 [7] with two compiler implementation, GC and GCCGO. [8] The language were designed for high-speed compilation, support for concurrency and communication, and efficient or latency-free garbage collection. It is C-like and statically typed language that compiles into single binary with go compiler to reduce compile time. Go allow developer to model problems with a random order of events, optimize data operations, and utilize parallel processing of machines and network with concurrency programming. [9]

In this paper, we are going to focus on utilizing concurrent programming concepts of RUST and Go language in data processing activities. We will develop programs with Go and Rust to carry out retrieving, transform, cleaning, parsing and migration operations in data processing cycle. This paper attempts to expose important concepts of these languages and conduct a comparison for the use of self-study material and propose an evaluation scheme.

### 1.1.1 Project Brief Description

We will use the Go and Rust programming language to process a combination of static data to represents a real time, concurrent processing system. For this project, we will covering the utilization of concurrent languages' elements and key concepts in entire data processing cycle. The cycle consists of collection of data sources (inputs), implementation of data processors (program filters/codes), and manipulation of data storage.

The Go and Rust programming language based application process mash-up of three unambiguous, free and informed consent dataset in stream. These application are developed to demonstrate the capabilities of concurrent features on data processing activities. These program attempts to transform specific structure of data into required format of data storage. In addition, the application capable to detect and correct inaccurate records found in datasets and import into PostgreSQL database, an object-oriented relational database management system (OORDBMS).

PL/pgSQL, a procedural language supported by PostgreSQL OORDMS is used to write data definition language (DDL) and data manipulation language (DML) to create objects, schemas and structure of database. The query are written into files and composed as scripts to be executed automatically to perform database creation, manipulation and control efficiently.

The performance execution of program developed from different programming language and programming style will be recorded, compared and discussed in detail. Further conclusions and inference can be drawn from execution results to identify the expressive power and concepts of these language.

### 1.1.2 Project Objectives

The objectives of this project are:

1. To learn and understand about Go and RUST programming language concepts and their concurrent processing features.
2. To explore different techniques on data processing, concurrent and distributed programming for big data.
3. To conduct performance comparison between Go and Rust language implementation in data processing with concurrent programming.
4. To conduct a comparison on Go and Rust concurrent programming language concepts in retrieving big data with different techniques.
5. To implement the handling of big data with PostgreSQL, an object-oriented relational database management system (OORDBMS).

### 1.1.3 Project Motivations

During my involvement and participation of industrial training in JobStreet.com (A SEEK ASIA Company), my colleague often discuss about Golang implementation in worker thread with session on server side scripting to handle concurrent request and reduce web server loads. In Tech Talk Thursday with Grab Singapore organised in MMU Cyberjaya in January, the speaker mentioned the companies use Go language as tool to build their backend on handling request. Indirectly, the discussion and seminar by technical professionals stimulate my curiosity on capabilities and usage of golang.

In my process of exploration, I had attended several Golang meetups and learning sections in Kuala Lumpur. I am impressed the new language helps company saving cost on building servers and running well in small hardware specs. Other than that, I had discovered various notable company and sites start migrated their essential services and critical component from other languages to Go. Within several years, Google's Go language has gone from being an unfamiliar language to well-known promising tools or significant source for a big technology company to develop fast-moving new projects.

As Go soared to a new height in Tiobe programming language popularity, it has inspired me to gather more information and knowledge regarding the capabilities of the language. After viewing online articles and journals, I had discover this concurrency-friendly programming language may be the future of development, and it stimulates my passion and excitement for learning the language.

Simultaneously, I notice this project was published as FYP title in this semester. Without any hesitation, I am exhilarated to pursuit and register this project in my final academic year in order unveil the capabilities of golang. It will be enjoyable and great to learn this language throughout the project.

## 1.2 Project Scope

### 1.2.1 Phase 1 Scope of Work

1. Research project interest and raise question in different categories of data repositories.
2. Setup boot partition for Ubuntu 16.04 LTS operating system with Window 10.
3. Install Go language compiler and RUST language compiler on PC.
4. Install Eclipse for Parallel Application IDE.
5. Install Goclipse and RUST GUI into Eclipse IDE.
6. Install Terminator application into Ubuntu; it is an application that produces multiple terminals in a single window so that developer can perform various task in a single environment.
7. Install Synaptic Package Manager that enable upgrade and remove software package a user-friendly way without dealing with dependencies issues.
8. Set up PostgreSQL into PC for big data handling.

### 1.2.2 Project Deliverables for Phase 1

1. Acquire free, consent and big UK's basic company data published by Companies House in data.gov.uk that containing basic company data of live companies on the register for data processing.
2. Acquire institution subject data published by UK Higher Education site and create a mashup in a project which works with two sets of data and process them to provide output.
3. Acquire postcode data for UK location as the linker of basic company data with institution subject data.
4. Develop a proof of concepts and understanding on concurrent and program with Go language.
5. Write Go code for sequential and concurrent programs which able to process raw CSV data and PostgreSQL database.
6. Conduct comparison on sequential and concurrent programming with Go programming language on retrieving 300 rows of data.

### 1.2.3 Phase 2 Scope of Work

1. Perform data encoding to convert dirty data into consistent and valid format.
2. Perform database normalization to eliminate data redundancy and improve data integrity.
3. Write a Go programming language based sequential and concurrent program as ORM tool to export data from raw CSV file and PostgreSQL database convert into object model.

4. Write a Rust programming language based sequential and concurrent program as ORM tool to export data from raw CSV file and PostgreSQL database convert into object model.
5. Perform data cleaning to eliminate missing data and standardize the fields in consistent format.
6. Develop Go programming language based data cleaning parser to clean company raw datasets and import into PostgreSQL database.
7. Perform database tuning to optimize database's performance on handling extra workloads and increase client's connection limit.
8. Perform query tuning to increase query execution performance on data processing.
9. Develop several Go programming language based concurrent program as data migration tool to transfer data from legacy storage into new storage within PostgreSQL database.
10. Write several PL/pgSQL scripts to import raw data from legacy storage into normalized table within PostgreSQL database.
11. Perform data verification to verify the consistency and accuracy of database records after the data migration is complete.
12. Perform distributed programming to process data through multiple nodes.

### 1.2.4 Project Deliverables for Phase 2

1. A dirty raw CSV datasets shall be encoded and consistent format.
2. A Go programming language based and a Rust programming based ORM tools capable to retrieve 4 millions row of data from CSV datasets and

PostgreSQL database in sequential and concurrent manner.

3. Duplication, missing, corruption and inconsistency of data shall be eliminated with data cleaning.
4. The database shall capable to handle extra workloads and allow more clients to establish concurrent connection on perform transaction simultaneously.
5. Conduct performance comparison of sequential and concurrent programming with Go programming language on retrieving 4 millions row of data.
6. Conduct performance comparison of sequential and concurrent programming with Rust programming language on retrieving 4 millions row of data.
7. Conduct performance comparison between Go and Rust programming language on data retrieval.
8. A Go programming language based data migration tool capable to migrate 4 millions of data from legacy storage to normalized table within PostgreSQL database.
9. Several PL/pgSQL scripts capable to perform table creation, data manipulation and migrate data from legacy storage into normalized table within PostgreSQL database.
10. Ensure the migrated data are consistent and accurate.

# Chapter 2

# Literature Review

## 2.1 Sequencial Programming vs Concurrent Programming

Sequential programming involves process execution one after another [10] and have no linguistic design construct for concurrent computations. [11] The processes will only run after other is successful and executed chronologically in predetermined manner. [12] However, it's difficult to implement complex interaction and handle problems in parallel and concurrent environments with single-threaded. [13]

Concurrency had cause major turning point force in software development for developing concurrent software in order to exploit greater efficiency and performance optimization by fully utilize multiple core. To leverage the full power of hardware resource in software industry, concurrency and clouds will be the things every developer requires to deal with future software development and it is essential for both concurrent and distributed system. [14] Future generation computing system likely being developed by concurrent programming on multiprocessors. [15]

## 2.2 Concurrent Programming

Concurrent programming is form of computing where two or more threads cooperate to achieve common goals, inter-process communication and synchronization without require multi-processors. [16] Implementing concurrency into system requires imperative and functional language which allow programmer to take in control of concurrency by specifying step-by-step changes to variables and data structures in manipulation of data. [17] Therefore, concurrent programming language possess the ability to enable express concurrent computation easily by making synchronization requirements achievable and facilitate parallelism. Moreover, concurrent programming language possess programming notation, package and techniques for expressing potential parallelism and solving resulting synchronization and computer system communication problems. [4]

## 2.3 Distributed Programming

Concurrency and distributed programming often discuss together on implementing for a wide application of computer platforms from mobile devices to distributed servers. Distribute programming is form of computing where various source of parallelism running program on multiple machines simultaneously. It allow a distributed server make efficient use of network resources to communicate and coordinate in order to provide closer service for clients. [18] Concurrent programming is used to implement distributed process for real-time applications operate by microcomputer networks which possess distributed storage. The concurrent program is implemented into distributed server or storage in order to execute sequential processes simultaneously. Concurrent Pascal is possible to satisfy the efficiency, reliability and consistency of distributed storage. [19]

## 2.4 PostgreSQL

PostgreSQL is general object-oriented relational database management system that first possesses MVCC feature before Oracle. It is an open source object oriented relational database management system (OORDBMS) created by University of California [20] and currently maintained by the PostgreSQL Global Development Group with companies and contributors. PostgreSQL supports various concurrent programming language such as C, C++ and Java, etc and guarantees data consistency while performing concurrency transaction. [21] Other than that, PostgreSQL store multiple version of records in the database by keeping the latest version of tuple and garbage collects old records no longer required. [22]

The database is implemented with TelegraphCQ data flow system for processing continuous queries in data streaming environment. Research has found the open source database system possess extensibility feature and reusable component to improve adaptivity and concurrent read-write. [23] Ultimately, PostgreSQL is used to optimize pipeline on handle runtime update request for conventional data warehouse to process data analysis concurrent queries efficiently. The database system offers a modern feature to support adaptive query processing and maximize work sharing during execution. [24]

The advantage of PostgreSQL are listed as follow:

1. **Multi version concurrency control (MVCC).** The database system allows client to perform concurrent request and transaction to data and enforcing data consistency. [25] It provided support for concurrency model and designed for high volume environments with serializable transaction

isolation level to prevent dirty reads and better than row-level locking provided by several enterprise database systems such as MySQL. [26]

2. **Process-based.** PostgreSQL server is process-based and not threaded-based which increase robustness and stabilization during querying data compare to other database systems for this project. This can be explained by the difference between multiprocessing and multi-threading. A single thread die kills whole multi threaded environment dies but single process terminate will not affect other process running.

3. **Support Ubuntu OS.** PostgreSQL provides lifetime support for Ubuntu version. The database system repositories such as core database server (postgresql-9.5), client libraries and binaries (postgresql-client-9.5) and other additional modules (postgresql-contrib-9.5) are supported and consistent with various Linux distribution. [27]
4. **Security.** PostgreSQL make data processing more safety compare to direct retrieval with CSV because it is not open for modification by normal user.

## 2.5 Go language

Go's principle focus on simplicity, orthogonal, succinct and safe to provide its expressiveness to support efficient large scale programming, faster compilation speed and utilized multi-core hardware. [28] In the past, Go had been used to implement high-performance, scalable radio access system to evaluate its suitability and language functionality. [29]

The language had also utilized to assess text data processing in information system and mentioned Go is promising featuring native support for distributed applications. [30] Other than that, Go's concurrency primitives is used to implement an artificial intelligence and graph theory based sliding-puzzle game for Unix terminals. The language concepts and package are supportive to developed real-time notification delivery architecture with its what s. [31]

## 2.6 Rust language

Rust is a new and multi-paradigm programming language developed by Mozilla Research. [32] Earlier projects were using the Rust programming language to built several higher level abstractions on GPU kernels. They show how Rust advanced features enable to support both system-level concept and high-level operators on GPU computing. [33] Small model of RUST called Patina was experimented and study for claiming the language memory is safety without garbage collection by identify whether there are leaks during deallocating memory and ensure data initialized correctly on the runtime memory. [34]

## 2.7 Comparison of concurrent programming language concepts

Experimental design and demonstration are often conducted by the previous researcher to compare concurrent programming languages concepts with debugging existing system and writing correct new programs. [35] Structure embedding concepts in several concurrent programming languages has been examined by demonstrating mapping to a parallel composition to test its expressive power of these languages through results. [36]

Moreover, a general method is developed by previous research for comparing concurrent programming languages based on categories of language embeddings to obtain separation results. The programming language's properties affect the concept and performance of concurrent programming language. As an example, even though CSP and Actors possess common characteristic with non-compositional observable equivalence and interference free but CSP contains composition with hiding while Actors don't. [37]

In addition, expressive power of concurrent programming languages often compared by previous research to investigate how synchronization and logical control construction affect the efficiency of resulting word from three computational model. [38] Several conventional techniques and concurrent programming structures were analyze for implementing objects related to critical sections with concurrent programming languages. [39] Furthermore, previous researchers had proposed classification frameworks to study relevant elements of architecture description languages by present definition for comparing language components, connectors and configurations. [40]

Surveys is conducted on a preference of design and language features on 13 concurrent languages and found available architectural supports profoundly influence the language's style. The results indicate the concurrent feature of programming language will influence the intended use and application of the language. [41] In addition, previous research is conducted to compare implicit and explicit parallel programming with SISAL and SR to evaluate for programmability and performance. [42] Detailed performance measurements are presented with the comparison of various parallel architecture and measured with Beowulf-class parallel architecture. [43]

## 2.8 Comparison of Go and Rust language

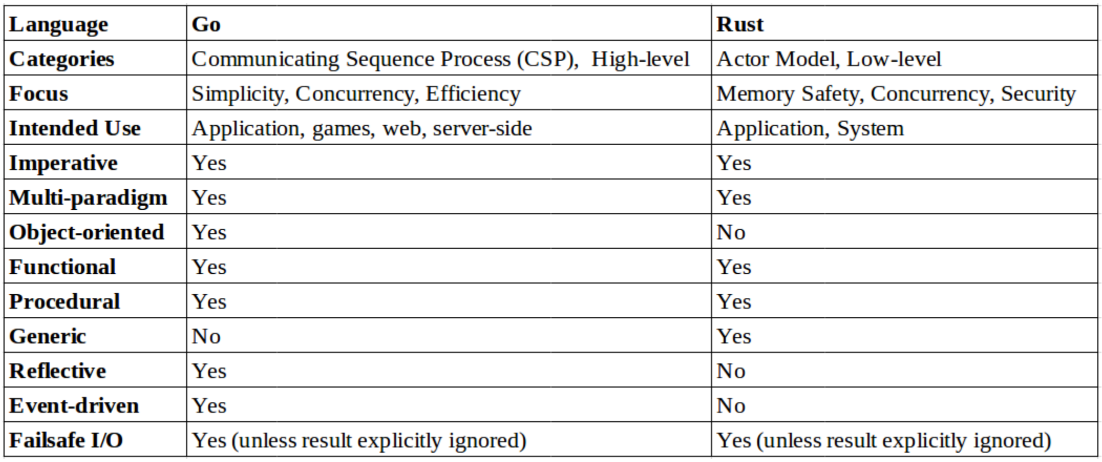
Go and RUST has start to gain popularity among the trends. [44] Rust and Go are also some of the developers most loved programming language. [45] The

Rust and Go programming languages are new programming languages for implementing concurrent and distributed based system. [46]

Go and RUST are both new concurrent programming language create after the year 2000. Go had become language of the year in Tiobe programming language ranking in 2009 and 2016. [47] Simultaneously, Rust won first place in most love programming language in Stack Overflow survey 2016 and 2017. [48]

Both concurrency programming languages support functional and imperative procedural paradigms. [49] [50]. Go is a CSP-based language provide rich support concurrency with goroutines and channel [51] but Rust is an actor model language focus on memory safety over performance. [52] Go and Rust often used to be compared with current software industry in concurrent computing implementation. [53]

Figure 2.5 shows characteristic and paradigm of Go and RUST programming language. All the language characteristic below will be discussed in the following subsection.



| **Language** | **Go** | **Rust** |
|---|---|---|
| **Categories** | Communicating Sequence Process (CSP), High-level | Actor Model, Low-level |
| **Focus** | Simplicity, Concurrency, Efficiency | Memory Safety, Concurrency, Security |
| **Intended Use** | Application, games, web, server-side | Application, System |
| **Imperative** | Yes | Yes |
| **Multi-paradigm** | Yes | Yes |
| **Object-oriented** | Yes | No |
| **Functional** | Yes | Yes |
| **Procedural** | Yes | Yes |
| **Generic** | No | Yes |
| **Reflective** | Yes | No |
| **Event-driven** | Yes | No |
| **Failsafe I/O** | Yes (unless result explicitly ignored) | Yes (unless result explicitly ignored) |

FIGURE 2.1: Comparison of Go and Rust language characteristic

### 2.8.1 Comparison of language categories and focus

Go is a high-level language focus on simplicity, reliability and efficiency. The language is designed with communicating sequential process (CSP) to express concurrency based on message passing channels. The processes and messages communicate via goroutine and gochannel within a shared memory. [54] The language is intended to use for building web application programming interface (API) or networking application such as TCP or HTTP server to handle request.

Go possess simple syntax, garbage collector and runtime which allow developer to increase code readability and implement concurrency easier. However, Go is lack of language extensibility which leads to a limitation on implement manual memory management. [55]

Rust is a low-level language focus on memory safety, security and fault tolerance. The language designed with actor model concurrent programming language that use “actors” as fundamental agent on message passing. The actor takes input, send output after performing functions. [56] The processes and message communicate point-to-point via actors in a consistent state. The language intended use for system programmings such as building game engines, driver and embedded devices.

Rust doesn’t possess garbage collection and runtime which promote extensibility and deterministic on implement memory management. [57] However, Rust has much inherent complexity of syntax and semantics and has a high learning curve for a developer.

### 2.8.2 Similarities of Go and Rust language

The similarities of both languages are discussed as follow:

1. **Imperative.** Go and Rust are imperative programming paradigm where a value can be assigned into a variable to perform operation on information located in memory. Moreover, these languages allow declaration of a variable to store the results in memory for later use, affect the global state of a variable.
2. **Functional.** Go and Rust language can be written with mathematical functions to express control flow by combining function calls. The function avoid changing global state of variable.
3. **Procedural.** Go and Rust language can be written into statement structured and divided into function. The function known as procedure takes input processes it and produces output.
4. **Multi-paradigm.** Go and Rust language are support various programming paradigm and provide developer to use suitable programming style to develop a program to achieve project objectives.
5. **Failsafe I/O and callbacks.** Go and Rust language compiler warn error or throw an exception if the system calls fail. Go language throw errors if developer doesn't use the declare function or variable and Rust language does not compile if found any dangling pointers.

### 2.8.3 Difference between Go and Rust language

The difference between both languages are discussed as follow:

1. **Object-oriented.** Go language support object-oriented programming with struct and interface. However, Rust is not an object-oriented language result of the idiomatic language and its appearance in an OO language. [58]
2. **Generic.** Go language is lack of generic where the compiler doesn't allow declared a function or variable written in to-be-specified-later types await to be instantiated when needed for a specific purpose. However, Rust is possible to specify generalized function and avoid codes rewriting.
3. **Reflective.** Go language possess the ability to observe and modify type, object, function execution on runtime by import "reflect". However, Rust doesn't have reflection.
4. **Event-Driven.** Go is a high-level language enable write application respond to demand and expectation from mobile devices, multicore architectures and cloud computing environments. However, Rust is a low-level language prevent the flow of program interrupt by an event from user actions to enforce security and safety.

## 2.9 Ubuntu 16.04.03 LTS 64-bit OS

Ubuntu OS is an open source operating system with Linux distribution system and based on Debian architecture which provides long-term support (LTS) on security and fixes. [59] The advantage of Ubuntu operating system are described below:

1. **Free and customizable.** The openness of using Ubuntu OS offers a wide range of choices for the programmer to conduct development activities with Linux terminal. The APT packaging system allows developer to manage software and programming languages package efficient compared to Window operating system. The OS provides freedom in customization for a developer to catered different sets of need with source access and root permission to meet project requirements.

2. **Security.** The system files are owned by root in Ubuntu OS and not accessible by casual user, malware and third party software without root privilege. [60] As the operating system is maintained and contributed by vast amount of developer and programmer due to its open source and environment, the bugs are fixed efficiently with regular updates and provide less vulnerability for the attacker to exploit the system. [61] The key factors underline within Ubuntu security provide sufficient statement to prove Ubuntu is more secure than Window or Mac OS on this project.

3. **Consistent.** Ubuntu OS provide excellent consistent from front-end (UIUX) to back end. The user interface and user experience of Ubuntu operating system increase usability and efficiency in development,

maintenance and deployment activities in the different version.

4. **Stable and Reliable.** UNIX preceded and outshine MS-DOS kernel with hardware abstraction, security model, resource management and various services that ran as background processes. [62] Ubuntu promotes multitasking and multi-user which is suitable and ideal for this project to conduct concurrent and distributed processing activities with PostgreSQL. Last but not least, MS-DOS is an image loader system that preload memory addresses without memory or resource management quickly leads to BSOD and data corruption during data processing.

## 2.10 Debugging tools

Debugging could be painful for a software engineer to monitor and identify the performance of applications running in concurrent and distributed on sophisticated operating systems like Ubuntu.

Debugging with printf() for program bring many disadvantages and limitation during concurrency programming. The function could consume much memory in the multi-threaded environment because it's not lightweight and thread safety. [63] Moreover, it is not an efficient way to identify problems occurs related to memory allocation or interruption.

Therefore, debugger is used in this project to understand event or consequence happens in a running software system without consuming the enormous amount of memory. Simultaneously, it helps developer to save times on finding coding and logic errors in source codes. [64]

### 2.10.1 GDB Debugger

GDB is a build in GNU debugger for UNIX systems to debug programs to obtain information of root cause that cause the program to fail. [65] GDB allows set breakpoints and watchpoints on certain functions and print values during the program execution with terminal interface. Unfortunately, GDB possess limitation on finding bugs cause by memory leakage and compile errors.

## 2.11 Eclipse for Parallel Application Developers Oxygen Release (4.7.0) IDE.

Eclipse is an integrated development environment create and maintain by Eclipse Open Source Project teams. The Eclipse Oxygen release possess better functionality and performance for a developer to manage, build and deploy software system. The advantage of Eclipse IDE are listed as follows:

1. **Auto Completion.** The openness of using Ubuntu OS offers a wide range of choices for the programmer to conduct development activities with Linux terminal. The APT packaging system allows developer to manage software and programming languages package efficient compared to Window operating system. The OS provides freedom in customisation for a developer to catered different sets of need with source access and root permission to meet project requirements.

2. **Integrated Environment.** The system files are owned by root in Ubuntu OS and not accessible by casual user, malware and third party

software without root privilege. [60] As the operating system is maintained and contributed by vast amount of developer and programmer due to its open source and environment, the bugs are fixed efficiently with regular updates and provide less vulnerability for the attacker to exploit the system. [61] The key factors underline within Ubuntu security provide sufficient statement to prove Ubuntu is more secure than Window or Mac OS on this project.

3. **Debugger.** Ubuntu OS provide excellent consistent from front-end (UIUX) to backend. The user interface and user experience of Ubuntu operating system increase usability and efficiency in development, maintenance and deployment activities in the different version.

4. **Plugins.** UNIX preceded and outshine MS-DOS kernel with hardware abstraction, security model, resource management and various services that ran as background processes. [62] Ubuntu promotes multitasking and multi-user which is suitable and ideal for this project to conduct concurrent and distributed processing activities with PostgreSQL. Last but not least, MS-DOS is an image loader system that preload memory addresses without memory or resource management quickly leads to BSOD and data corruption during data processing.

## 2.12 Chapter Summary

The finding for literature review is concurrent programming language possess specific built-in notation, package and functions to build parallel and distributed application. PostgreSQL is suitable for this project because it possesses MVCC that able handle concurrent request with good adaptivity and accuracy. Golang and Rust are concurrent programming language support multi-paradigm programming with multiprocessing and multithreading. Go language focused on simplicity while Rust language focuses on security. Both programming languages invented with different model and concepts for a different purpose.

Concurrent language is often compared and evaluated with configuration, categories and architecture to obtain performance and expressive power. The language's feature is essential to prove the performance of specific concurrent language. Debugging tools play a main role on observing processes and threads activities during the development and debugging activity to ensure program's execution is observed and error are discovered.

# Chapter 3

# Project Design

## 3.1 Phase 1

### 3.1.1 Introduction

The primary focus of Phase 1 is implement prototype to prove theoretical concepts of the domain to research in this project. Requirements are listed as follow:

1. To acquire free large data set for big data processing.
2. To ensure data set acquired from the website are free, consent and clean with Devil Advocation Test.
3. A program will be implemented in RUST and Go programming language as a proof-of-concept (POC) that CSV raw data is capable of importing into PostgreSQL database.

4. A program will be implemented with Go programming language as POC that PostgreSQL database transaction can be sequential and concurrent.

### 3.1.2 Data Collection

The project is required to work with large data sets to utilize infrastructure and processing power of GO and RUST concurrent programming language. Data collection is conducted to identify of company recruitment preferences on higher education graduates of different subjects in the UK with basic company and LEO datasets. Data collected is required to be clean and able to solve interesting problem or question.

The characteristic of free, consent and licensed data sets acquired from UK government website provider (data.gov.uk) are as follow:

| **No** | **Name of Datasets** | **Column** | **Rows** | **Size** |
|---|---|---|---|---|
| 1. | Longitudinal Educations Outcomes (LEO) | 21 | 32706 | 1.8 GB |
| 2. | Basic Company Profile (Company) | 55 | 3595702 | 667.5 MB |
| 3. | National Statistics Postcode Lookup (NSPL) | 35 | 1754882 | 4.2 MB |

Table 3.1: The detail of datasets obtained.

The file format of all large dataset obtained are Comma Separated Values (CSV) format which the information is organized with one record as one line and each field is separated by comma (,). CSV format is used for data processing in this project because it is human readable and simple to be parse. It can be handle using PostgreSQL database and retrievable by programs.

#### 3.1.2.1 Longitudinal Education Outcomes (LEO) dataset

The data set focus on employment and earnings outcome of Bachelor's Degree graduate in Great Britain after five years. It contains information about students include personal characteristics, education or qualification achieved, employment and income earnings. The data dictionary of longitudinal education outcome is created and placed in Appendix I.1.1.

#### 3.1.2.2 Basic Company dataset

The data set possesses up-to-date basic companies information on UK register. It contains company names, annual returns filing dates, location details, account and basic information about mortgage and business changes. The data dictionary of basic company dataset is created and placed in Appendix I.1.2.

#### 3.1.2.3 National Statistics Postcode Lookup (NSPL) dataset

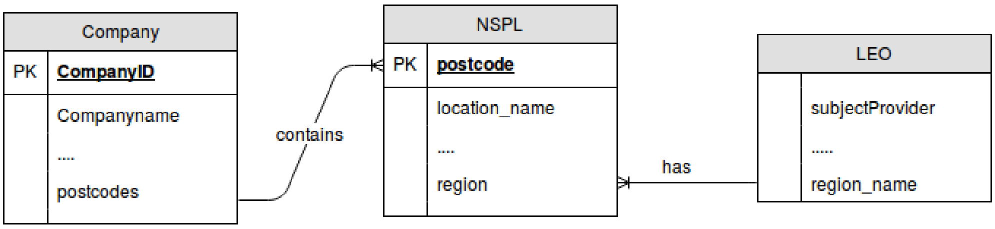


Figure 3.1: Entity Relationship Diagram

As postcode data for every location on earth is unique. Company data sets possess **postcode** field in the business address, but LEO dataset do not have the **postcode** field which leads to difficulty of defining a relationship between these two datasets. Figure above show NSPL dataset serves as a linker to map **region** column from LEO data to link with **postcode** column found in company datasets.

The data set possesses current postcode for the United Kingdom. It contains information relates postcode number, location, country name, parliamentary constitution, electoral and other geographical details. The data dictionary of National Statistics Postcode Lookup (NSPL) dataset is created and placed in Appendix I.1.3.

### 3.1.3 Data Validation

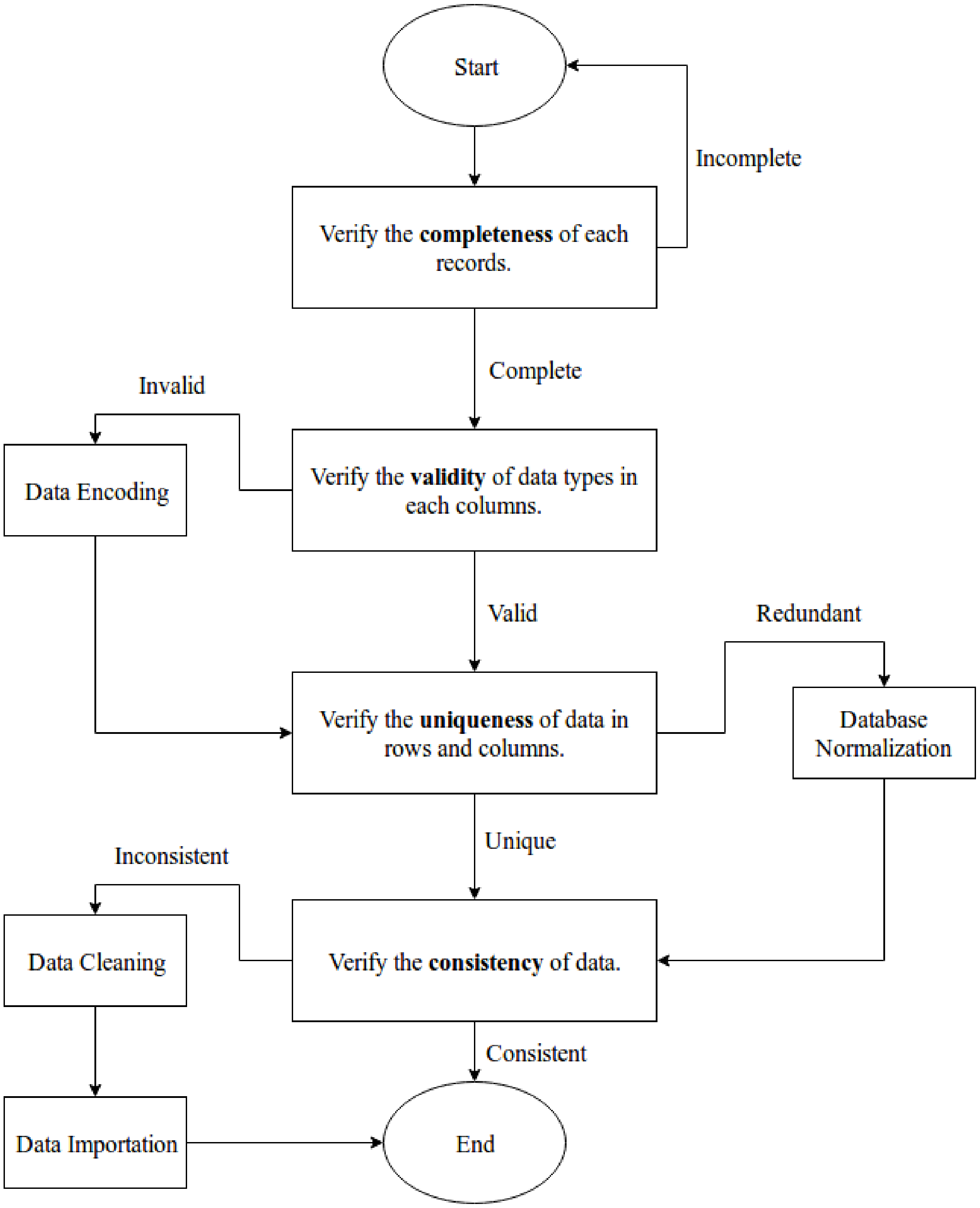

Start
Incomplete
Verify the **completeness** of each records.
Complete
Invalid
Data Encoding
Verify the **validity** of data types in each columns.
Valid
Redundant
Verify the **uniqueness** of data in rows and columns.
Database Normalization
Unique
Inconsistent
Data Cleaning
Verify the **consistency** of data.
Consistent
Data Importation
End

FIGURE 3.2: Data Validation Procedure Flowchart

Data validation is conducted to inspect the quality dimension of data sources acquire in Data Collection (Section 3.1.2) to prevent corruption, inconsistency and conflicts during importing, using and processing. It is performed to ensure the data acquired are clean and in excellent quality.

The important steps taken on validation of data are shown in Figure 3.5. The **completeness** of datasets will be examine to assures the characteristic of data fulfill Comma Separate Values (CSV) standard and requirement. The common test performed during data completeness check are using aggregate functions such as max, min or counts. [66]

Furthermore, the **validity** of data types in each columns are measured to prevent incompatible data types during Data Importation, Object Relational Mapping (ORM) and Data Migration. The types of data stored in each columns of obtained datasets shall be identify to describe suitable data type for Database Definition Language (DDL) during database table creation. As an example, the alphanumeric and text field are usually defined as VARCHAR and field contains only number will be declared as INTEGER.

In addition, the **uniqueness** of records will be verify to discover wasteful and duplication of data. The data redundancy indicates same piece of data are exist in multiple place. [67] This condition will results in waste of space, data inconsistency and violates data integrity. If the duplication of data is discovered, database normalization will be performed to eliminate the duplication of records.

Last but not least, the **consistency** of data will be analyze to ensure datasets obtained are conform to specific standards and meet requirements. The data consistency check shall be performed during data preparation to inspect discover missing, corrupted or invalid data in record. The conformity and consistency of data in specific column should be handled in wariness to prevent affect the outcomes and efficiency of data processing. If the data is found inconsistent, Data Cleaning and Data Importation will be conducted to fix the defects discovered in the datasets.

### 3.1.4 Performance Benchmarking

To conduct a comparison between Go and RUST language, benchmarking plays an important role to achieve fairness in compare performance and expressive power of language.

The component that are benchmarked are listed below:

1. **SQL Queries run on program.** Go and Rust program execute the same amount of database retrieval query to achieve the fairness of comparison.

2. **Table configurations.** The space of table of this project should be same for Go and Rust program to test the performance.

3. **Hardware configurations.** Both Go and Rust program are required to run on same hardware configuration to achieve fairness of comparison on performance.

### 3.1.5 Database Retrieval Program

#### 3.1.5.1 Phase 1 System Context Diagram

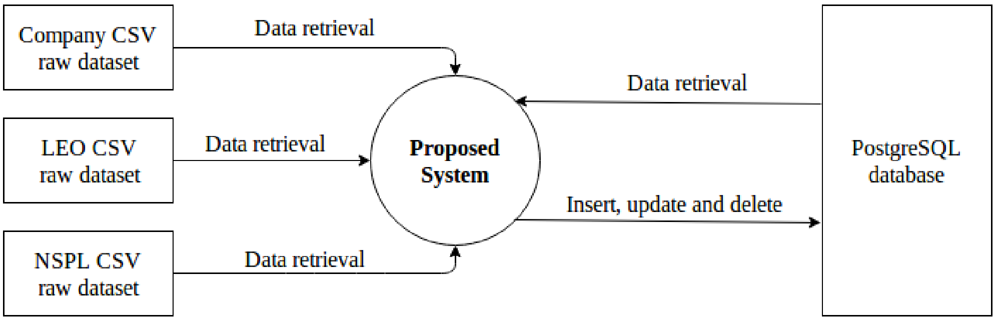


FIGURE 3.3: Phase 1 System Context Diagram

System context diagram provide high level view that defines relationship between proposed system with external entities. The proposed system is written in Go and Rust programming language with sequential and concurrent computing. The system shall process raw dataset stores in different nodes and dataset stores in PostgreSQL database. Moreover, the system should process data from raw CSV dataset and PostgreSQL database in sequential and concurrent manner.

#### 3.1.5.2 Phase 1 Block Diagram

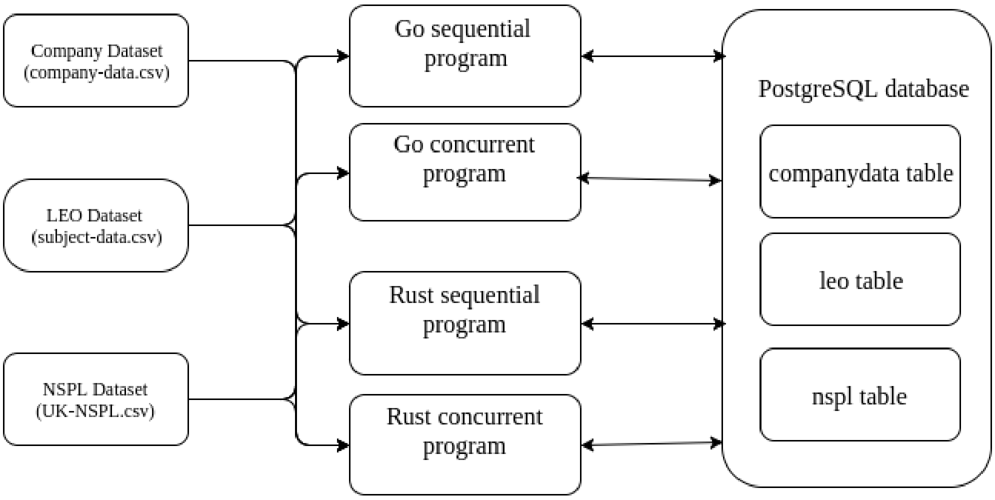


FIGURE 3.4: Phase 1 Block Diagram

The block diagram provides a high-level overview of importation CSV into PostgreSQL with Go and Rust program. The large dataset store is store in different nodes with CSV format. Data stores in PostgreSQL database and raw CSV data at different nodes will be processed by Go and Rust program with sequential and concurrent manner. The database table is created with query in the terminal before Go and Rust program is executed.

### 3.1.6 PostgreSQL Database Retrieval with Go and Rust program

#### 3.1.6.1 Phase 1 Sequential Program Flowchart

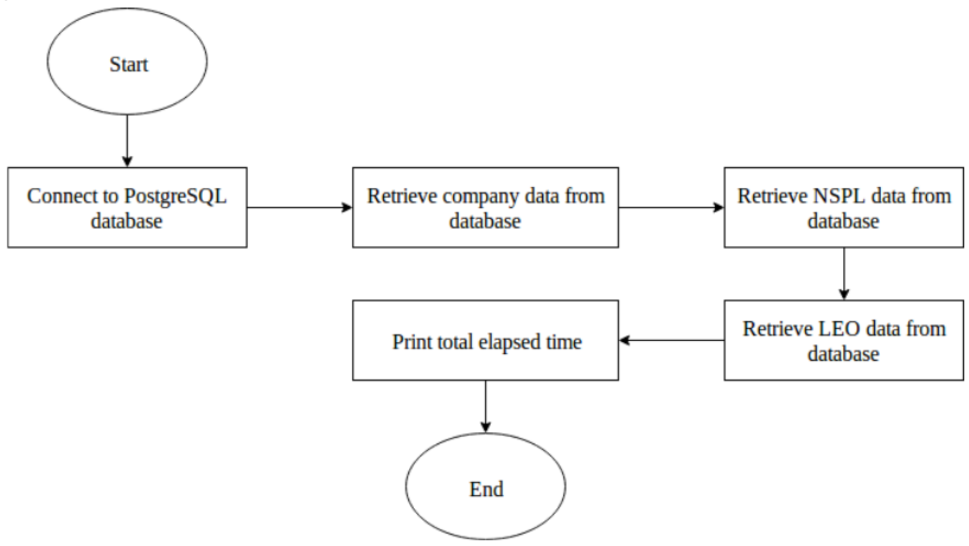

Start
Connect to PostgreSQL database
Retrieve company data from database
Retrieve NSPL data from database
Retrieve LEO data from database
Print total elapsed time
End

FIGURE 3.5: Phase 1 Sequential program flowchart

The flowchart provides a high-level view of concurrent manner during data retrieval in PostgreSQL with Go and Rust program. The program first establishes connection with PostgreSQL database with a connection string. Afterwards, it will retrieve a different set of data from various database table concurrently. The total elapsed time for entire program execution will be print.

#### 3.1.6.2 Phase 1 Concurrent Program Flowchart

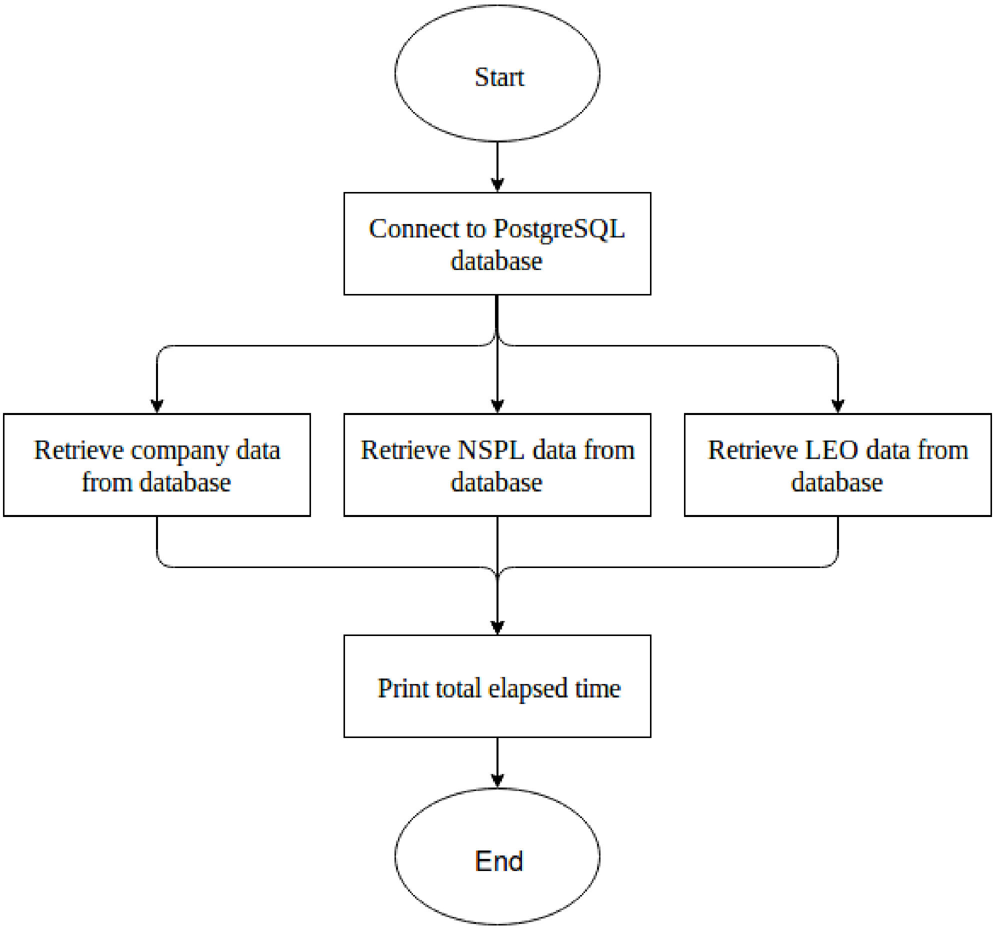

Start
Connect to PostgreSQL database
Retrieve company data from database
Retrieve NSPL data from database
Retrieve LEO data from database
Print total elapsed time
End

FIGURE 3.6: Phase 1 Concurrent program flowchart

The flowchart provides a high-level view on concurrent manner during data retrieval in PostgreSQL with Go and Rust program. The program first establish connection with PostgreSQL database with connection string. Afterwards, it will retrieve different set of data from different database table in concurrent manner. The total elapsed time for entire program execution will be print.

### 3.1.7 Raw CSV Data Retrieval with Go and Rust program

#### 3.1.7.1 Phase 1 Sequential Program Flowchart

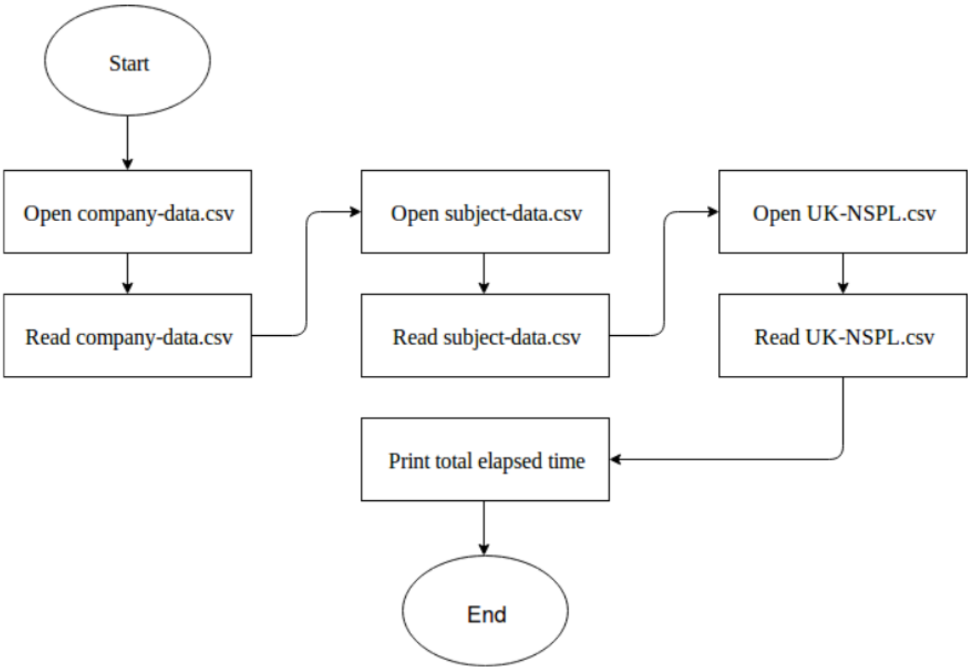


Figure 3.7: Phase 1 Sequential program flowchart

The flowchart provides a high-level view on sequential manner on reading CSV file with Go and Rust program. The program will open csv file and read containing data concurrently. The total elapsed time for entire program execution will be print.

#### 3.1.7.2 Phase 1 Concurrent Program Flowchart

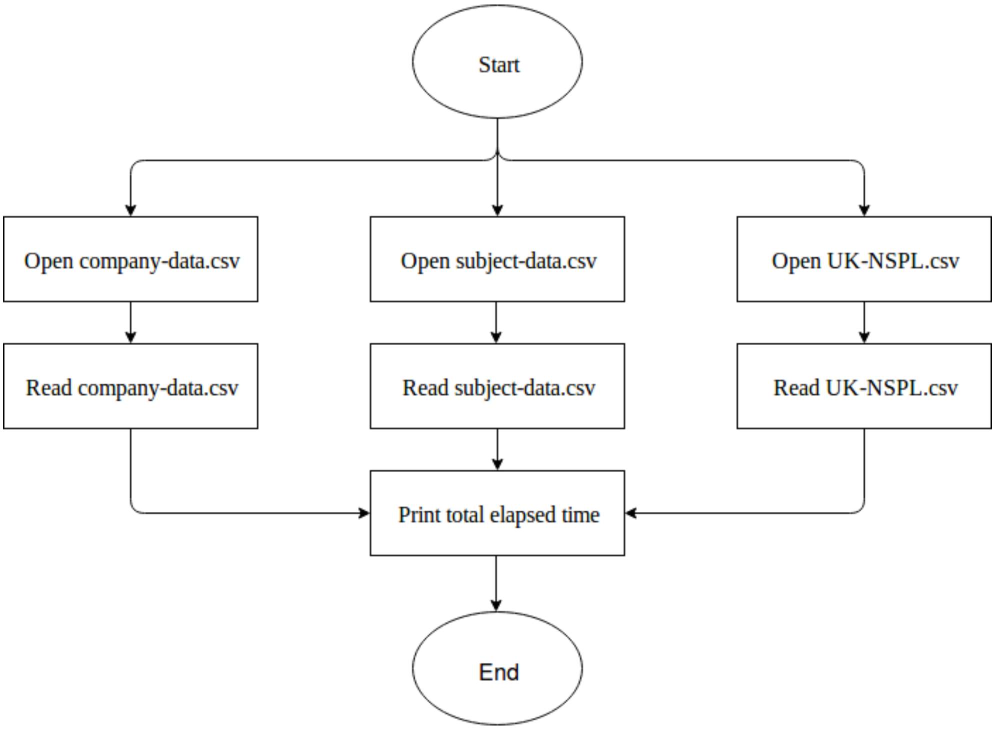


Figure 3.8: Phase 1 Concurrent program flowchart

The flowchart provides a high-level view on concurrent manner on reading CSV file with Go and Rust program. The program will open csv file and read containing data in particular order of sequence. The total elapsed time for entire program execution will be print.

### 3.1.8 Proof of Concept in Phase 1

#### 3.1.8.1 Phase 1 Deployment Diagram

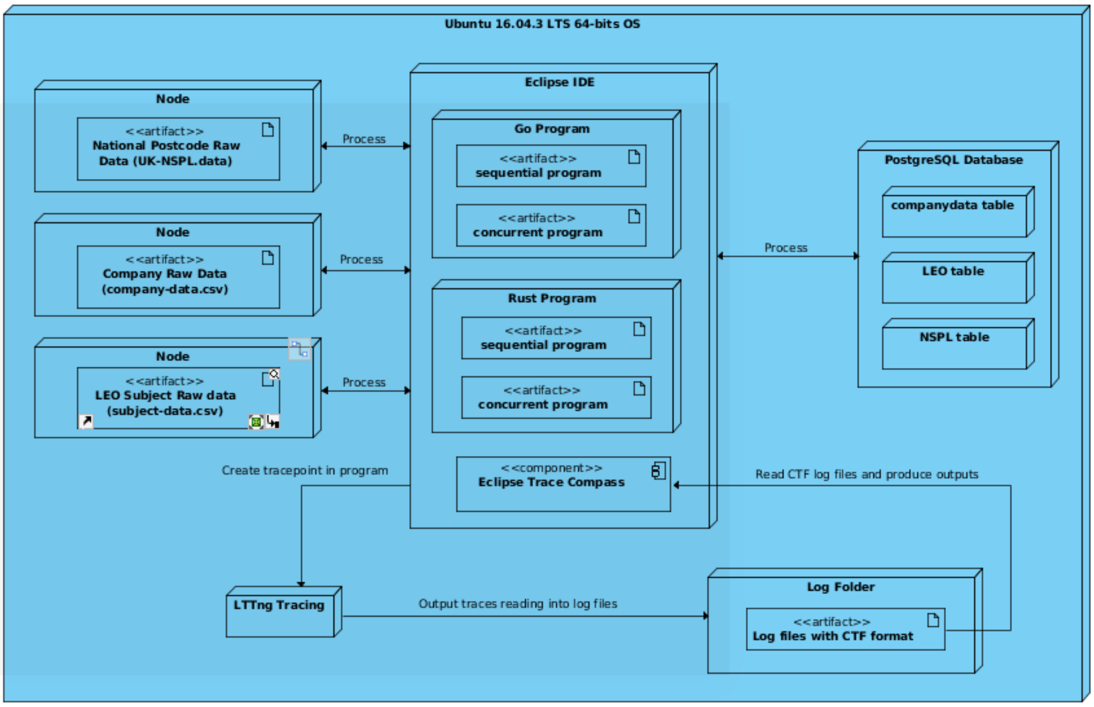


Figure 3.9: Phase 1 Deployment Diagram

The deployment diagram describes the proof of concept of phase 1 in specification level and overall architecture of the project. Three database table is created in PostgreSQL database prepare to be processed. Simultaneously, three large data sets are stored in different nodes await to be process or retrieved. The Go and Rust program are written in sequentially and concurrently to process data from CSV file or PostgreSQL database system.

## 3.2 Phase 2

### 3.2.1 Introduction

Figure below shows Data Processing Cycle to provide an overview of activities carried out to process big data with the utilization of concurrent programming language and Structure Query Language (SQL).

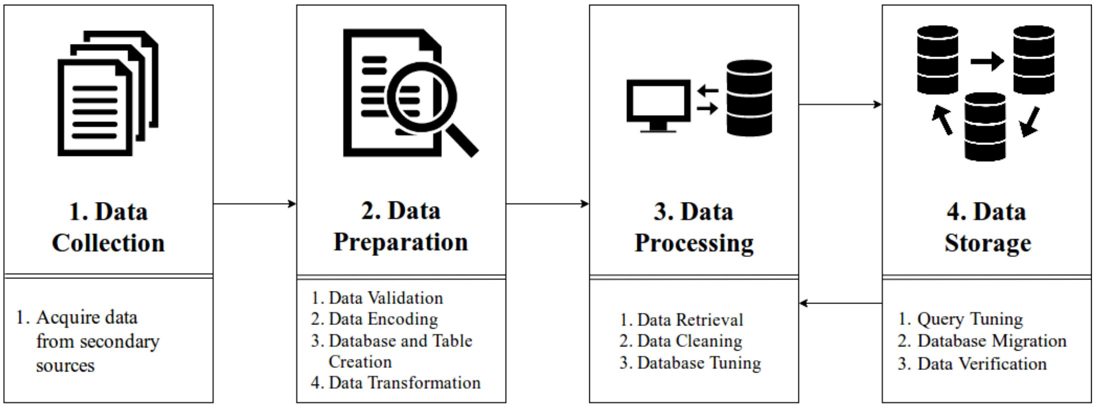


FIGURE 3.10: Data Process Cycle

In Phase 2, we have established an extensive understanding on concurrent language characteristic by utilized the languages' feature on each activity in data processing cycle. The requirement as listed as follow:

1. Data encoding will be conducted with stream editor to convert dirty data into consistent format.
2. Data transformation will be conducted to extracted data from CSV file and import into PostgreSQL database for data handling.
3. Database normalization will be perform to eliminate data redundancy and improve data integrity.

4. The structure of database schema and object (user and tables) will be created with scripts written in Data Definition Language (DDL) of PL/pgSQL (Procecural Language/PostgreSQL).
5. A **sequential** and **concurrent** program will be implemented with Go programming language as an Object Relational Mapping (O/R mapping tool) to convert raw data from CSV data sources into object model, the performance execution will be recorded and compared.
6. A **sequential** and **concurrent** program will be implemented with Go programming language as ORM tool to convert data retrieve from PostgreSQL database into object model, the performance execution will be recorded and compared.
7. A **sequential** and **concurrent** program will be implemented with Rust programming language as ORM tool to convert raw data from CSV data sources into object model, the performance execution will be recorded and compared.
8. A **sequential** and **concurrent** program will be implemented with Rust programming language as ORM tool to convert data retrieve from PostgreSQL database into object model, the performance execution will be recorded and compared.
9. Data cleaning will be performed on CSV raw data to eliminate missing records and standardize the fields in common format.
10. Database tuning will be conducted to configure PostgreSQL database's environment for performance optimization on processing large-scale data and handling workloads.
11. A **sequential** and **concurrent** program will be implemented with Go

programming language as data importation tool to export Company raw data from CSV data sources and import into PostgreSQL database.

12. Query tuning will be conducted to increase query execution performance on data processing.
13. Several **concurrent** program will be implemented with Go programming language as data migration tool to transfer company and NSPL data from legacy storage into normalized table within PostgreSQL database.
14. Data Manipulation Language (DML) scripts will be written with PL/pgSQL to transfer raw data from legacy storage into normalized table within PostgreSQL database.
15. Data verification will be conducted with UNIX command line to check the accuracy and consistency of database records after the data migration is complete.

## 3.3 Data Encoding

### 3.3.1 Phase 2 Architecture Diagram

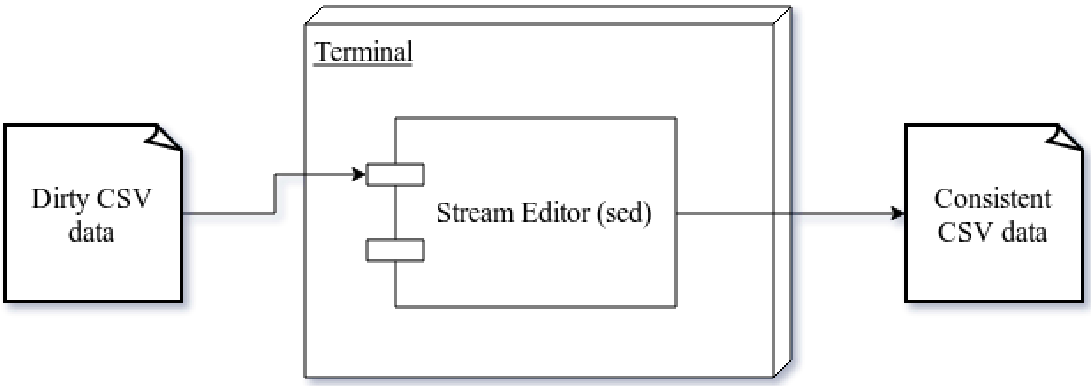


Figure 3.11: Data Encoding Architecture Diagram

Data encoding is a conversion of records or fields into specialized format for efficient transformation, importation and migration. [68] Figure 3.14 shows an architecture diagram that describe a high-level view of data encoding flow. The sed stream editor provide powerful feature to perform editing operations coming from a file to remove inconsistency data. [69]

The stream editor allow developer to make editing decisions by calling the commands on terminal. It consumes the dirty raw data as input file and perform text substitution line-by-line based on the text patterns of regular expressions provided in the commands. Ultimately, the encoded file will be output and store into the same directory.

## 3.4 Data Transformation

### 3.4.1 Phase 2 Architectural Diagram

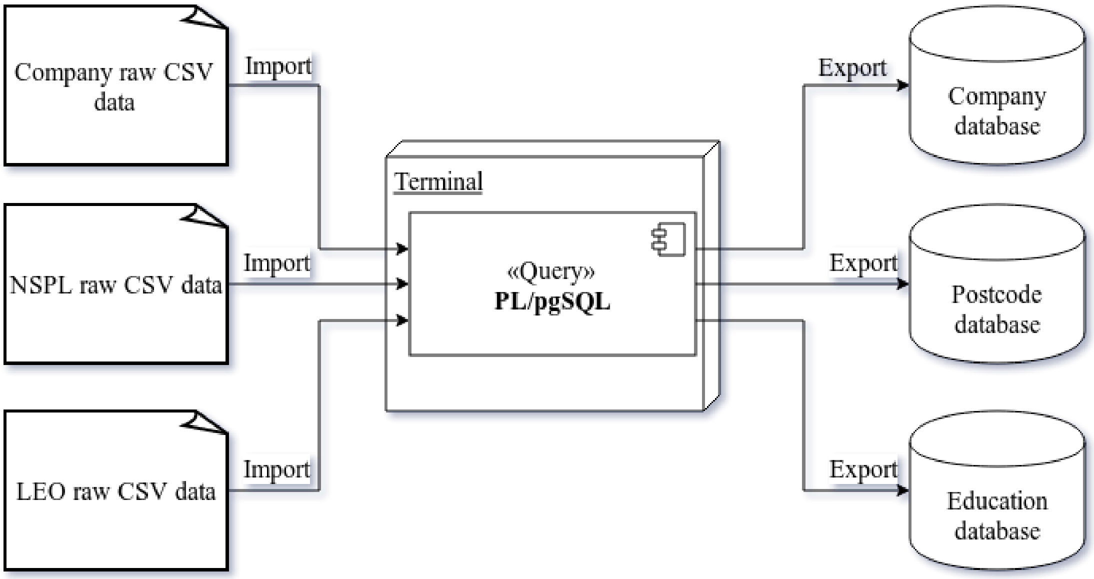


Figure 3.12: Data Transformation Architectural Diagram

Data transformation is the process of converting one format to another by extracting from source application into data warehouse. [70]

Figure 3.15 shows the architectural diagram of data transformation process in this project. After the data inconsistency is eliminated with data encoding (performed in Section 3.3), the data in CSV format is extracted and import into PostgreSQL database with PL/pgSQL commands in terminal environment.

## 3.5 Data Retrieval

### 3.5.1 Phase 2 Deployment Diagram

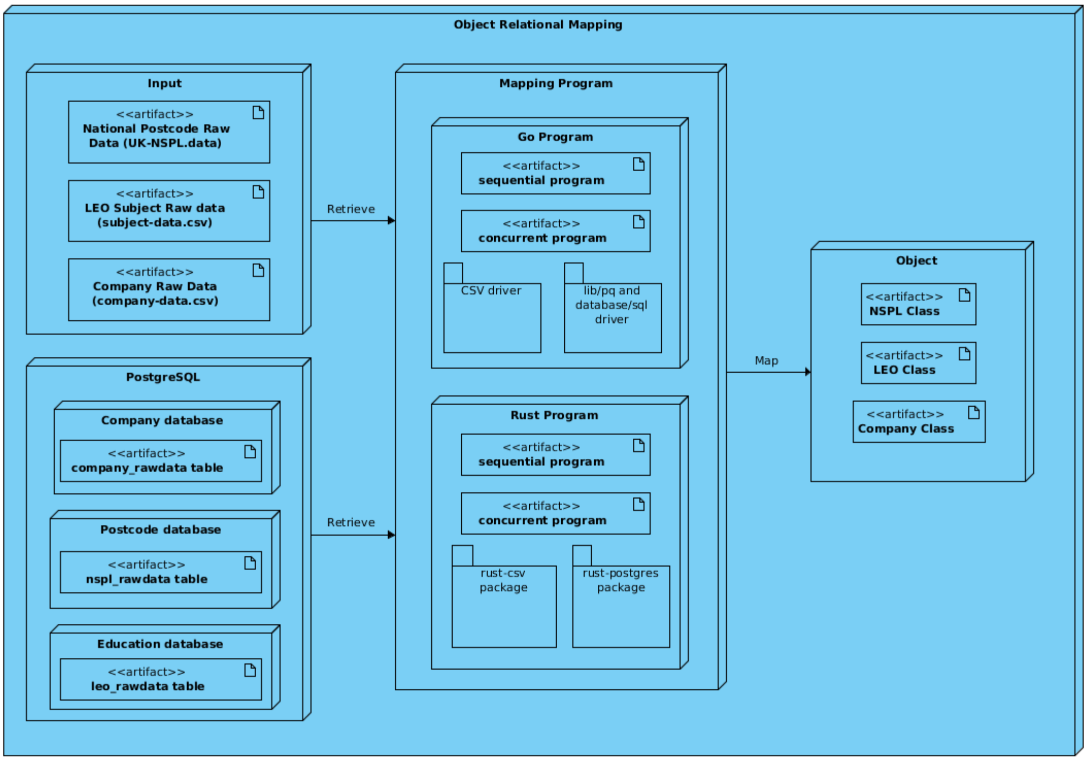

Object Relational Mapping
Input
<<artifact>> National Postcode Raw Data (UK-NSPL.data)
<<artifact>> LEO Subject Raw data (subject-data.csv)
<<artifact>> Company Raw Data (company-data.csv)
PostgreSQL
Company database
<<artifact>> company_rawdata table
Postcode database
<<artifact>> nspl_rawdata table
Education database
<<artifact>> leo_rawdata table
Retrieve
Retrieve
Mapping Program
Go Program
<<artifact>> sequential program
<<artifact>> concurrent program
CSV driver
lib/pq and database/sql driver
Rust Program
<<artifact>> sequential program
<<artifact>> concurrent program
rust-csv package
rust-postgres package
Map
Object
<<artifact>> NSPL Class
<<artifact>> LEO Class
<<artifact>> Company Class

FIGURE 3.13: Data Retrieval with ORM Deployment Diagram

Object-Relational Mapping (ORM) is a technique to manipulate data from database with object-oriented paradigm. The data retrieve from CSV data sources and PostgreSQL database will be convert into object model to ease the manipulation of data in discipline manner. [71] The approach increase usability, flexibility and improve data handling for Data Cleaning and Data Migration.

Figure 3.16 shows the ORM deployment diagram that provide graphic representation of mapping between object and data with mapping program written in Go and Rust programming language. In this project, we will construct our own ORM tools tool for data retrieval from CSV file and PostgreSQL database with the assistance of CSV package driver, PostgreSQL driver and built-in SQL library from respective language.

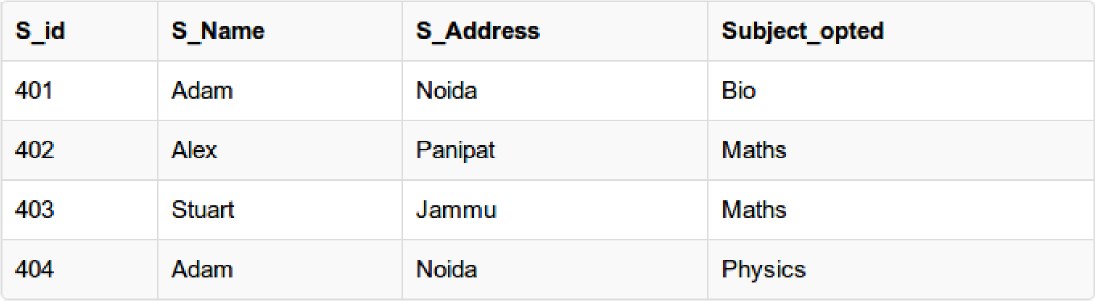
All the rows of data will be retrieved from PostgreSQL database and CSV file with Go and Rust's ORM to conduct performance comparison between sequential and concurrent execution and concurrent programming languages' expressive power. The results will be recorded and compared.

## 3.6 Data Cleaning

### 3.6.1 Introduction

Data cleaning is the action of detecting and removing missing, incomplete and data redundancy within database. [72] The inconsistencies and incorrect records will be detected in the datasets obtained from the secondary sources because we have lack of control over the data quality.

Data redundancy occurs within a data storage when same piece of data exists in two separate places or two different fields within a single database. Database without normalization will cause updation, deletion and insertion anomalies. The table below is used to understand these the impact of these anomalies on causing data inconsistencies.



| S_id | S_Name | S_Address | Subject_opted |
|---|---|---|---|
| 401 | Adam | Noida | Bio |
| 402 | Alex | Panipat | Maths |
| 403 | Stuart | Jammu | Maths |
| 404 | Adam | Noida | Physics |

Figure 3.14: Student table without normalization

1. **Insertion anomaly.** If student don't enroll any subject and **subject** is a mandatory field, the records cannot be insert into the database without the presence of other attributes or columns.
2. **Deletion anomaly.** If specific student willing to drop a subject, the entire records are forced to delete. As a result, certain attributes or part of the records are lost due to deletion of specific attributes without awareness which leads to missing data.
3. **Updation anomaly.** To update student address in the table, the entire **address** column are required to be updated. If the duplicate records in the database are partially updated, it will leads to data inconsistency.

Therefore, **database normalization** and **data standardization** is conducted to improve the quality and reliability of the datasets.

## 3.6.2 Database Normalization

### 3.6.2.1 Introduction

Data normalization is conducted to eliminate data redundancy and improve data integrity. The mentioned method is an approach to remove all the data anomalies and recover the database into consistent state. [73] Normalization is a multi-step approach and require rules to organize data into tabular forms and define relationships among them. The normalization rules and description are listed as follow:

1. **First Normal Form (1NF).** The rule required to eliminate repeating groups, identify primary key and discover **partial dependencies** or **transitive dependencies** among column by determine the determinant of the records.
2. **Second Normal Form (2NF).** The rule required to create new table with primary key assigned for **partial dependencies** elimination.
3. **Third Normal Form (3NF).** The rule required to create new table with primary key assigned for transitive dependencies elimination.

Relational database design is conducted to define entities, attributes, relationships and keys to fulfill normalization rules on eliminating data redundancy. The information contains in raw data are divided and separated into specific table and establish relationship among them to form an organized database. Ultimately, naming conventions and standards are used to form table to increase the usability and maintainability of database.

#### 3.6.2.2 Phase 2 Normalized Company Entity Relationship Diagram

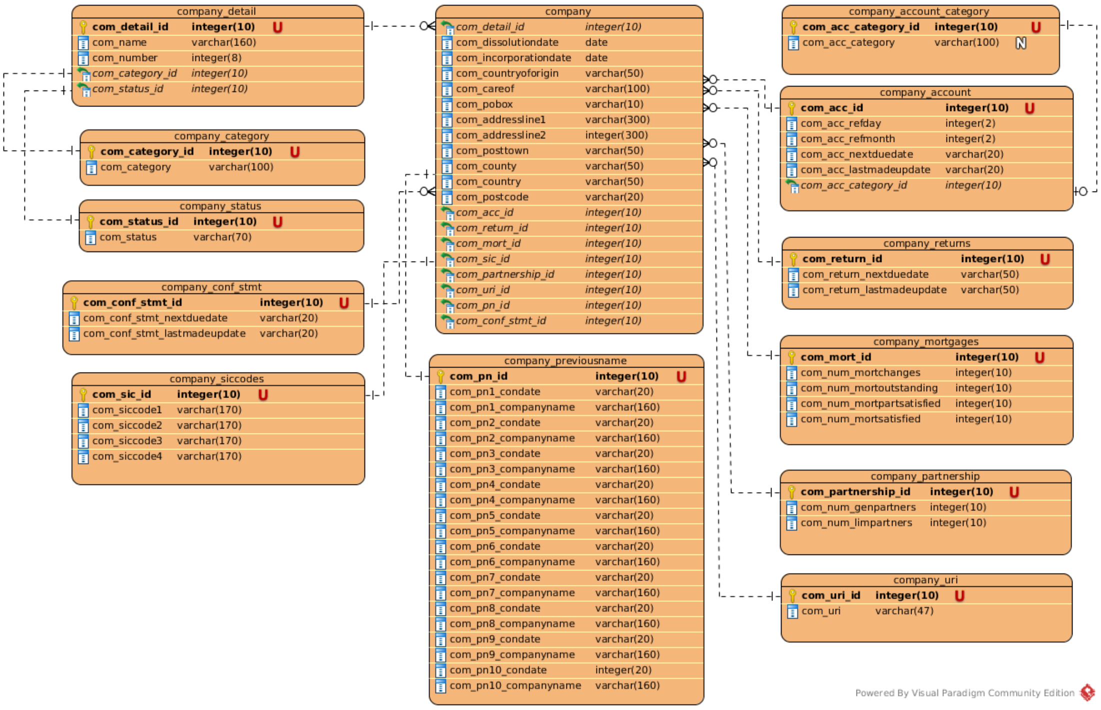

company_detail
com_detail_id integer(10) U
com_name varchar(160)
com_number integer(8)
com_category_id integer(10)
com_status_id integer(10)
company_category
com_category_id integer(10) U
com_category varchar(100)
company_status
com_status_id integer(10) U
com_status varchar(70)
company_conf_stmt
com_conf_stmt_id integer(10) U
com_conf_stmt_nextduedate varchar(20)
com_conf_stmt_lastmadeupdate varchar(20)
company_siccodes
com_sic_id integer(10) U
com_siccode1 varchar(170)
com_siccode2 varchar(170)
com_siccode3 varchar(170)
com_siccode4 varchar(170)
company
com_detail_id integer(10)
com_dissolutiondate date
com_incorporationdate date
com_countryoforigin varchar(50)
com_careof varchar(100)
com_pobox varchar(10)
com_addressline1 varchar(300)
com_addressline2 integer(300)
com_posttown varchar(50)
com_county varchar(50)
com_country varchar(50)
com_postcode varchar(20)
com_acc_id integer(10)
com_return_id integer(10)
com_mort_id integer(10)
com_sic_id integer(10)
com_partnership_id integer(10)
com_uri_id integer(10)
com_pn_id integer(10)
com_conf_stmt_id integer(10)
company_previousname
com_pn_id integer(10) U
com_pn1_condate varchar(20)
com_pn1_companyname varchar(160)
com_pn2_condate varchar(20)
com_pn2_companyname varchar(160)
com_pn3_condate varchar(20)
com_pn3_companyname varchar(160)
com_pn4_condate varchar(20)
com_pn4_companyname varchar(160)
com_pn5_condate varchar(20)
com_pn5_companyname varchar(160)
com_pn6_condate varchar(20)
com_pn6_companyname varchar(160)
com_pn7_condate varchar(20)
com_pn7_companyname varchar(160)
com_pn8_condate varchar(20)
com_pn8_companyname varchar(160)
com_pn9_condate varchar(20)
com_pn9_companyname varchar(160)
com_pn10_condate integer(20)
com_pn10_companyname varchar(160)
company_account_category
com_acc_category_id integer(10) U
com_acc_category varchar(100) N
company_account
com_acc_id integer(10) U
com_acc_refday integer(2)
com_acc_refmonth integer(2)
com_acc_nextduedate varchar(20)
com_acc_lastmadeupdate varchar(20)
com_acc_category_id integer(10)
company_returns
com_return_id integer(10) U
com_return_nextduedate varchar(50)
com_return_lastmadeupdate varchar(50)
company_mortgages
com_mort_id integer(10) U
com_num_mortchanges integer(10)
com_num_mortoutstanding integer(10)
com_num_mortpartsatisfied integer(10)
com_num_mortsatisfied integer(10)
company_partnership
com_partnership_id integer(10) U
com_num_genpartners integer(10)
com_num_limpartners integer(10)
company_uri
com_uri_id integer(10) U
com_uri varchar(47)
Powered By Visual Paradigm Community Edition

FIGURE 3.15: Company Normalized Database Design

The figure above shows Company's entity relationship diagram (ERD) to provide a graphical representation of normalized database design that display the relationships of entity stored in a database.

### 3.6.2.3 Phase 2 Normalized Postcode Entity Relationship Diagram

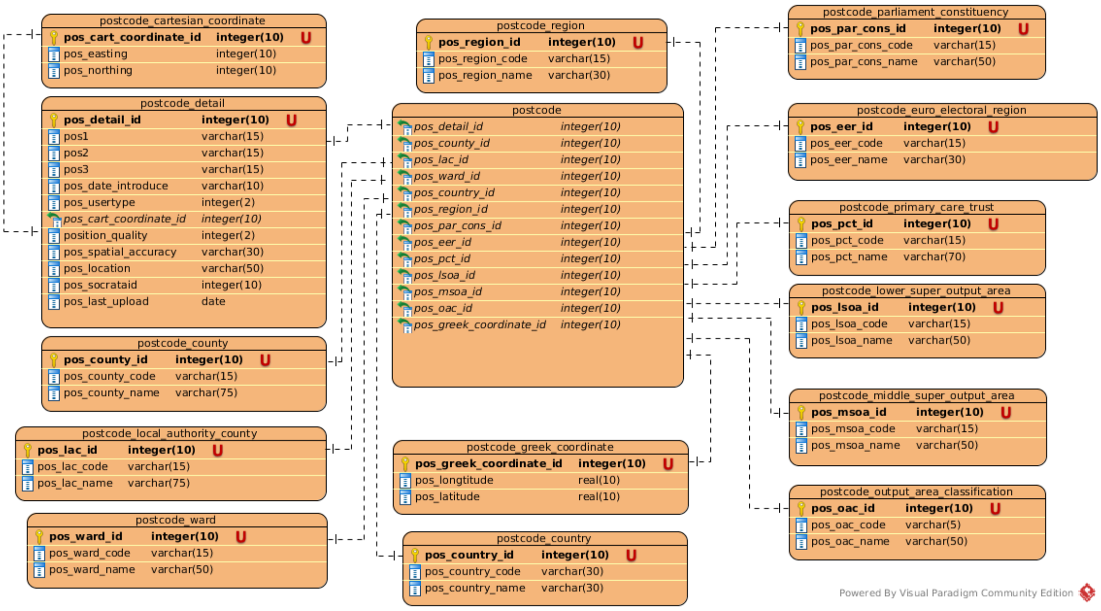

postcode_cartesian_coordinate
pos_cart_coordinate_id integer(10) U
pos_easting integer(10)
pos_northing integer(10)
postcode_detail
pos_detail_id integer(10) U
pos1 varchar(15)
pos2 varchar(15)
pos3 varchar(15)
pos_date_introduce varchar(10)
pos_usertype integer(2)
pos_cart_coordinate_id integer(10)
position_quality integer(2)
pos_spatial_accuracy varchar(30)
pos_location varchar(50)
pos_socrataid integer(10)
pos_last_upload date
postcode_county
pos_county_id integer(10) U
pos_county_code varchar(15)
pos_county_name varchar(75)
postcode_local_authority_county
pos_lac_id integer(10) U
pos_lac_code varchar(15)
pos_lac_name varchar(75)
postcode_ward
pos_ward_id integer(10) U
pos_ward_code varchar(15)
pos_ward_name varchar(50)
postcode_region
pos_region_id integer(10) U
pos_region_code varchar(15)
pos_region_name varchar(30)
postcode
pos_detail_id integer(10)
pos_county_id integer(10)
pos_lac_id integer(10)
pos_ward_id integer(10)
pos_country_id integer(10)
pos_region_id integer(10)
pos_par_cons_id integer(10)
pos_eer_id integer(10)
pos_pct_id integer(10)
pos_lsoa_id integer(10)
pos_msoa_id integer(10)
pos_oac_id integer(10)
pos_greek_coordinate_id integer(10)
postcode_greek_coordinate
pos_greek_coordinate_id integer(10) U
pos_longtitude real(10)
pos_latitude real(10)
postcode_country
pos_country_id integer(10) U
pos_country_code varchar(30)
pos_country_name varchar(30)
postcode_parliament_constituency
pos_par_cons_id integer(10) U
pos_par_cons_code varchar(15)
pos_par_cons_name varchar(50)
postcode_euro_electoral_region
pos_eer_id integer(10) U
pos_eer_code varchar(15)
pos_eer_name varchar(30)
postcode_primary_care_trust
pos_pct_id integer(10) U
pos_pct_code varchar(15)
pos_pct_name varchar(70)
postcode_lower_super_output_area
pos_lsoa_id integer(10) U
pos_lsoa_code varchar(15)
pos_lsoa_name varchar(50)
postcode_middle_super_output_area
pos_msoa_id integer(10) U
pos_msoa_code varchar(15)
pos_msoa_name varchar(50)
postcode_output_area_classification
pos_oac_id integer(10) U
pos_oac_code varchar(5)
pos_oac_name varchar(50)
Powered By Visual Paradigm Community Edition

FIGURE 3.16: Postcode Normalized Database Design

The figure above shows Postcode's entity relationship diagram (ERD) to provide a graphical representation of normalized database design that display the relationships of entity stored in a database.

### 3.6.2.4 Phase 2 Normalized Education Entity Relationship Diagram

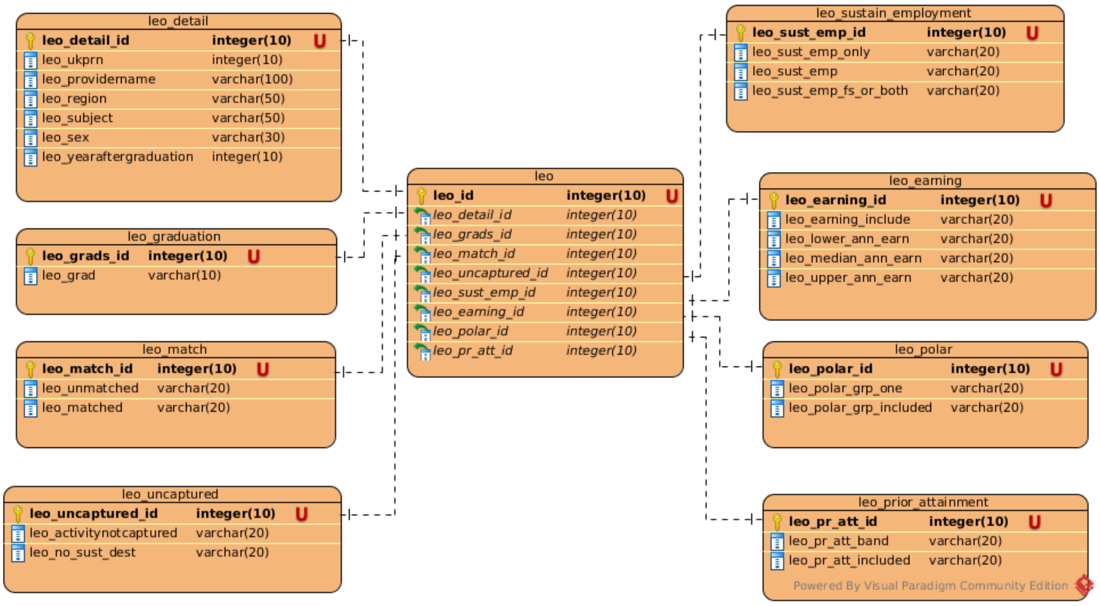

leo_detail
leo_detail_id integer(10) U
leo_ukprn integer(10)
leo_providername varchar(100)
leo_region varchar(50)
leo_subject varchar(50)
leo_sex varchar(30)
leo_yearaftergraduation integer(10)
leo_graduation
leo_grads_id integer(10) U
leo_grad varchar(10)
leo_match
leo_match_id integer(10) U
leo_unmatched varchar(20)
leo_matched varchar(20)
leo_uncaptured
leo_uncaptured_id integer(10) U
leo_activitynotcaptured varchar(20)
leo_no_sust_dest varchar(20)
leo
leo_id integer(10) U
leo_detail_id integer(10)
leo_grads_id integer(10)
leo_match_id integer(10)
leo_uncaptured_id integer(10)
leo_sust_emp_id integer(10)
leo_earning_id integer(10)
leo_polar_id integer(10)
leo_pr_att_id integer(10)
leo_sustain_employment
leo_sust_emp_id integer(10) U
leo_sust_emp_only varchar(20)
leo_sust_emp varchar(20)
leo_sust_emp_fs_or_both varchar(20)
leo_earning
leo_earning_id integer(10) U
leo_earning_include varchar(20)
leo_lower_ann_earn varchar(20)
leo_median_ann_earn varchar(20)
leo_upper_ann_earn varchar(20)
leo_polar
leo_polar_id integer(10) U
leo_polar_grp_one varchar(20)
leo_polar_grp_included varchar(20)
leo_prior_attainment
leo_pr_att_id integer(10) U
leo_pr_att_band varchar(20)
leo_pr_att_included varchar(20)
Powered By Visual Paradigm Community Edition

FIGURE 3.17: Education Normalized Database Design

The figure above shows Education's entity relationship diagram (ERD) to provide a graphical representation of normalized database design that display the relationships of entity stored in a database.

## 3.6.3 Data Cleaning Parser

### 3.6.3.1 Phase 2 Company Data Cleaning Parser Deployment Diagram

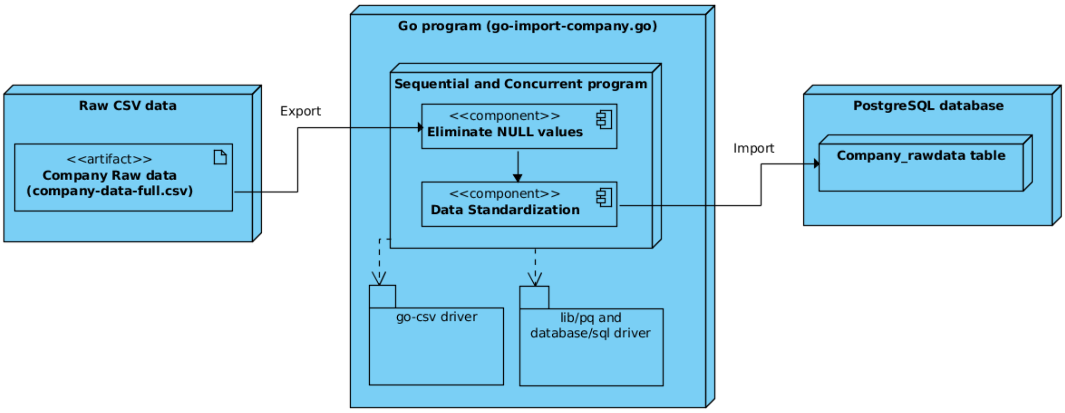


Figure 3.18: Company Data Cleaning Parser Deployment Diagram

The figure 3.18 shows the deployment diagram of company data cleaning parser.

The cleaning parser is written with Go program language that consume encoded company raw data (performed in Section 3.3) as input and make execution decisions to eliminate NULL values and perform data standardization to repair missing and incorrect data. Afterwards, the cleaned data will be stored into PostgreSQL database await to be processed. The program work similarly as ORM (mentioned in Section 3.5) by utilizing go-csv driver to retrieve data from CSV files and lib/pq or database/sql driver to establish connection and perform transaction with the PostgreSQL database.

## 3.7 Database Tuning

### 3.7.1 Phase 2 Database Tuning Flowchart

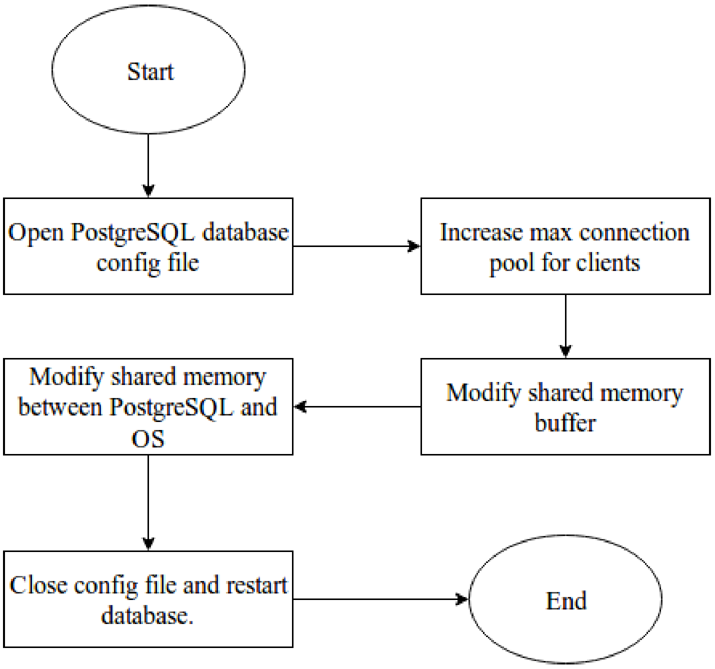

Start
Open PostgreSQL database config file
Increase max connection pool for clients
Modify shared memory buffer
Modify shared memory between PostgreSQL and OS
Close config file and restart database.
End

FIGURE 3.19: Database Tuning Flowchart

Database tuning is a process of configure PostgreSQL database's environment to optimize performance by increase throughput and decrease response time. The approach required to open PostgreSQL database configuration file with root access in Linux Operating System environment. The configuration made and reason to perform are describe as follow:

1. **Max Connection.** The number max connection of PostgreSQL database is modified to allow more *Goroutines* from Go program to establish database connection concurrently and perform parallelize transaction.

This modification helps increase performance on Data Cleaning and Data Migration in this project. If the connection pool is not modified, the database system will display FATAL error and terminate the process immediately.

2. **Shared Buffer.** The parameter of shared memory buffer shall be modified as 25 percents of memory in our systems. Increase the amount of memory PostgreSQL database uses for shared memory buffers allow the database to handle extra workloads.
3. **Shared Memory.** The maximum size of shared memory segment shall be modified to allow *Goroutines* or *threads* access to PostgreSQL database simultaneously for better data passing and avoid redundant copies. This configuration parameter determine dedicated memory for PostgreSQL to caching data and increase the space for threads to communicate with the database. The parameter shall be modify with Bytes(B).

Ultimately, restart of PostgreSQL database is required to update the changes and modifications.

## 3.8 Data Migration

Data migration is the process of transferring data within storage system for database migration. [74] Data migration is extremely challenging as we need to take care of performance issues, data integrity, data consistency and prevent data corruption. The data should be protect carefully and prevent missing during the migration process.

### 3.8.1 Phase 2 Data Migration Deployment Diagram

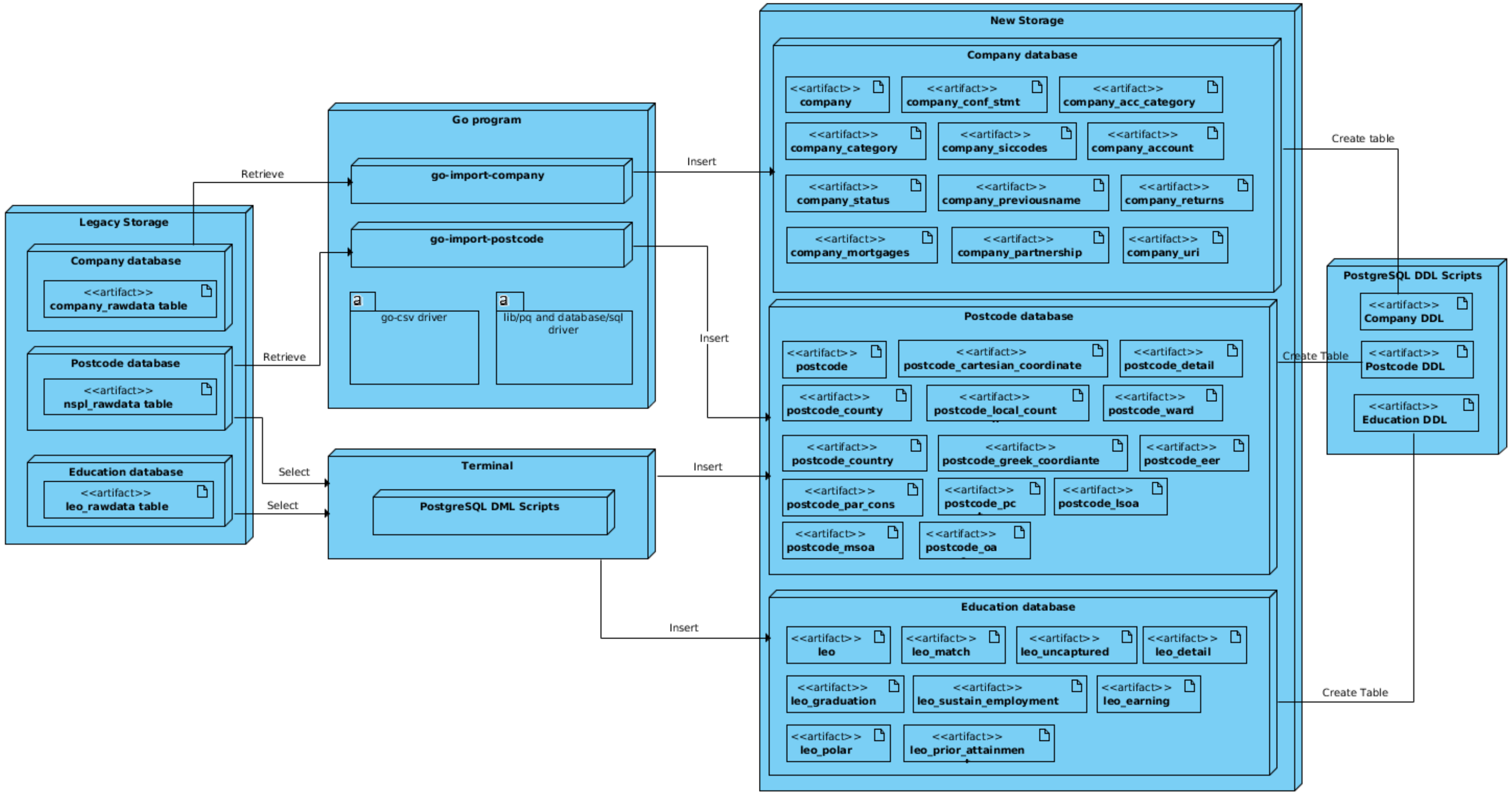

Legacy Storage
Company database
<<artifact>> company_rawdata table
Postcode database
<<artifact>> nspl_rawdata table
Education database
<<artifact>> leo_rawdata table
Retrieve
Retrieve
Select
Select
Go program
go-import-company
go-import-postcode
go-csv driver
lib/pq and database/sql driver
Terminal
PostgreSQL DML Scripts
Insert
Insert
Insert
Insert
New Storage
Company database
<<artifact>> company
<<artifact>> company_conf_stmt
<<artifact>> company_acc_category
<<artifact>> company_category
<<artifact>> company_siccodes
<<artifact>> company_account
<<artifact>> company_status
<<artifact>> company_previousname
<<artifact>> company_returns
<<artifact>> company_mortgages
<<artifact>> company_partnership
<<artifact>> company_uri
Postcode database
<<artifact>> postcode
<<artifact>> postcode_cartesian_coordinate
<<artifact>> postcode_detail
<<artifact>> postcode_county
<<artifact>> postcode_local_count
<<artifact>> postcode_ward
<<artifact>> postcode_country
<<artifact>> postcode_greek_coordiante
<<artifact>> postcode_eer
<<artifact>> postcode_par_cons
<<artifact>> postcode_pc
<<artifact>> postcode_lsoa
<<artifact>> postcode_msoa
<<artifact>> postcode_oa
Education database
<<artifact>> leo
<<artifact>> leo_match
<<artifact>> leo_uncaptured
<<artifact>> leo_detail
<<artifact>> leo_graduation
<<artifact>> leo_sustain_employment
<<artifact>> leo_earning
<<artifact>> leo_polar
<<artifact>> leo_prior_attainmen
Create table
Create Table
Create Table
PostgreSQL DDL Scripts
<<artifact>> Company DDL
<<artifact>> Postcode DDL
<<artifact>> Education DDL

Figure 3.20: Data Migration Deployment Diagram

Figure 3.20 shows the deployment diagram of data migration process.

The normalized table in all database are created with PL/pgSQL DDL scripts. Once the creation of table is successful, the data migration of education database is performed with PL/pgSQL DML script running in terminal environment. The mentioned database is migrated with script because it only contains 30000+ rows and its lightweight to be process with queries.

Afterwards, the postcode and company database are migrated from legacy storage to new storage with the execution of scripts and Go program as shown in Figure 3.23. Both company and postcode data are migrated with Go program because it contains more than 4 millions rows in total and its difficult to handle with queries. The unique data is extracted from legacy storage and stored into the normalized table in new storage.

The postcode migration program is developed with **Channel Synchronization** concepts to perform data migration execution across goroutines to form an concurrent execution. The synchronization primitives of Go programming language is used to perform communication between threads within channel in mutual exclusion locks.

Other than that, the company migration program is developed with **Semaphore** concepts to apply control access of 400,000 *Goroutines* on common resource provided by PostgreSQL database and operating system environment. The concurrency of data migration execution in this program are controlled and limited to prevent race condition. These Goroutines are required to communicate with each other to utilized 299 open connection with PostgreSQL database on migrating 3.5 millions of data with specific resource provided.

The migration program is written with Go programming language with the inclusion of database/sql driver to establish connection and lib/pq driver to perform transaction with PostgreSQL database. All the migration process does not modify the source data in legacy storage to serve as backup for in case of emergence. In addition, the changes of migration can be easily tracked for verification purposes. Ultimately, the migration duration is recorded and measured.

# Chapter 4

# Implementation Methodology

## 4.1 Software Engineering Methodology

Software engineering life cycle (SDLC) is a well structured and iterative sequence of stages in to deliver quality research which meet or exceed project scope. It involves five major activities in this project which are: :

- **Communication.** Student initiate the request to supervisor for apply specific project title offered in this semester. Requirement gathering is conducted in order to discuss the expectation of project and understand the critical factors to achieve project scope or objective. The process required mass amount of communication and collaboration between student and supervisor to ensure requirement are fully understood.
- **Planning.** Project management plan is define and prepare with Gantt Chart to manage project execution by considering risk assessment, resources estimation, time and task management. The tools and

techniques to be used requires to be understand in detail and comprehensive manner to achieve solid understand on whole project execution.

- **Construction.** The creation of project documentation and program through a combination of verification, coding, writing, debugging and testing. The complexity of project are required to be minimize and reduce with the use of standards. The program is construct based on requirement designed in software design phase to ensure the outcomes meet project objectives.
- **Testing.** The project outcomes and deliveries are required to update for supervisor and hand-in to the institution. Documentation and outcomes are required to conform with requirement specification and meet project requirements to ensure the project is doing right.

### 4.1.1 Prototyping Model Method

The software prototyping method is build prototypes with limited functionality as preliminary design to represent an approximation of concept. The prototype is implemented as proof of concepts for project objectives and reviewed by supervisor to enhance the prototype.

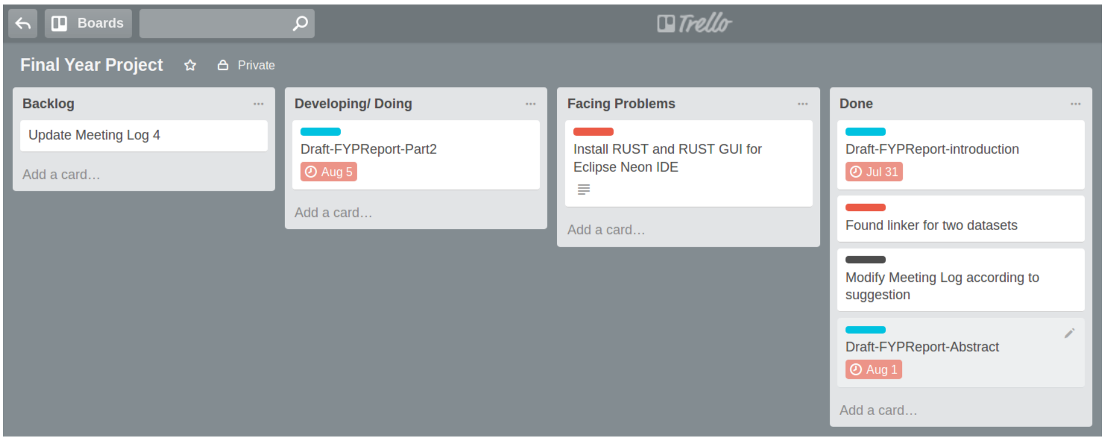
Prototyping helps strengthen understanding the requirement of project through communication and negotiation. The characteristic and basic features of program are demonstrate to collect feedback for enhancement and improvement. This method helps improve familiarity and early determination of requirement specification before development process to reduce chances of fail in the project. Time and project resources can be estimated throughout the process to conduct task and time management in order to deliver the final product.

## 4.2 Agile Software Methodology

The process decision framework used by this project is Agile Methodology. The mentioned methodology simplified process decisions around incremental and iterative solution delivery, rapid deliver features and update in order to satisfy requirement for weekly project updates. Agile methodology provide flexibility for the project progress respond to change and modification from FYP weekly meeting.

Agile software development describes set of principles for product and technology development under which requirements and solutions evolve through the collaborative effort of self-organizing management. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change according to feedback provide by supervisor. The SDLC or paradigm involved in agile methodology in this project is Kanban.

### 4.2.1 Kanban



Final Year Project

Backlog

- Update Meeting Log 4

Developing/ Doing

- Draft-FYPReport-Part2 (Aug 5)

Facing Problems

- Install RUST and RUST GUI for Eclipse Neon IDE

Done

- Draft-FYPReport-introduction (Jul 31)
- Found linker for two datasets
- Modify Meeting Log according to suggestion
- Draft-FYPReport-Abstract (Aug 1)

FIGURE 4.1: Kanban board

Kanban provide visual information of workflow by using sticky notes on a whiteboard to create a “picture” of our work. The board allow visualize the project development process or work flows within process and it helps ease the communicate status but also give and receive context for the work. Trello is used in this project as online Kanban board to manage the task in this methodology.

There are an amount of work-in-progress (WIP) on each simple phased process to prevents overproduction and reveals bottlenecks dynamically to aware several roles whether are in bottlenecks. As an example, if the software pipelines are Backlog, Developing, Facing Problems and Done. There are WIP limits on each phased to increase the inspection and create awareness in order to facilitate adaptation based on the work loads.

When a new requirement or changes requested, the task is insert into the backlog. The priority of the task are influenced by time constraint and importance. Afterwards, the task will be move into “developing” to began construction of documentation or codes. Once the task is encountered difficulty and problem, it will move to “”facing problem”. Alternatively, the task will move to “done” once the task is completed and ready to submit or show to supervisor during meeting.

The Kanban events required to developed immediately and unknown incident may interrupt the progress depends on project feedback and requirement needs. A new high priority fix or changes may requested and it will break off the current project flow. Kanban allow the project respond to change efficiently and provide continuous update on progress to supervisor in order to submit quality works at end of project phase.

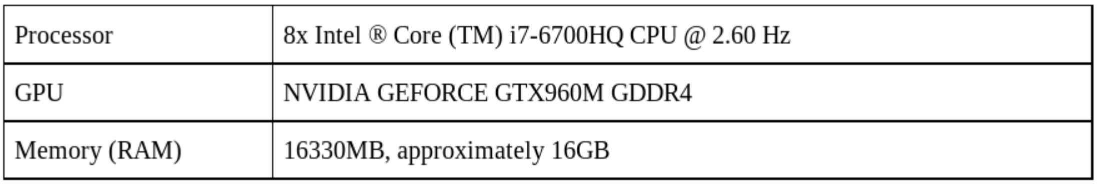
### 4.2.2 Methodology for this Project

In this project, we will be developing Go and Rust program for conduct concurrent and distributed programming. To achieve the required tasks, rapid communication and modification is conducted to improve quality of program and satisfy project objectives. Prototyping method and Kanban will be use in this project.

## 4.3 Project Infrastructure

### 4.3.1 List of Hardware Resources

1. **64-bit Personal Computer.** This machine is used for research and development activities of this project. The details are tabulated and shown below:



| | |
|---|---|
| Processor | 8x Intel ® Core (TM) i7-6700HQ CPU @ 2.60 Hz |
| GPU | NVIDIA GEFORCE GTX960M GDDR4 |
| Memory (RAM) | 16330MB, approximately 16GB |

Figure 4.2: Personal Computer Hardware table

### 4.3.2 List of Software Resources

1. **Linux Ubuntu 16.04.3 LTS 64-bit.** The community driven and open source operating system is used to conduct concurrent and distributed computing with Go and Rust compiler installed. The details are discussed in Chapter 3.2.1.
2. **Golang language compiler 1.8.3.** The linux amd64 gccgo compiler build Go source code into binary executable with “go build” and run the go program with “go run”. It is use to compile and run Go files this project.
3. **Rust language compiler 1.20.0.** The linux amd64 rustc compiler compile Go source code into executable with “rustc”. It is use to compile Rust files in this project.

4. **PostgreSQL database 9.5.8.** The open source database management system is use for data handling and data storage for this project. The details are discussed in Chapter 3.2.3.
5. **Eclipse for Parallel Application Developers Oxygen Release (4.7.0) IDE.** The open source IDE provide perspective feature and integrated debugger to ease the coding and development activities for this project. The details are discussed in Chapter 3.2.2.
6. **Goclipse Plugin for Eclipse IDE.** The plugin provide debugging functionality, content assist, auto code indentation, open definition and integrated compiler for Go language on Eclipse IDE.
7. **RustDT Plugin for Eclipse IDE.** The plugin provide syntax highlighting, error reporting, outline support, auto code indentation, debugging functionality and integrated compiler for Rust language on Eclipse IDE.
8. **TeXstudio 2.10.8.** The software provide writing environment for create LaTeX document with numerous feature such as syntax-highlighting, reference checking with bibtex and various assistant. It is use for creating documentation for this project.

9. **Visual Paradigm 14.1 free edition for non-commercial use.** The software is a free Unified Modelling Language Computer-Aided Software Engineering tool support 13 UML diagram types for software design and modelling. It is use to draw diagrams for this project.

### 4.3.3 Other Project Resources

1. **Synaptic Package Manager.** The software system is a graphical package management program of APT libraries and provide same features as apt-get command. It provide great assist and help on managing software package dependencies. It is installed with *"sudo apt-get install synaptic"* in terminal.
2. **Terminator.** Terminator provide multiple tabs, safe quit, UTF-8 encoding, automatic logging to ease the development activities for developer. The system is required to update source list with *"sudo apt-get update"* and run *"sudo apt-get install terminator"* to install the repository.

### 4.3.4 Infrastructure Setup and Installation

The required hardware and software resources are listed and discussed in Chapter 4.2.1 and Chapter 4.2.2.

#### 4.3.4.1 Go language compiler installation

1. Ensure Golang go1.8.3.linux-amd64.tar.gz is downloaded using wget in terminal.
2. Ensure downloaded file is extract, move and rename Golang directory.
3. Ensure Golang's compiler export to system path.
4. Ensure Goroot and Gopath is set.
5. Ensure path to user profile .bashrc file is append.
6. Ensure Go executable and Go version installation is success.
7. Ensure Go libraries such as gocode, golint, guru, goimports, gorename and godef into Gopath directory are installed.
8. Ensure Godef Gometalinter is downloaded and executed.

The full installation steps for Go language compiler is found in Appendix A.1.

#### 4.3.4.2 RUST language compiler installation

1. Install Rust toolchain with command line.
2. Export rust executable to system path.
3. Install Racer, Rustfmt, Rainicorn.
4. Ensure all the required Rust executables are installed.

The full installation steps for RUST language compiler is found in Appendix A.2.

#### 4.3.4.3 Eclipse IDE installation

1. Ensure Java is installed before start download Eclipse.
2. Run *"sudo apt-get update"* and *"sudo apt-get upgrade"* before start download.
3. Make eclipse-workspace folder as default storage for better management.

The installation details for Eclipse IDE is found in Appendix A.3.

#### 4.3.4.4 GoClipse plugin for Eclipse IDE installation

1. Install Goclipse plugin with Eclipse marketplace.
2. Ensure Goclipse preferences and setting are correct.

The full installation steps for Goclipse plugin on Eclipse IDE is found in Appendix A.4.

#### 4.3.4.5 RustDT plugin for Eclipse IDE installation

1. Install RustDT plugin with Eclipse marketplace.
2. Ensure RustDT preferences and setting are correct.

The full installation steps for RustDT plugin on Eclipse IDE is found in Appendix A.5.

#### 4.3.4.6 PostgreSQL database installation and setup

1. Install postgreSQL in command line.
2. Ensure database for FYP1 is created.
3. Create new user for database.
4. Ensure database connection is established with user access.

The full installation steps for PostgreSQL database is found in Appendix A.6.

# Chapter 5

# Implementation Plan

## 5.1 Project Task Identification

### 5.1.1 Identification of Critical Success Factors

Critical success factors are a key requirement which is necessary and essential to be identified to achieve the project objectives in this project. The requirement for our design objectives are listed below:

1. **Determine a suitable operating system.** The operating system should be reliable, secure and appropriate for data processing, concurrent and distributed computing activities. If the selected operating system does not meet requirements, a new operating system has to be considered.
2. **Acquire free public data set for big data processing.** Large data set is required for data processing with concurrent and distributed computing to make use of concurrent programming language's package

and architecture. If the data set obtains not clean and useful, data cleansing and data deduplication have to be conducted.

3. **Selection of database management system (DBMS).** The database-management system for this project should support for operating system, concurrent programming language and project activities. If the selected DBMS does not compatible and suitable, a new DBMS capability has to be considered.

4. **Installation and setup DBMS for big data handling.** The selected database-management system should be installed and running on the operating system for data storing and data handling. The database system allows developer to conduct development activities for manage concurrency control for update and retrieval in this project.

5. **Selection of Go and RUST concurrent programming language for comparison.** There are many types of concurrent programming language for system development. The selected language for this project is RUST and Go. This programming language architecture, packages and capabilities should be considered to conduct performance comparison.

6. **Coding of "Import CSV into database" with Go program.** The program is required to write with Go language to read CSV and upload into PostgreSQL database. This task is conduct for data definition and data preparation before data processing is performed.

7. **Coding of "Import CSV into database" with RUST program.** The program is required to write with Go language in order to read CSV and upload into PostgreSQL database. This task is conduct for data definition and data preparation before data processing is performed.

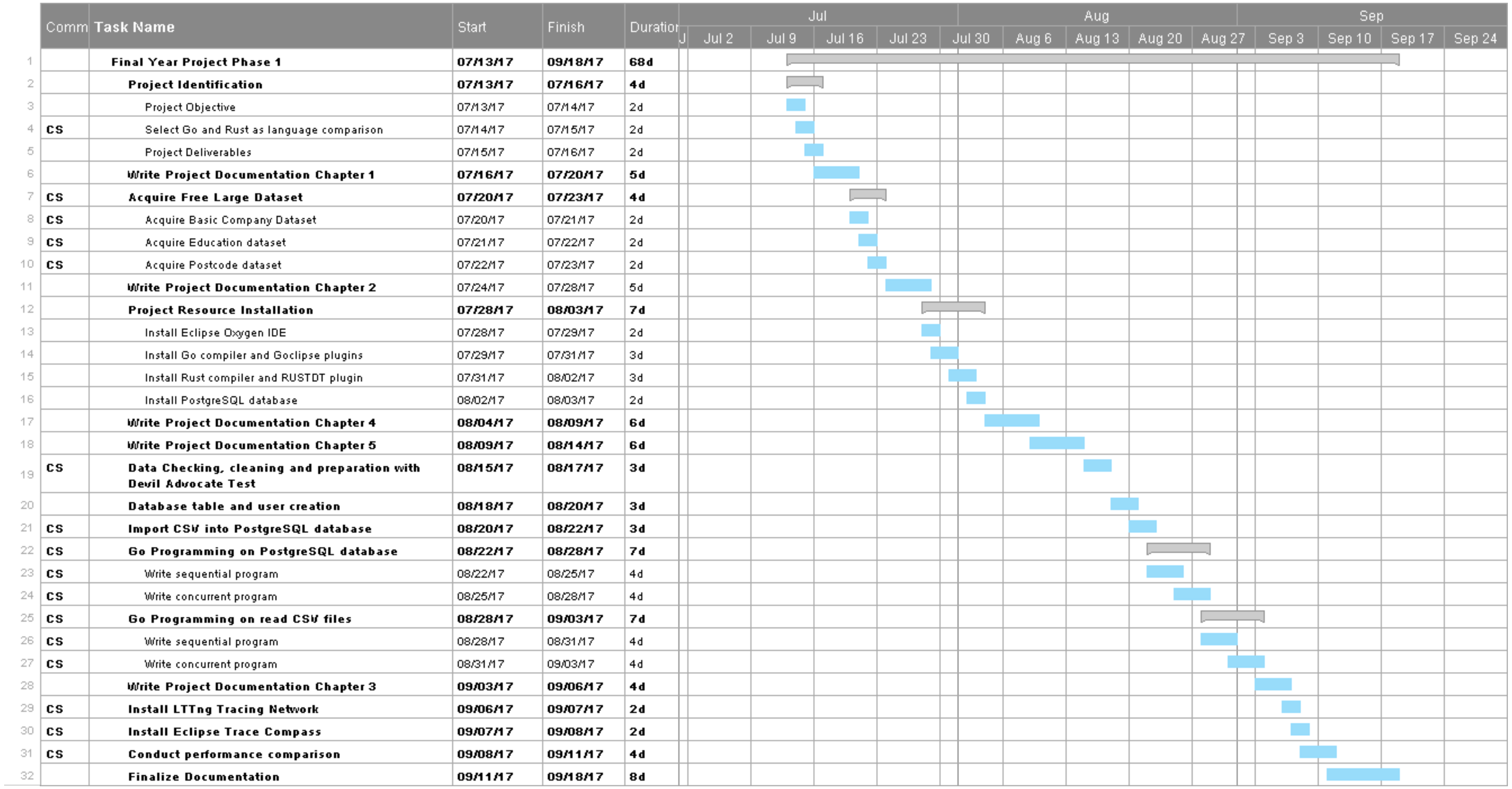
8. **Conduct minor comparison on sequential and concurrent programming with Go and RUST language on PostgreSQL database transaction.** The sequential and concurrent program is required to write with Go and RUST language in order to conduct a comparison of execution time for database retrieval on PostgreSQL.

9. **Conduct minor comparison on sequential and concurrent programming with Go and RUST language on reading CSV files.** The sequential and concurrent program is required to write with Go and RUST language to conduct a comparison of execution time on reading CSV files.

### 5.1.2 Project Tasks for FYP Phase 1

1. Installation of Ubuntu 16.04 LTS 64-bit operating system.
2. Acquire free public data set for big data processing.
3. Installation of Eclipse Parallel Application IDE Parallel Oxygen version.
4. Selection of Go and RUST concurrent programming language for comparison.
5. Installation of Go language compiler and Goclipse plugin for Eclipse IDE.
6. Installation of RUST language compiler and RustDT plugin for Eclipse IDE.
7. Selection of PostgreSQL object-oriented relational database management system (OORDBMS).
8. Installation and setup PostgreSQL database system intro PC for data handling.
9. Golang programming for import CSV files into PostgreSQL database.
10. Sequential and concurrent programming with Golang on PostgreSQL database retrieval.
11. Sequential and concurrent programming with Golang on reading CSV files.
12. Big data checking, cleaning and preparation with Data Validation.

### 5.1.3 Gantt Chart for Phase 1



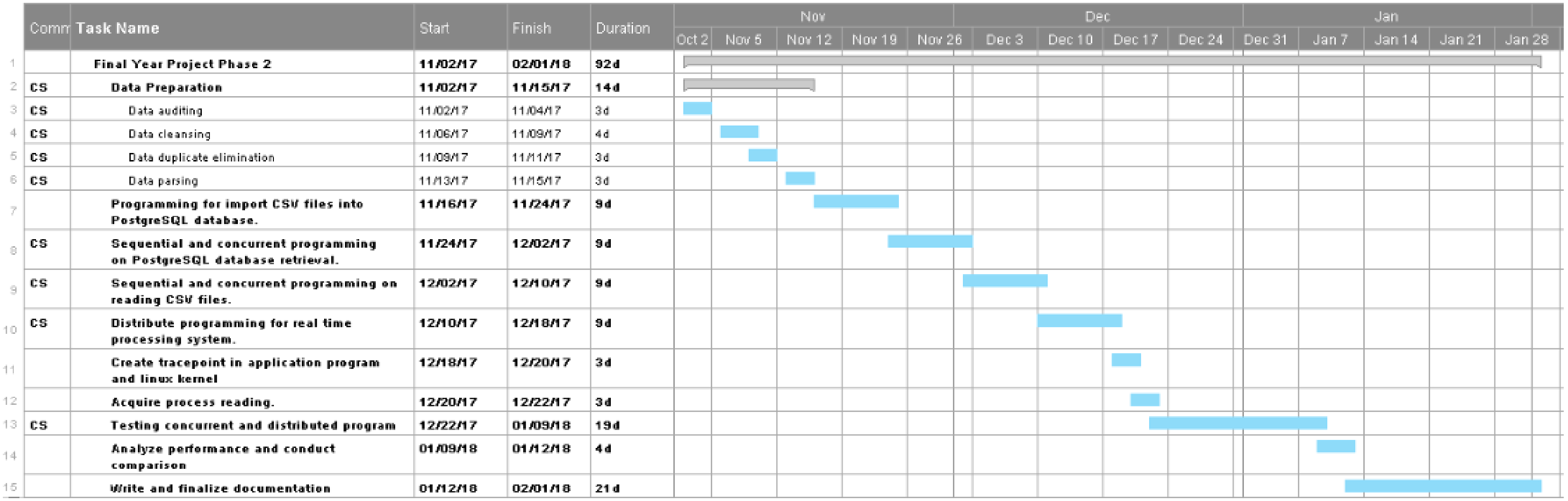
| | Comm | Task Name | Start | Finish | Duration |
|---|---|---|---|---|---|
| 1 | | **Final Year Project Phase 1** | **07/13/17** | **09/18/17** | **68d** |
| 2 | | **Project Identification** | **07/13/17** | **07/16/17** | **4d** |
| 3 | | Project Objective | 07/13/17 | 07/14/17 | 2d |
| 4 | **CS** | Select Go and Rust as language comparison | 07/14/17 | 07/15/17 | 2d |
| 5 | | Project Deliverables | 07/15/17 | 07/16/17 | 2d |
| 6 | | **Write Project Documentation Chapter 1** | **07/16/17** | **07/20/17** | **5d** |
| 7 | **CS** | **Acquire Free Large Dataset** | **07/20/17** | **07/23/17** | **4d** |
| 8 | **CS** | Acquire Basic Company Dataset | 07/20/17 | 07/21/17 | 2d |
| 9 | **CS** | Acquire Education dataset | 07/21/17 | 07/22/17 | 2d |
| 10 | **CS** | Acquire Postcode dataset | 07/22/17 | 07/23/17 | 2d |
| 11 | | **Write Project Documentation Chapter 2** | 07/24/17 | 07/28/17 | 5d |
| 12 | | **Project Resource Installation** | **07/28/17** | **08/03/17** | **7d** |
| 13 | | Install Eclipse Oxygen IDE | 07/28/17 | 07/29/17 | 2d |
| 14 | | Install Go compiler and Goclipse plugins | 07/29/17 | 07/31/17 | 3d |
| 15 | | Install Rust compiler and RUSTDT plugin | 07/31/17 | 08/02/17 | 3d |
| 16 | | Install PostgreSQL database | 08/02/17 | 08/03/17 | 2d |
| 17 | | **Write Project Documentation Chapter 4** | **08/04/17** | **08/09/17** | **6d** |
| 18 | | **Write Project Documentation Chapter 5** | **08/09/17** | **08/14/17** | **6d** |
| 19 | **CS** | **Data Checking, cleaning and preparation with Devil Advocate Test** | **08/15/17** | **08/17/17** | **3d** |
| 20 | | **Database table and user creation** | **08/18/17** | **08/20/17** | **3d** |
| 21 | **CS** | **Import CSV into PostgreSQL database** | **08/20/17** | **08/22/17** | **3d** |
| 22 | **CS** | **Go Programming on PostgreSQL database** | **08/22/17** | **08/28/17** | **7d** |
| 23 | **CS** | Write sequential program | 08/22/17 | 08/25/17 | 4d |
| 24 | **CS** | Write concurrent program | 08/25/17 | 08/28/17 | 4d |
| 25 | **CS** | **Go Programming on read CSV files** | **08/28/17** | **09/03/17** | **7d** |
| 26 | **CS** | Write sequential program | 08/28/17 | 08/31/17 | 4d |
| 27 | **CS** | Write concurrent program | 08/31/17 | 09/03/17 | 4d |
| 28 | | **Write Project Documentation Chapter 3** | **09/03/17** | **09/06/17** | **4d** |
| 29 | **CS** | **Install LTTng Tracing Network** | **09/06/17** | **09/07/17** | **2d** |
| 30 | **CS** | **Install Eclipse Trace Compass** | **09/07/17** | **09/08/17** | **2d** |
| 31 | **CS** | **Conduct performance comparison** | **09/08/17** | **09/11/17** | **4d** |
| 32 | | **Finalize Documentation** | **09/11/17** | **09/18/17** | **8d** |

FIGURE 5.1: Gantt Chart for Phase 1

### 5.1.4 Project Tasks for FYP Phase 2

1. Data encoding.
2. Data transformation.
3. Data parsing.
4. Data cleansing.
5. Data normalization.
6. Database tuning.
7. Query tuning.
8. Data migration.
9. Sequential and concurrent programming with Go and RUST on PostgreSQL database retrieval.
10. Sequential and concurrent programming with Go and RUST on reading CSV files.

### 5.1.5 Gantt Chart for Phase 2



| | Comm | Task Name | Start | Finish | Duration |
|---|---|---|---|---|---|
| 1 | | Final Year Project Phase 2 | 11/02/17 | 02/01/18 | 92d |
| 2 | CS | Data Preparation | 11/02/17 | 11/15/17 | 14d |
| 3 | CS | Data auditing | 11/02/17 | 11/04/17 | 3d |
| 4 | CS | Data cleansing | 11/06/17 | 11/09/17 | 4d |
| 5 | CS | Data duplicate elimination | 11/09/17 | 11/11/17 | 3d |
| 6 | CS | Data parsing | 11/13/17 | 11/15/17 | 3d |
| 7 | | Programming for import CSV files into PostgreSQL database. | 11/16/17 | 11/24/17 | 9d |
| 8 | CS | Sequential and concurrent programming on PostgreSQL database retrieval. | 11/24/17 | 12/02/17 | 9d |
| 9 | CS | Sequential and concurrent programming on reading CSV files. | 12/02/17 | 12/10/17 | 9d |
| 10 | CS | Distribute programming for real time processing system. | 12/10/17 | 12/18/17 | 9d |
| 11 | | Create tracepoint in application program and linux kernel | 12/18/17 | 12/20/17 | 3d |
| 12 | | Acquire process reading. | 12/20/17 | 12/22/17 | 3d |
| 13 | CS | Testing concurrent and distributed program | 12/22/17 | 01/09/18 | 19d |
| 14 | | Analyze performance and conduct comparison | 01/09/18 | 01/12/18 | 4d |
| 15 | | Write and finalize documentation | 01/12/18 | 02/01/18 | 21d |

FIGURE 5.2: Gantt Chart for Phase 2

### 5.1.6 Milestone Deliverables

The milestone deliverables are:

1. Go program for data parsing, object relational mapping and data migration.
2. RUST program for data parsing and object relational mapping.
3. PL/pgSQL's DDL and DML scripts for database creation, manipulation and migration control.
4. A report based of this project.

## 5.2 Planned Execution Activities

### 5.2.1 Phase 1

1. **Data Validation.** The Data Validation is conducted to ensure obtained raw CSV data set is clean and useful. The expected result of this test is the number of commas in the record should not exceed the number of columns in a database. In addition, the data content itself should be unique and suitable for storing in the database. More information is provided in Appendix B.1.
2. **Golang programming for import CSV files into PostgreSQL database.** The Golang programming for import CSV raw data into PostgreSQL is to ensure Go language is capable of processing raw CSV data and PostgreSQL database. The expected result for this program

should read 100 rows of data from raw CSV file and insert into PostgreSQL database. More information is provided in Appendix C.

3. **Sequential and concurrent programming with Golang on PostgreSQL database retrieval.** The Go program should retrieve 300 rows of data from three tables (each table 100 rows) in PostgreSQL database sequentially and concurrently. The expected result for this program is concurrent processing should have better performance than sequential. More information is provided in Appendix D.
4. **Sequential and concurrent programming with Golang on reading CSV files.** The Go program should retrieve 100 rows of data from raw CSV file sequentially and concurrently. The expected result for this program is concurrent processing should have better performance than sequential. More information is provided in Appendix E.

### 5.2.2 Phase 2

1. **Data encoding.** This activity is a deliverable of Phase 2 in this project. It is conducted to ensure that the dirty and corrupted datasets are converted into consistent format so that it will be safe to used for Object Relational Mapping, Data Transformation and Data Parsing. More information is provided in Appendix J.1 to J.2.
2. **Development of PL/pgSQL scripts for data transformation.** This activity is a deliverable of Phase 2 in this project. It is conducted to extract data in CSV format from raw datasets and import into PostgreSQL database. More information is provided in Appendix K.1 to K.2.

3. **Development of Go and Rust Object Relational Mapping (ORM) program for data retrieval.** This activity is a deliverable of Phase 2 in this project. The data from CSV file and PostgreSQL database are retrieved and map into object model. Go and Rust program should retrieve 4 millions row of data from raw CSV file and PostgreSQL database in sequential and concurrent manner. The execution duration of each program are tabulated and recorded for comparison purposes. More information is provided in Appendix L.1 to L.4.
4. **Development of PL/pgSQL DDL scripts for normalized entity creation.** This activity is a deliverable of Phase 2 in this project. It can eliminate redundancy and data anomalies to improve data integrity. Database design is performed to define table and establish relationship between entity to create a relational database schema. Moreover, normalized table will be created correctly with PL/pgSQL's DDL scripts based on the Entity Relationship Diagram shown in Section 3. More information is provided in Appendix M.
5. **Development of Go data parser program.** This activity is a deliverable of Phase 2 in this project. The missing fields will be eliminated and data standardization is conducted to promote conformity and usability of data. More information is provided in Appendix N.1.
6. **Database tuning.** This activity is a deliverable of Phase 2 in this project. It is performed to configure PostgreSQL's database environment and setting to increase performance on data processing. More information is provided in Appendix O.
7. **Development of PL/pgSQL's DML scripts and Go concurrent program for database migration.** This activity is a deliverable of

Phase 2 in this project. The data that are transformed and cleaned will be import into normalized table. These data are migrated from legacy storage into new storage within PostgreSQL database. More information is provided in Appendix P.1.

# Chapter 6

# Results and Findings

## 6.1 Phase 1

1. **Data validation.** This activity has been successfully achieved. It has been found the method can detect unmatched numbers of commas, unsuitable data types during data importation from CSV to PostgreSQL database and identify the uniqueness of rows and columns in data. Results and detailed information is provided in Appendix B.2 to B.4.
2. **Golang programming for import CSV files into PostgreSQL database.** This activity has been successfully achieved. The program is capable to read 100 rows of data from three datasets and import into PostgreSQL database. Results and detailed information is provided in Appendix H.

3. **Sequential and concurrent programming with Golang on PostgreSQL database retrieval.** This activity has been successfully achieved. The program is capable to prove concurrent processing is faster than sequential in data retrieval with PostgreSQL database. Results and detailed information is provided in Appendix G.
4. **Sequential and concurrent programming with Golang on reading CSV files.** This activity has been successfully achieved. The program is capable to prove concurrent processing is faster than sequential in reading CSV data. Results and detailed information is provided in Appendix F.

## 6.2 Phase 2

1. **Data encoding.** This activity has been successfully achieved. The dirty and corrupted CSV raw datasets can be converted into consistent format with stream editor. Result and detailed information is provided in Appendix J.3.
2. **Development of PL/pgSQL scripts for data transformation.** This activity has been successfully achieved. The developed scripts is capable to extract data from CSV format from raw datasets and import into PostgreSQL database. Result and detailed information is provided in Appendix K.3.
3. **Development of Go and Rust Object Relational Mapping (ORM) program for data retrieval.** This activity has been successfully achieved. The Go and Rust program developed is capable to retrieved data from CSV file and PostgreSQL database and map into object model

in sequential and concurrent manner. The activity proves concurrent processing is faster than sequential in data retrieval with PostgreSQL database and reading CSV data. Moreover, it proves Go programming languages possess faster processing time compared to Rust programming languages. Result and detailed information is provided in Appendix Q.

4. **Development of PL/pgSQL DDL scripts for normalized entity creation.** This activity has been successfully achieved. The database design is capable to define table and establish relationship between entity. In addition, the PL/pgSQL's DDL scripts developed is able to create database entity based on the database design correctly. Result and detailed information is provided in Appendix M.4.

5. **Development of Go data parser program.** This activity has been successfully achieved. The developed Go program is capable to eliminate NULL values and standardize the records of specific columns to promote conformity and usability of data. Result and detailed information is provided in Appendix N.2.

6. **Database Tuning.** This activity has been successfully achieved. The number of database maximum connections, amount of shared buffer utilized and maximum of shared memory segments are configured to increase performance and transaction efficiency of concurrent program. Result and detailed information is provided in Appendix O.

7. **Development of PL/pgSQL's DML scripts and Go concurrent program for database migration.** This activity has been successfully achieved. The PL/pgSQL's DML, DCL scripts is capable to retrieve unique data from legacy storage and insert into normalize table. In addition, the scripts and Go program are capable to migrate more than 4

millions row of data into normalized table without causing missing of records. Other than that, the size of database is reduced and all records are correct after the migration. Result and detailed information is provided in Appendix P.3 and P.4.

# Chapter 7

# Discussion

## 7.1 Problems Encountered & Overcoming Them

### 7.1.1 Acquisition of free large datasets for data processing

The problem encountered during data gathering of this project is difficulty on finding suitable free big data from websites. It is a challenge to find problem and raise question by going into data details. It took huge amount of time to understand the focus of project and gather desired data for problem solving.

With the help of supervisor, I had successfully obtained suitable datasets for this project. He provides guidance and helping hand to clear my doubts and confusion by suggests several website and introduce various data repositories during the meeting.

### 7.1.2 Goclipse plugin compile error

Eclipse IDE could not compile and build my Go files, this is because the IDE couldn't find GOROOT in usr/local/go. The development activities cannot proceed and face impediment on executing critical success factors. The cause of the problem is Golang compiler executable doesn't possess a copy in usr/local/go, which caused Eclipse fail to compile Go file because couldn't file the compiler.

The problem is resolved with help of supervisors, he guides me to execute Linux command line to resolve the problem during FYP meeting. Moreover, he helps identify the root cause of problem with Google Hangout in the midnights.

### 7.1.3 Unclear and doubts on writing documentation

The problem encountered during writing documentation is unclear about the purpose and objectives of each section which leads to messy and poor content deliveries in writing. A certain standard and requirement should be achieved in writing the FYP document.

The problem is resolved with the help of supervisor as he patiently guide us to arrange the content layout of document and writing citation with references.

### 7.1.4 Difficulty on understand concurrent programming concepts

The problem encountered during coding process is to understand concurrent concepts. It took an enormous amount of time to implement the ideas of

Goroutine and Go channel into the program to achieve concurrency with Go programming language. This is because I do not possess the experiences and knowledge to build a concurrent program.

The problem is resolved with the help of official documentation and StackOverflow websites which provide clear explanation and enlightenment for me to understand the concepts and semantics of languages.

### 7.1.5 Difficulty on develop PG/pgSQL scripts on data migration.

The problem encountered during development process is writing PG/pgSQL scripts to perform data migration. The DML query requires to use *insert with select* query to retrieve primary key from each entity and insert as foreign key into specific table. In addition, the query is shall possess high throughput on data processing while maintaing the data consistency and validity during the migration process. The mentioned difficulty has caused impediment on development progress and stuck for a week.

The problem is posted on Stackoverflow forum as database question and it was discussed by various database expert with high reputation in the community. Ultimately, the issue is resolved with suggested answer provided and the query with JOIN works on my project.

### 7.1.6 Difficulty on perform database tuning.

The problem encountered during database tuning are listed as follow:

1. **Understand the risk of modification.** Modified number of maximum concurrent connection, parameter of shared memory buffer and maximum size of shared memory segment utilized by PostgreSQL database could result in database corruption and data loss. The PostgreSQL will running inconsistently and caused freezing or termination of any process if one of these value are not configure correctly.
2. **Limitation of knowledge on database system configuration.** Database tuning is an advanced techniques and incredibly difficult task perform by database administrator in mid-sized and large company to configure the database environment for situational usage. The process require deep understanding on hardware memory resources and database concepts to prevent the error on resource management between system and database.
3. **Performance bottleneck of programming langugae with database.** Each programming language utilized threads and stack differently. The maximum number of OS stacks and OS threads allow the language to utilized shall be carefully inspected and measure to prevent crash during the runtime. As an example, Go programming language only allow 1 GB of stack utilized on 64-bit system which indicates it only allow 10,000 threads (1,000,000 goroutines )to be assigned in each execution.

The understanding is established and discovered from Go official documentation and PostgreSQL 9.5 documentation to resolve this problem. The information provided in the documentation is clear and easy to be learnt as it helps resolve all the problem mentioned.

### 7.1.7 Distributed Programming

The project objectives had been reduce to concurrent programming on data processing instead of distributed programming.

It is possible to perform distributed programming on data processing activities in this project. However, the development required extra duration on experimentation, design and testing. Based on my current understanding and knowledge on the subject, it will require extra 3 months to develop distributed programming based program.

# Chapter 8

# Conclusions

## 8.1 Conclusions

In phase 1, we have review many concepts and addressed the details of concurrent programming language concepts.

The project objectives for Phase 1 are:

1. To learn and understand about Go and RUST programming language concepts and their concurrent processing features.
2. To conduct a comparison on Go programming language concepts in processing big data with different techniques.
3. To implement the handling of big data with PostgreSQL, an object-oriented relational database management system (OORDBMS)

What we have achieved on Phase 1:

1. We reviewed different concepts and characteristics of concurrent programming language.
2. We established the fundamentals of concurrent programming knowledge and possess confident advance to the next phase of development.
3. We established a development platform for concurrency programming.
4. We demonstrated the capability of concurrent programming language, which is provide better performance and throughput on data processing compare to sequential programming with results.

The project objectives for Phase 2 are:

1. To learn and understand the importance of data processing activities in data process cycle.
2. To understand the limitation of concurrent programming language and PostgreSQL database resource utilization.
3. To perform text substitution with data encoding on eliminate incompatible data type on data source.
4. To implement Go and Rust concurrent programming features on data processing activities.
5. To perform database cleansing on eliminate defects and error found in big data.
6. To develop Go and Rust program as ORM tool on data retrieval from CSV file and PostgreSQL database and map into object model.
7. To conduct database and query tuning to optimize performance on data processing execution.

8. To produce PL/pgSQL's DDL, DML and DCL scripts for database entity creation and database migration.
9. To develop a Go programming based data migration system to transfer data from legacy storage into new storage within PostgreSQL database.

What we have achieved on Phase 2:

1. We understand the purpose and benefits of each data processing activities in data process cycle.
2. We established understanding of limitation of concurrent programming language and PostgreSQL database resource utilization to prevent crashes in system execution.
3. We demonstrated the capabilities of data encoding on text substitution of raw CSV files with regular expression as input command.
4. We demonstrated strength and limitation of Go and Rust concurrent programming features on data retrieval through execution times and language structure.
5. We have implemented Go program to eliminate NULL values in every single row and perform data standardization to increase usability of data.
6. We have conducted database normalization to eliminate data redundancy, resolve anomalies and improve data integrity.
7. We have implemented Go and Rust program as ORM tool to retrieve 4 millions of data from CSV file and PostgreSQL database in sequential and concurrent manner.
8. We have conducted database and query tuning to optimize data processing performance and allow more threads to establish connection with database concurrently.

9. We have developed PL/pgSQL's DDL, DML and DCL scripts to create database table, establish relationship between entity and perform database migration for 30,000 rows of data.
10. We have developed a Go program to migrate 4 millions rows of data from legacy storage to new storage within PostgreSQL database without violates data consistency, validity and consistency.
11. We have prove concurrent programming has better performance than sequential programming.
12. We have prove Go programming language has better performance than Rust programming language on data processing.

## 8.2 Lessons Learned

1. **Data science knowledge.** Data science is being use as competitive weapon and it transform the way how companies operate with information. It is a totally new knowledge and experience for me as Software Engineering student to learn and explore.
2. **Concurrent programming concepts.** Concurrent concepts is difficult to be understand and never thought in subject syllabus. Learning the art of concurrent programming for building applications in this project provide satisfaction and motivation to fulfill my desire to build a real-time system.
3. **Consistent update with FYP Supervisor.** FYP supervisor ensure the project is on track and doing right. It is essential to make available time for consultation and rapidly update the progress for supervisor via email

to enhance the work quality. Moreover, FYP supervisor review my work ensure the time and resource is not waste on doing the wrong task.

4. **Ubuntu Operating System.** The project allow me to learn Linux Bash commands through practice. The Ubuntu operating system is found not difficult to be learned and it is more safety, reliable and consistent to conduct development activities due to its lightweight.

5. **PostgreSQL database.** The project allow me to learn the basics of PostgreSQL database configuration and developed PL/pgSQL scripts through development activities. It is enjoyable and joyful to learn the world's most advanced open source database and establish deeper understanding on database's feature. Other than that, the feeling of accomplishment emerged in my mind as I possess the flexibility to manipulate database settings and communicate with data source through query.

6. **Database normalization.** The project allow me to learn and implement the normalization rules to perform excellent data management with good database designs. My supervisor patiently guides the important procedure to perform database normalization and data migration during FYP meeting and constantly provide example to perform data cleaning.

## 8.3 Recommendations for Future Work

### 8.3.1 Phase 1

1. **GORM for CRUD on data processing.** GORM is an Object-relational mapping (ORM) library for Golang that converting data from incompatible files types into struct or interface. For instance, this project does not use GORM to import data and possess poor readability, error handling and maintainability in program. It is recommend to import data with GORM package because it supports auto migration, associations with database and every features are tested.
2. **Benchmark on language performance comparison.** Although this project possess well-defined of benchmarking on database table spacing, hardware configuration and amount of query execution on data retrieval to conduct language performance comparison. These benchmarks are insufficient to determine the accurateness of programming language performance. This is because the CPU usage might be running on other processes or program while conducting the performance test. It is recommend to unified number of processes running in background and programming style for performance comparison between different concurrent programming languages.
3. **Data quality.** Although this project use data validation to identify raw dataset quality. The method is insufficient to ensure data obtained is valid, complete and accurate to be processed. It is recommend to use several scripting language such as Python and Perl to identify internal data consistency and validity.

### 8.3.2 Phase 2

1. **Company database normalized design.** Although the company datasets is normalized correctly, there are several transitive functional dependencies found in the table and required to be divide with 3NF (Third Normal Form) rules. The database still possess insert, delete and update anomalies on **company** tables and require extra efforts on reduce the complexity of tables.
2. **Data types for date attributes.** Although the data transformation, data parsing and data migration of company datasets are conducted successfully. The date values are declared as *VARCHAR* in PostgreSQL database and declared as *String* in Go and Rust program to reduce errors on date format conversion. The declaration increase difficulty on sorting and does not comply to unambiguous input format (ISO 8601). It is recommend to use *date* data types to store date values for better data analysis and processing results.
3. **Code structure of data cleaning parser.** Although the data cleaning parser is able to eliminate NULL values in every single rows and provide standardization support on each field, the program use more than 40 if-else statement within a loops and it reduces the performance on program execution. It is recommend to use better control flow statement to reduce the effort on data checking and resources utilization for the program.

# Bibliography

[1] Ibrahim Abaker Targio Hashem et al. *The rise of big data on cloud computing: Review and open research issues*, 2014. URL `https://www.acm.org/publications/authors/reference-formatting`. Retrieved on 28/07/2017.

[2] David Geer. *Chip Makers Turn to Multicore Processors*, 2015. URL `http://ieeexplore.ieee.org/document/1430623/?part=1`. Retrieved on 28/07/2017.

[3] Bob Pike. *Google Tech Talk*, 2005. URL `http://9p.io/sources/contrib/ericvh/go-plan9/doc/go_talk-20091030.pdf`. Retrieved on 28/07/2017.

[4] Schneider F. B. Andrew G. R. Concept and notation of concurrent programming. *Computing Surveys*, pages 1–2, 1983. doi: http://babel.ls.fi.upm.es/teaching/concurrencia/material/concepts_and_notations.pdf. Retrieve on 04/08/2017.

[5] M. Ben-Ari. *Principle of Concurrent Programming.* Pearson 2nd Edition, 2005. ISBN 9780321312839.

[6] Kurt Guntheroth. *Why did Google develop Go?*, 2017. URL `https://www.quora.com/Why-did-Google-develop-Go/`. Retrieved on 29/07/2017.

[7] golang.org. The go programming language, 1999. URL `https://golang.org/`. Retrieved on 29/07/2017.

[8] GCC Organization. Ada, go and objective-c++ are not default languages, 2011. URL `https://gcc.gnu.org/install/configure.html`. Retrieved on 29/07/2017.

[9] Uwe R.Zimmer Benjamin J.L. Wang. College of engineering and computer sciences the australian national university. *Pure Concurrent Programming*, 2017. URL `http://ieeexplore.ieee.org/abstract/document/7965126/?part=1`. Retrieved on 29/07/2017.

[10] Britannica. Control structures, 2017. URL https://www.britannica.com/technology/computer-programming-language/Control-structures#ref849883. Retrieved on 05/08/2017.

[11] Joe Armstrong. Sequential vs concurrent programming languages. *Programming Erlang 2nd Edition*, 2013. doi: https://www.safaribooksonline.com/library/view/programming-erlang-2nd/9781941222454/f_0018.html.

[12] Brian Harvey and Matthew Wright. Sequential programming. *Simply Scheme: Introducing Computer Science*, 1999. doi: https://www.safaribooksonline.com/library/view/programming-erlang-2nd/9781941222454/f_0018.html.

[13] Herb Sutter. Will concurrency be the next revolution in software development?, 2005. URL http://www.drdobbs.com/the-concurrency-revolution/184401916. Retrieved on 05/08/2017.

[14] Jan Stenberg. Concurrent and distributed programming in the future, 2017. URL https://www.infoq.com/news/2017/03/distributed-programming-qcon. Retrieved on 06/08/2017.

[15] Gul Agha. Concurrent object-oriented programming. *Magazine Communications of the ACM*, pages 125–141, 1990. doi: 10.1145/83880.84528. URL http://dl.acm.org/citation.cfm?id=84528. Retrieved on 06/08/2017.

[16] Theodore Norvell. What is concurrent programming? pages 1–2, 2009. URL http://www.engr.mun.ca/~theo/Courses/cp/pub/cp0.pdf. Retrieved on 06/08/2017.

[17] Herb Sutter and James Larus. Software and concurrency revolution. *Queue – Multiprocessors*, pages 59–60, 2005. doi: 10.1145/1095408.1095421. URL http://dl.acm.org/citation.cfm?id=1095421. Retrieved on 06/08/2017.

[18] Tribaud. Top programming language to learn in 2017, 2017. URL https://www.codingame.com/blog/top-programming-languages-to-learn-in-2017. Retrieved on 07/08/2017.

[19] Horning J.J. Distributed processes: A concurrent programming concept. *Communication of the ACM*, 1978. doi: 10.1145/359642.359651. URL http://dl.acm.org/citation.cfm?id=359651. Retrieved on 07/08/2017.

[20] PostgreSQL Global Development Group. What is postgresql? *PostgreSQL 9.5.9 Documentation: Preface.*, 2017. URL `https://www.postgresql.org/docs/9.5/static/intro-whatis.html`. Retrieved on 08/09/2017.

[21] What is postgresql?, 2017. URL `http://www.postgresqltutorial.com/what-is-postgresql/`. Retrieved on 18/08/2017.

[22] Dibyendu Majumdar. A quick survey of multiversion concurrency algorithms. *MVCC Survey*, 2006. URL `forge.ow2.org/docman/view.php/237/132/mvcc-survey.pdf`. Retrieved on 19/08/2017.

[23] Sirish et al. Telegraphcq: continuous dataflow processing. *SIGMOD '03 Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, page 668, 2003. doi: 10.1145/872757.872857. URL `http://dl.acm.org/citation.cfm?id=872857`. Retrieved on 19/08/2017.

[24] George et al. Predictable performance and high query concurrency for data analytics. *The VLDB Journal*, pages 227–248, 2011. doi: 10.1007/s00778-011-0221-2. URL `http://delivery.acm.org.proxyvlib.mmu.edu.my/10.1145/1970000/1969355/778_2011_Article_221.pdf?ip=203.106.62.29&id=1969355&acc=ACTIVE%20SERVICE&key=69AF3716A20387ED%2EE854CB4DB8D6D408%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=801067487&CFTOKEN=72015032&__acm__=1503554298_5a4d19e623542c1086bd72577837f01a#URLTOKEN#`. Retrieved on 19/08/2017.

[25] PostgreSQL Global Development Group. Concurrency control. *Introduction*, 2017. URL `https://www.postgresql.org/docs/9.5/static/mvcc-intro.html`. Retrieved on 10/09/2017.

[26] PostgreSQL Global Development Group. Postgresql concurrency with mvcc. *How MVCC Works*, 2017. URL `https://devcenter.heroku.com/articles/postgresql-concurrency`. Retrieved on 10/09/2017.

[27] PostgreSQL Global Development Group. Linux downloads. *PostgreSQL Support Documentation*, 2017. URL `https://www.postgresql.org/download/linux/ubuntu/`. Retrieved on 10/09/2017.

[28] Rob Pike. Expressiveness of go, 2010. URL `http://www.intercapedine.net/documenti/ExpressivenessOfGo.pdf`. Retrieved on 07/08/2017.

[29] Forsby Filip and Persson Martin. Evaluation of golang for high performance scalable radio access systems, 2015. URL http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A873124&dswid=-8907#sthash.gj7rKTc5.dpbs. Retrieved on 07/08/2017.

[30] Slavomir Polak and Tomas Pitner. Text processing performance in go language. pages 149–152, 2014. URL http://www.cssi-morava.cz/new/doc/IT2014/sbornik.pdf#page=149. Retrieved on 08/08/2017.

[31] Pravenda Singh. Implementing an intelligent version of the classical sliding-puzzle game for unix terminals using golang's concurrency primitives. 2015. URL https://arxiv.org/pdf/1503.08345.pdf. Retrieved on 08/08/2017.

[32] Hoare. The rust programming language, 2013. URL http://www.rust-lang.org/. Retrieved on 08/08/2017.

[33] Eric Holk et al. Gpu programming in rust: Implementing high-level abstractions in a systems-level language. *Indiana University*, 2013. doi: 10.1109/IPDPSW.2013.173. URL http://ieeexplore.ieee.org/abstract/document/6650903. Retrieved on 08/08/2017.

[34] Eric Reed. Patina: A formalization of the rust programming language. university of washington. 2015. URL https://www.cs.washington.edu/tr/2015/03/UW-CSE-15-03-02.pdf. Retrieved on 08/08/2017.

[35] Sebastian Nanz et al. Design of an empirical study for comparing the usability of concurrent programming languages. *Information of Software Technology*, 55(7):1304–1315, 2013. URL http://www.sciencedirect.com/science/article/pii/S0950584912001802. Retrieved on 09/08/2017.

[36] Ehud Shapiro. Embeddings among concurrent programming languages (preliminary version). *Lecture Notes in Computer Science*, 630, 2006. URL https://link.springer.com/chapter/10.1007%2FBFb0084811?LI=true. Retrieved on 09/08/2017.

[37] Ehud Shapiro. Separating concurrent languages with categories of language embeddings. 2006. URL https://pdfs.semanticscholar.org/7d2a/9a3954922741472f5ff06d2c1dafb258420e.pdf. Retrieved on 09/08/2017.

[38] Ehud Shapiro. The family of concurrency programming languages. *ACM Computing Surveys (CSUR)*, 21(3):413–510, 1989. URL `http://dl.acm.org/citation.cfm?id=72555`. Retrieved on 10/08/2017.

[39] Maurice Herlihy. A methodology for implementing highly concurrent data objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(5), 1993. doi: 10.1145/161468.161469. URL `http://dl.acm.org/citation.cfm?id=161469`. Retrieved on 13/08/2017.

[40] Nenad Medvidovic and Richard N. Taylor. A framework for classifying and comparing architecture description languages. *ACM SIGSOFT Software Engineering Notes Homepage*, 22(6):60–76, 1997. doi: 10.1145/267896.267903. URL `http://dl.acm.org/citation.cfm?id=267903`. Retrieved on 13/08/2017.

[41] Stotts P.D. A comparative survey of concurrent programming languages. *ACM SIGPLAN*, 17:50–61, 1982. doi: 10.1109/2.73. URL `http://research.cs.queensu.ca/home/cordy/cisc860/Biblio/drb/CE/stotts82.pdf`. Retrieved on 15/08/2017.

[42] Vincent W.F. A comparison of implicit and explicit parallel programming. *Journal of Parallel and Distributed Computing*, 34(1):50–65, 1996. URL `http://www.sciencedirect.com/science/article/pii/S0743731596900453`. Retrieved on 15/08/2017.

[43] H.W. Loidl. Comparing parallel functional languages: Programming and performance. *Higher-Order and Symbolic Computation*, 16(3):203–251, 2003. doi: 10.1023/A:1025641323400. URL `http://dl.acm.org/citation.cfm?id=940872`. Retrieved on 15/08/2017.

[44] Simon Marlow. Distributed programming. *Parallel and Concurrent Programming in Haskell*, 2013. URL `https://www.safaribooksonline.com/library/view/parallel-and-concurrent/9781449335939/ch14.html`. Retrieved on 08/08/2017.

[45] Stackoverflow. Ii. most loved, dreaded, and wanted., developer survey results, 2016. URL `https://insights.stackoverflow.com/survey/2016`. Retrieved on 08/08/2017.

[46] Ty. Rust vs go adventures in error handling, 2017. URL `https://insights.stackoverflow.com/survey/2016`. Retrieved on 08/08/2017.

[47] Tiobe Software BV. The go programming language. *TIOBE Index*, 2017. URL `https://www.tiobe.com/tiobe-index/go/`. Retrieved on 11/09/2017.

[48] Stackoverflow. Most love programming language. *Developer Survey Results 2017*, 2017. URL `https://insights.stackoverflow.com/survey/2017`. Retrieved on 11/09/2017.

[49] golang.org. The go programming language, 2017. URL `https://golang.org/doc/`. Retrieved on 08/08/2017.

[50] rustlang.org. The rust programming language, 2017. URL `https://www.rust-lang.org/en-US/`. Retrieved on 08/08/2017.

[51] Caleb Doxsey. *Concurrency.* An Introduction to Programming in Go. 2017. URL `https://www.golang-book.com/books/intro/10`. Retrieved on 08/08/2017.

[52] Chua Yong Wen. Appreciating rust's memory safety guarantees, 2017. URL `https://blog.gds-gov.tech/appreciating-rust-memory-safety-438301fee097`. Retrieved on 09/08/2017.

[53] Hackernews. Rust vs go, 2017. URL `https://news.ycombinator.com/item?id=13430108`. Retrieved on 09/08/2017.

[54] Arild Nilsen. Communication sequential process (csp). *An alternative to the actor model*, 2017. URL `https://arild.github.io/csp-presentation/`. Retrieved on 11/09/2017.

[55] Will Yager. The problem. *Why Go is no good*, 2017. URL `http://yager.io/programming/go.html`. Retrieved on 11/09/2017.

[56] Techopedia. Actor model. *Programming Tools*, 2017. URL `https://www.techopedia.com/definition/25150/actor-model`. Retrieved on 11/09/2017.

[57] Ticki. Why should i use rust? *The RUST programming language*, 2016. URL `https://www.reddit.com/r/rust/comments/4l44z3/why_should_i_use_rust/`. Retrieved on 11/09/2017.

[58] Rust lang organization. How do i map object-oriented concepts to rust? *Design Patterns*, 2017. URL `https://www.rust-lang.org/en-US/faq.html#how-do-i-map-object-oriented-concepts-to-rust`. Retrieved on 11/09/2017.

[59] Simon Hoare. What is the difference between unix, linux and ubuntu? *Ask Ubuntu Forum*, 2012. URL https://askubuntu.com/questions/183723/whats-the-difference-between-unix-linux-and-ubuntu. Retrieved on 08/09/2017.

[60] Invert. Why is ubuntu is more secure than windows or mac os x? *Ask Ubuntu Forum*, 2010. URL https://askubuntu.com/questions/1069/why-is-ubuntu-more-secure-than-windows-or-mac-os-x. Retrieved on 08/09/2017.

[61] Katherine Noyes. Why linux is more secure than windows? *Linux Line*, 2017. URL https://www.pcworld.com/article/202452/why_linux_is_more_secure_than_windows.html. Retrieved on 08/09/2017.

[62] James McInnes. What are key differences between unix and ms-dos? *Programming language comparisons*, 2015. URL https://www.quora.com/What-are-the-key-differences-between-Unix-and-MS-DOS. Retrieved on 08/09/2017.

[63] Spehro Pefhany. Why is printf() bad for debugging embedded systems? *Electrical Engineering Stack Exchange*, 2014. URL https://electronics.stackexchange.com/questions/105283/why-is-printf-bad-for-debugging-embedded-systems. Retrieved on 11/09/2017.

[64] The LTTng project. What is tracing? *Trace Compass Documentation*, 2017. URL http://lttng.org/docs/v2.9/#doc-what-is-tracing. Retrieved on 11/09/2017.

[65] Tutorialpoint. What is gnu debugger? *How GDB debugs?*, 2017. URL https://www.tutorialspoint.com/gnu_debugger/what_is_gdb.htm. Retrieved on 11/09/2017.

[66] Tutorialspoint. Data profile validation. *ETL Testing - Data Completeness*, 2017. URL https://www.tutorialspoint.com/etl_testing/etl_testing_data_completeness.htm. Retrieved on 03/02/2018.

[67] Techopedia. Data redundancy. *Unified Communication*, 2018. URL https://www.techopedia.com/definition/18707/data-redundancy. Retrieved on 03/02/2018.

[68] Margaret Rouse. Encoding and decoding. *Programming*, 2005. URL http://searchnetworking.techtarget.com/definition/encoding-and-decoding. Retrieved on 03/02/2018.

[69] Justin Eillingwood. The basics of using the sed stream editor to manipulate text in linux. *Linux Basics and Commands*, 2013. URL https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-editor-to-manipulate-text-in-linux. Retrieved on 03/02/2018.

[70] MuleSoft. Data transformation. *SQA Resources*, 2018. URL https://www.mulesoft.com/resources/esb/data-transformation. Retrieved on 03/02/2018.

[71] Satis. Introduction. *What is an ORM and where can I learn about it*, 2009. URL https://stackoverflow.com/questions/1279613/what-is-an-orm-and-where-can-i-learn-more-about-it. Retrieved on 03/02/2018.

[72] Experian. What is data cleansing. *Data Cleansing*, 2014. URL https://www.edq.com/uk/glossary/data-cleansing/. Retrieved on 03/02/2018.

[73] Vikrant Oberoi. What is data redundancy in a dbms? what is a simple explanation? *Database management software of Quora*, 2017. URL https://www.quora.com/What-is-data-redundancy-in-a-DBMS-What-is-a-simple-explanation. Retrieved on 04/02/2018.

[74] Margaret Rouse. Definition. *data migration*, 2017. URL http://searchstorage.techtarget.com/definition/data-migration. Retrieved on 04/02/2018.

# Appendices

# Appendix A

# Infrastructure Setup and Installation

## A.1 Linux command for Go compiler installation

```

============================================================
(1) DOWNLOAD GOLANG go1.8.3.linux-amd64.tar.gz
AT URL https://golang.org/dl/ USING wget IN TERMINAL
============================================================

yinghua@yinghua-NL8C:~/Downloads/temp$ wget -c https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz
...
go1.8.3.linux-amd64 100%[===================>]  85.86M  5.93MB/s    in 14s
yinghua@yinghua-NL8C:~/Downloads/temp$

===========================================================
(2) EXTRACT DOWNLOADED SOURCE
===========================================================
yinghua@yinghua-NL8C:~/Downloads/temp$ tar -xzvf go1.8.3.linux-amd64.tar.gz
....
yinghua@yinghua-NL8C:~/Downloads/temp$

===========================================================
(3) MOVE AND RENAME GOLANG DIRECTORY
===========================================================
yinghua@yinghua-NL8C:~/Downloads/temp$ mkdir -p ~/Desktop/apps/golang1.8.3
yinghua@yinghua-NL8C:~/Downloads/temp$ mv go ~/apps/golang1.8.3
yinghua@yinghua-NL8C:~/Downloads/temp$

===========================================================
(4) CHECK GOLANG DIRECTORY
===========================================================
yinghua@yinghua-NL8C:~/Downloads/temp$ cd ~/Desktop/apps/
yinghua@yinghua-NL8C:~/Desktop/apps$ ls -l
total 24
drwxr-xr-x  8 yinghua yinghua 4096 Sep 11 03:03 eclipse-oxygen
drwxrwxr-x  4 yinghua yinghua 4096 Sep  7 23:19 eclipse-workspace
drwxr-xr-x 11 yinghua yinghua 4096 May 25 02:16 golang1.8.3

===========================================================
(5) GO INTO GOLANG INSTALLED DIRECTORY
===========================================================
yinghua@yinghua-NL8C:~/Desktop/apps$ cd golang1.8.3/
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$ ls -l
total 160
drwxr-xr-x  2 yinghua yinghua  4096 May 25 02:15 api
-rw-r--r--  1 yinghua yinghua 33243 May 25 02:15 AUTHORS
drwxr-xr-x  2 yinghua yinghua  4096 May 25 02:16 bin
drwxr-xr-x  4 yinghua yinghua  4096 May 25 02:16 blog
```

```
-rw-r--r--  1 yinghua yinghua  1366 May 25 02:15 CONTRIBUTING.md
....
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$

=========================================================
(5.1) CHECK GOLANG EXECUTABLES
=========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$ ls -al bin
total 28120
drwxr-xr-x  2 yinghua yinghua     4096 May 25 02:16 .
drwxr-xr-x 11 yinghua yinghua     4096 May 25 02:16 ..
-rwxr-xr-x  1 yinghua yinghua 10073055 May 25 02:16 go
-rwxr-xr-x  1 yinghua yinghua 15226597 May 25 02:16 godoc
-rwxr-xr-x  1 yinghua yinghua  3481554 May 25 02:16 gofmt
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$


=========================================================
(5.2) CHECK GOLANG LIBRARIES
=========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$ ls -al lib
total 12
drwxr-xr-x  3 yinghua yinghua 4096 May 25 02:15 .
drwxr-xr-x 11 yinghua yinghua 4096 May 25 02:16 ..
drwxr-xr-x  2 yinghua yinghua 4096 May 25 02:15 time
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$

=========================================================
(5.3) CHECK GOLANG PACKAGES
=========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$ ls -al pkg
total 28
drwxr-xr-x  7 yinghua yinghua 4096 May 25 02:16 .
drwxr-xr-x 11 yinghua yinghua 4096 May 25 02:16 ..
drwxr-xr-x  2 yinghua yinghua 4096 May 25 02:15 include
drwxr-xr-x 30 yinghua yinghua 4096 May 25 02:16 linux_amd64
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$
....

==========================================================
(6) SET PATH TO GOLANG BINARY EXECUTABLES AND EXPORT PATH
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$ cd bin
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ pwd
/home/yinghua/Desktop/apps/golang1.8.3/bin
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ export PATH=/home/yinghua/Desktop/apps/golang1.8.3/bin:$PATH
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$

==========================================================
(6.1) CHECK ADDED GOLANG PATH
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ echo $PATH
/home/yinghua/Desktop/apps/golang1.8.3/bin: <=== PATH ADDED
/home/yinghua/.cargo/bin:
/home/yinghua/bin:
/home/yinghua/.local/bin:
/usr/local/sbin:
....
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$


==========================================================
(6.2) SET GOROOT AND GOPATH
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ mkdir ~/Desktop/apps/gopath
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ ls -al ~/Desktop/apps
total 24
drwxr-xr-x  8 yinghua yinghua 4096 Sep 11 03:03 eclipse-oxygen
drwxrwxr-x  4 yinghua yinghua 4096 Sep  7 23:19 eclipse-workspace
drwxr-xr-x 11 yinghua yinghua 4096 May 25 02:16 golang1.8.3
drwxrwxr-x  5 yinghua yinghua 4096 Sep  7 23:05 gopath

yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ export GOROOT=/home/yinghua/Desktop/apps/golang1.8.3
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ export GOROOT=/home/yinghua/Desktop/apps/gopath
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ export PATH=$GOPATH/bin:$PATH

==========================================================
(6.3) CHECK GOROOT AND GOPATH
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ echo $GOROOT
/home/yinghua/Desktop/apps/golang1.8.3
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ echo $GOPATH
/home/yinghua/Desktop/apps/gopath
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$

==========================================================
(6.4) APPLY SYSTEM UPDATES
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ sudo updatedb
[sudo] password for yinghua:
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ sudo ldconfig
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ sudo depmod
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$

============================================================
(7) APPEND PATH TO USER PROFILE .bashrc FILE
============================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ nano ~/.bashrc

# ======= ADDED BY CYH INTO ~/.bashrc ============
# added by CYH for Golang1.8.3 Compiler
export GOROOT=/home/yinghua/Desktop/apps/golang1.8.3
export GOPATH=/home/yinghua/Desktop/apps/gopath
export PATH=$GOROOT/bin:$GOPATH/bin:$PATH

============================================================
(8) CHECK GO EXECUTABLE AND GO VERSION
============================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ which go
/home/yinghua/Desktop/apps/golang1.8.3/bin/go
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ go version
go version go1.8.3 linux/amd64
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$

===========================================================
(9) TEST GO EXECUTABLE
===========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ go help
Go is a tool for managing Go source code.
.....

===========================================================
(10) GO TO GOPATH DIRECTORY TO INSTALL TOOLS
===========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3/bin$ cd ..
yinghua@yinghua-NL8C:~/Desktop/apps/golang1.8.3$  cd ..
yinghua@yinghua-NL8C:~/Desktop/apps$  cd gopath/
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$  ls -l
total 0
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$

===========================================================
(11) DOWNLOAD GO PACKAGE TOOLS (EXECUTABLES)
===========================================================
Use git to download go libraries (gocode, golint, guru, goimports, gorename, godef)

yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get github.com/nsf/gocode
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get github.com/golang/lint/golint
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get golang.org/x/tools/cmd/guru
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get golang.org/x/tools/cmd/goimports
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get golang.org/x/tools/cmd/gorename

============================================================
(11.1) DOWNLOAD GODEF GOMETALINTER
===========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get github.com/rogpeppe/godef
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ go get -u gopkg.in/alecthomas/gometalinter.v1

============================================================
(11.2) EXECUTE GOMETALINTER
============================================================
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ cd bin
yinghua@yinghua-NL8C:~/Desktop/apps/gopath/bin$ gometalinter.v1 --install
.....
gocyclo
goimports
interfacer
safesql
unparam
wruslan@dell-ub1604-64b:~/apps/gopath/bin$

============================================================
(11.3) CHECK INSTALLED PACKAGES (LIBRARIES)
============================================================
yinghua@yinghua-NL8C:~/Desktop/apps/gopath$ ls -al bin
total 154644
drwxrwxr-x 2 yinghua yinghua     4096 Sep  7 23:09 .
drwxrwxr-x 5 yinghua yinghua     4096 Sep  7 23:05 ..
-rwxrwxr-x 1 yinghua yinghua  7521174 Sep  7 23:09 gas
-rwxrwxr-x 1 yinghua yinghua 10521898 Sep  7 23:05 gocode <=== FOR ECLIPSE IDE
-rwxrwxr-x 1 yinghua yinghua  3015835 Sep  7 23:09 goconst
-rwxrwxr-x 1 yinghua yinghua  2453860 Sep  7 23:09 gocyclo
-rwxrwxr-x 1 yinghua yinghua  5503061 Sep  7 23:06 godef <=== FOR ECLIPSE IDE
-rwxrwxr-x 1 yinghua yinghua  4898036 Sep  7 23:09 goimports
-rwxrwxr-x 1 yinghua yinghua  8309030 Sep  7 23:05 guru   <=== FOR ECLIPSE IDE
-rwxrwxr-x 1 yinghua yinghua  2494881 Sep  7 23:09 ineffassign
.....
```

```

==========================================================
END
==========================================================
```

Listing A.1: Linux command for Golang compiler installation

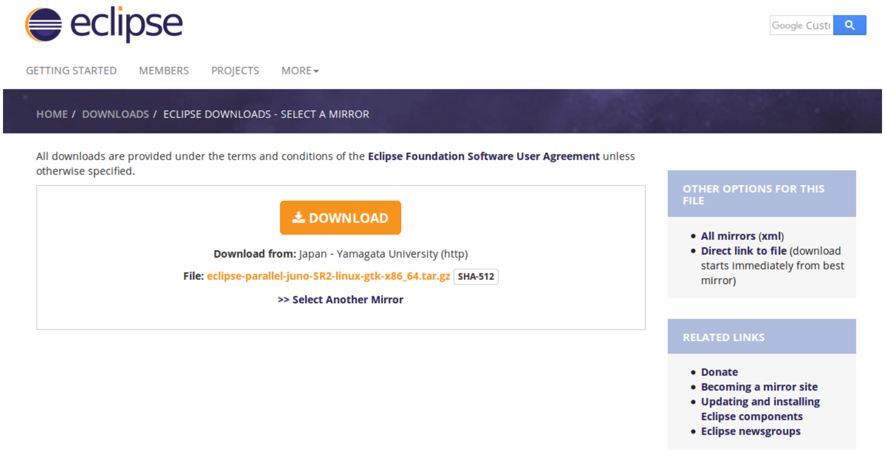
## A.2 Linux command for Rust compiler installation

```

==========================================================
(1) INSTALL COMMANDLINE Rust toolchain
==========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ curl https://sh.rustup.rs -sSf | sh

Welcome to Rust!

This will download and install the official compiler for the Rust programming
language, and its package manager, Cargo.

It will add the cargo, rustc, rustup and other commands to Cargo’s bin
directory, located at:

/home/yinghua/.cargo/bin

This path will then be added to your PATH environment variable by modifying the
profile file located at:

/home/yinghua/.profile

You can uninstall at any time with rustup self uninstall and these changes will
be reverted.

Current installation options:

default host triple: i686-unknown-linux-gnu
default toolchain: stable
modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation

info: syncing channel updates for ’stable-i686-unknown-linux-gnu’
156.7 KiB / 156.7 KiB (100 %) 126.1 KiB/s ETA:   0 s
info: downloading component ’rustc’
38.9 MiB /  38.9 MiB (100 %) 505.6 KiB/s ETA:   0 s
......

stable installed - rustc 1.17.0 (56124baa9 2017-04-24)


Rust is installed now. Great!

To get started you need Cargo’s bin directory in your PATH environment
variable. Next time you log in this will be done automatically.

To configure your current shell run source $HOME/.cargo/env
yinghua@yinghua-NL8C:~/Desktop/apps/rust$

==========================================================
(2) EXPORT RUST EXECUTABLE TO PATH
==========================================================

yinghua@yinghua-NL8C:~$ cd ~/Desktop/apps/rust/
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ rustc --version
rustc 1.20.0 (f3d6973f4 2017-08-27)
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ sudo updatedb
[sudo] password for yinghua:
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ locate bin/rustc
/home/yinghua/.cargo/bin/rustc
/home/yinghua/.rustup/toolchains/stable-x86_64-unknown-linux-gnu/bin/rustc
/usr/bin/rustc
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ export PATH=$PATH:$HOME/.cargo/bin
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ rustup component add rust-src
info: downloading component ’rust-src’
30.4 MiB /  30.4 MiB (100 %) 371.2 KiB/s ETA:   0 s
```

```
info: installing component ’rust-src’

===========================================================
(3) INSTALL RACER
===========================================================
yinghua@yinghua-NL8C:~$ cargo install racer
Updating registry ‘https://github.com/rust-lang/crates.io-index‘
.....
Finished release [optimized + debuginfo] target(s) in 928.10 secs
Installing /home/yinghua/.cargo/bin/racer
yinghua@yinghua-NL8C:~$

===========================================================
(4) INSTALL RUSTFMT
===========================================================
yinghua@yinghua-NL8C:~$ cargo install rustfmt
Updating registry ‘https://github.com/rust-lang/crates.io-index‘
....
Finished release [optimized] target(s) in 786.15 secs
Installing /home/yinghua/.cargo/bin/cargo-fmt
Installing /home/yinghua/.cargo/bin/rustfmt
yinghua@yinghua-NL8C:~$

===========================================================
(5) INSTALL RAINICORN
===========================================================
yinghua@yinghua-NL8C:~$ cargo install --git https://github.com/RustDT/Rainicorn --tag version_1.x
The program ’cargo’ is currently not installed. You can install it by typing:
sudo apt install cargo
yinghua@yinghua-NL8C:~$ export PATH=$PATH:$HOME/.cargo/bin
yinghua@yinghua-NL8C:~$ which cargo
/home/yinghua/.cargo/bin/cargo

yinghua@yinghua-NL8C:~$ cargo install --git https://github.com/RustDT/Rainicorn --tag version_1.x
Updating git repository ‘https://github.com/RustDT/Rainicorn‘
Installing rainicorn v1.3.0 (https://github.com/RustDT/Rainicorn?tag=version_1.x#365f819b)
Updating registry ‘https://github.com/rust-lang/crates.io-index‘
.....
Finished release [optimized] target(s) in 527.77 secs
Installing /home/yinghua/.cargo/bin/parse_describe
yinghua@yinghua-NL8C:~$

===========================================================
(6) CHECK RUST EXECUTABLES (11 NOS.)
===========================================================
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ which cargo
/home/yinghua/.cargo/bin/cargo
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ rustc --version
rustc 1.20.0 (f3d6973f4 2017-08-27)
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ which rustc
/home/yinghua/.cargo/bin/rustc
yinghua@yinghua-NL8C:~/Desktop/apps/rust$ ls -al /home/yinghua/.cargo/bin/
total 145404
drwxrwxr-x 2 yinghua yinghua     4096 Sep  7 22:39 .
drwxrwxr-x 5 yinghua yinghua     4096 Sep  7 22:36 ..
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 cargo
-rwxrwxr-x 1 yinghua yinghua  4126864 Sep  7 22:39 cargo-fmt
-rwxrwxr-x 1 yinghua yinghua  3828768 Sep  7 22:38 parse_describe
-rwxrwxr-x 1 yinghua yinghua 46240312 Sep  7 22:34 racer
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rls
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rustc
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rustdoc
-rwxrwxr-x 1 yinghua yinghua  8291104 Sep  7 22:39 rustfmt
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rust-gdb
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rust-lldb
-rwxr-xr-x 7 yinghua yinghua 12340104 Sep  7 22:19 rustup
yinghua@yinghua-NL8C:~/Desktop/apps/rust$

===========================================================
END
===========================================================
```

Listing A.2: Linux command for Rust compiler installation

## A.3 Eclipse IDE installation



All downloads are provided under the terms and conditions of the **Eclipse Foundation Software User Agreement** unless otherwise specified.

**Download from:** Japan - Yamagata University (http)

**File:** eclipse-parallel-juno-SR2-linux-gtk-x86_64.tar.gz

- **All mirrors** (xml)
- **Direct link to file** (download starts immediately from best mirror)

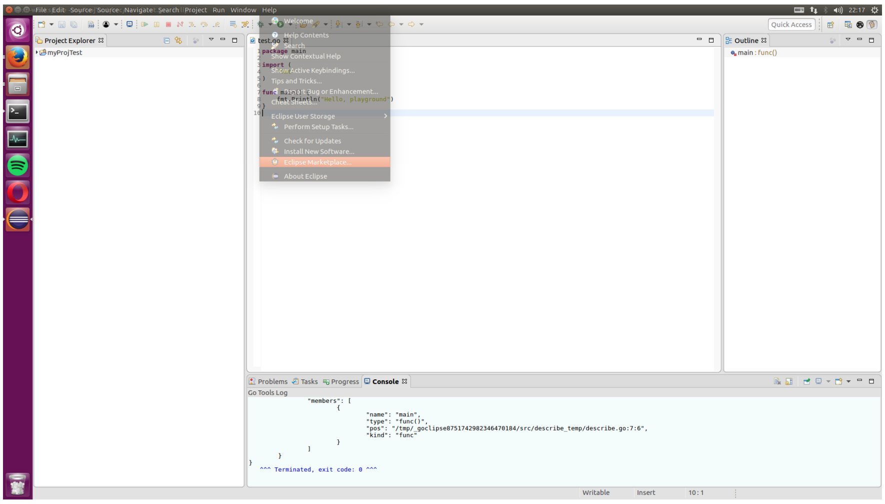
- **Donate**
- **Becoming a mirror site**
- **Updating and installing Eclipse components**
- **Eclipse newsgroups**

FIGURE A.1: Eclipse Oxygen Download Official Website

Ensure the Eclipse IDE version selected is compatible with 64-bit Ubuntu Operating System.

# A.4 GoClipse plugin for Eclipse IDE installation

## A.4.1 Eclipse Marketplace



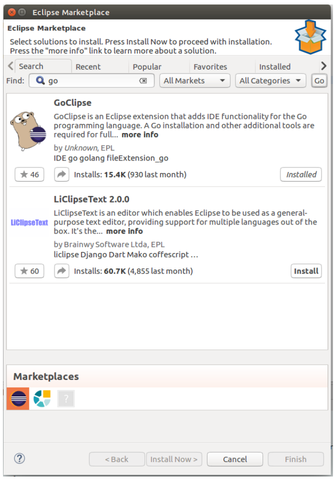
```
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello, playground")
}
```

```
		"members": [
			{
				"name": "main",
				"type": "func()",
				"pos": "/tmp/_goclipse8751742982346470184/src/describe_temp/describe.go:7:6",
				"kind": "func"
			}
		]
	}
}
   ^^^ Terminated, exit code: 0 ^^^
```

Figure A.2: Eclipse IDE Marketplace

Open Eclipse Marketplace from Help and select Eclipse Marketplace to search for GoClipse plugin.

### A.4.2 Search Marketplace



Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

**GoClipse**

GoClipse is an Eclipse extension that adds IDE functionality for the Go programming language. A Go installation and other additional tools are required for full... **more info**

by *Unknown*, EPL
IDE go golang fileExtension_go

Installs: **15.4K** (930 last month)

**LiClipseText 2.0.0**

LiClipseText is an editor which enables Eclipse to be used as a general-purpose text editor, providing support for multiple languages out of the box. It's the... **more info**

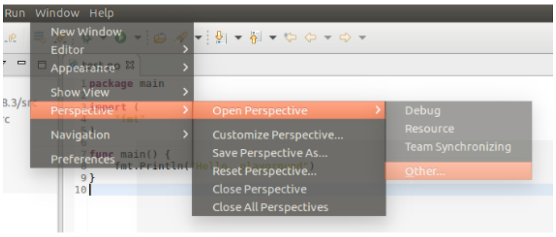
by Brainwy Software Ltda, EPL
liclipse Django Dart Mako coffescript ...

Installs: **60.7K** (4,855 last month)

FIGURE A.3: Search Eclipse IDE Marketplace

Type "Go" in search bar and press Go button to search for available plugin. Press install now to proceed with installation.

### A.4.3 Open Perspective

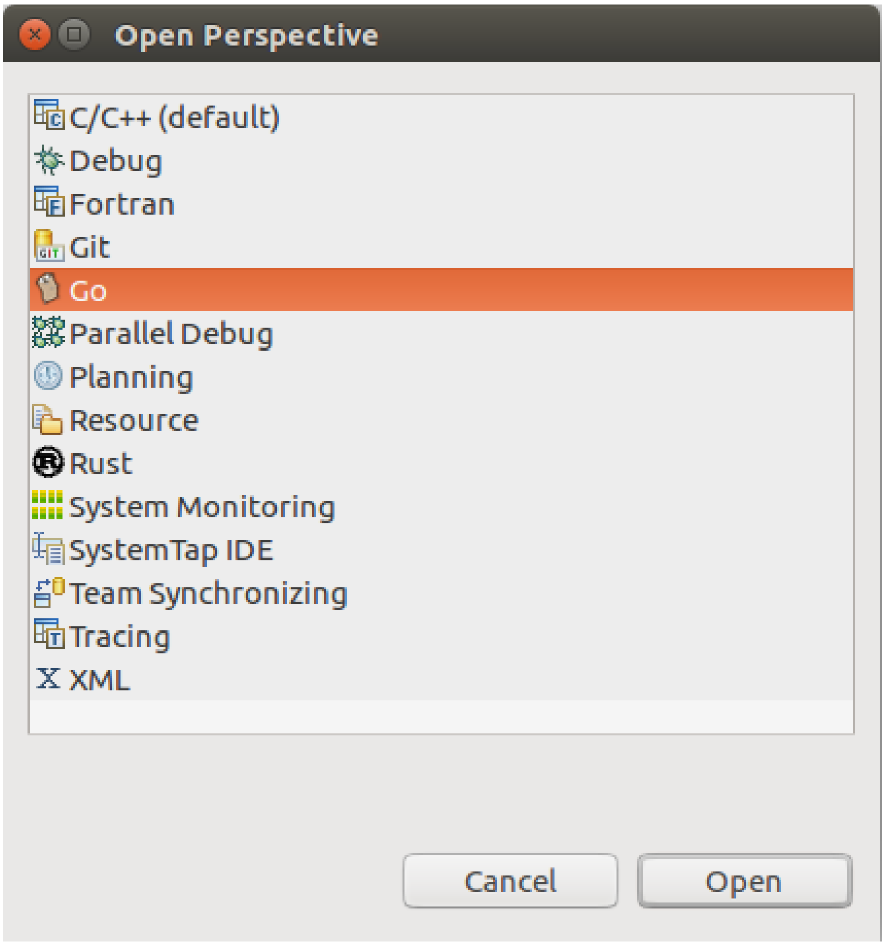


```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello, playground")
}
```

FIGURE A.4: Open Perspective

After the installation is done and success, open Eclipse Perspective by select Window, Perspective, Open Perspective and choose Other.

### A.4.4 Choose Perspective



Open Perspective

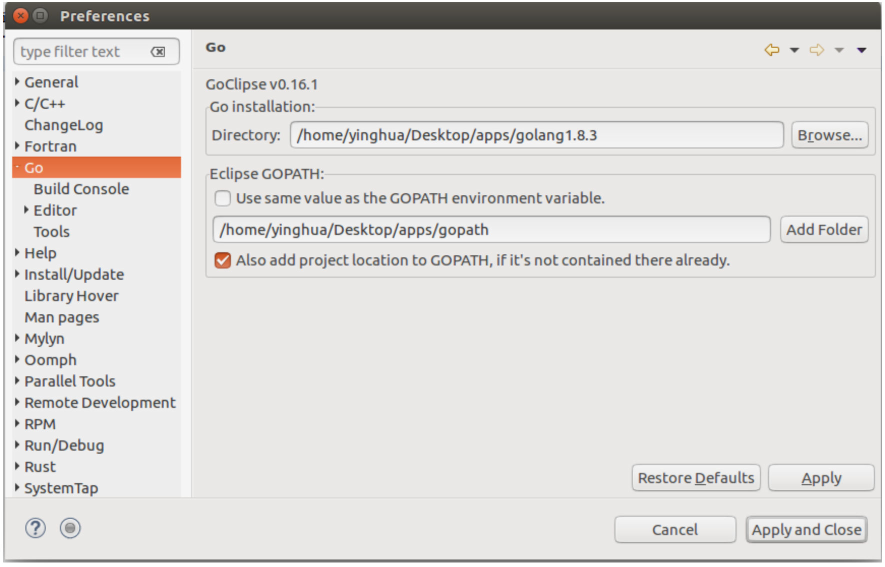
C/C++ (default)
Debug
Fortran
Git
Go
Parallel Debug
Planning
Resource
Rust
System Monitoring
SystemTap IDE
Team Synchronizing
Tracing
XML

FIGURE A.5: Choose Go Perspective

Choose Go Perspective and press Enter.

### A.4.5 Set Go compiler and GOPATH

goclipse-setting.png goclipse-setting.png



GoClipse v0.16.1

Go installation:

Directory: /home/yinghua/Desktop/apps/golang1.8.3

Eclipse GOPATH:

□ Use same value as the GOPATH environment variable.

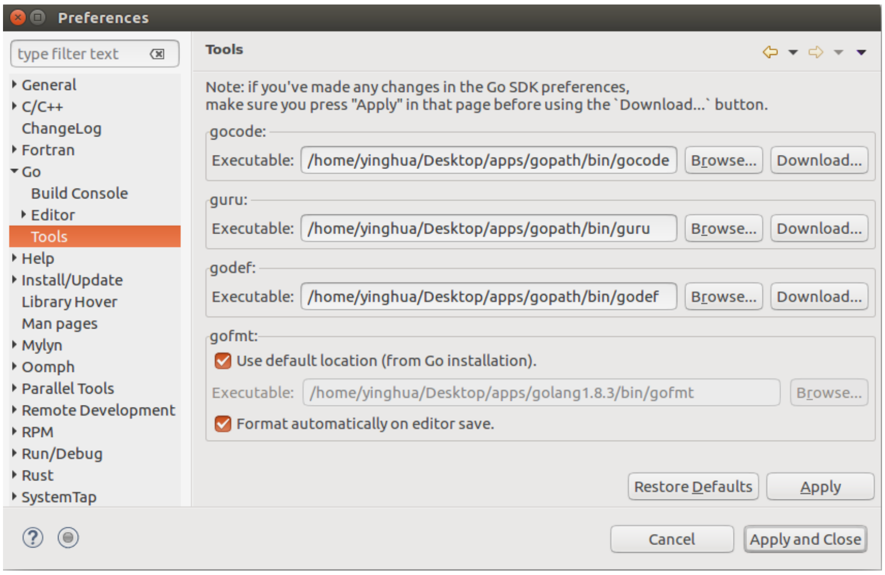
/home/yinghua/Desktop/apps/gopath

☑ Also add project location to GOPATH, if it's not contained there already.

Figure A.6: Set Go compiler and GOPATH

Set Go compiler and GOPATH into Goclipse plugins.

### A.4.6 Set GOCODE, GURU, GODEF and GOFMT path



Tools

Note: if you've made any changes in the Go SDK preferences,
make sure you press "Apply" in that page before using the `Download...` button.

gocode:
Executable: /home/yinghua/Desktop/apps/gopath/bin/gocode Browse... Download...

guru:
Executable: /home/yinghua/Desktop/apps/gopath/bin/guru Browse... Download...

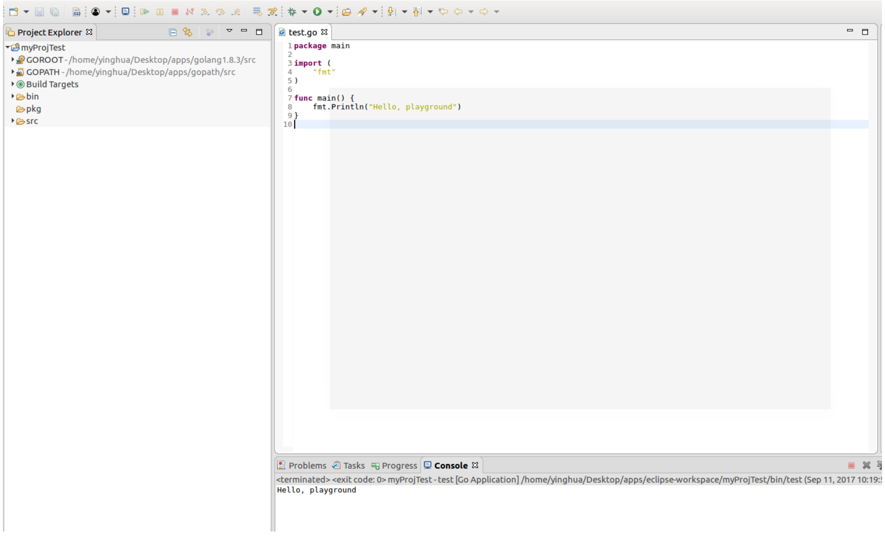
godef:
Executable: /home/yinghua/Desktop/apps/gopath/bin/godef Browse... Download...

gofmt:
☑ Use default location (from Go installation).
Executable: /home/yinghua/Desktop/apps/golang1.8.3/bin/gofmt Browse...
☑ Format automatically on editor save.

Figure A.7: Set GOCODE, GURU, GODEF and GOFMT path

Set GOCODE, GURU, GODEF and GOFMT executable path into Goclipse plugins and press "Apply and Close" to complete the setup process.

### A.4.7 Test Go compilation in Eclipse IDE


Project Explorer
myProjTest
GOROOT - /home/yinghua/Desktop/apps/golang1.8.3/src
GOPATH - /home/yinghua/Desktop/apps/gopath/src
Build Targets
bin
pkg
src
test.go
Problems
Tasks
Progress
Console

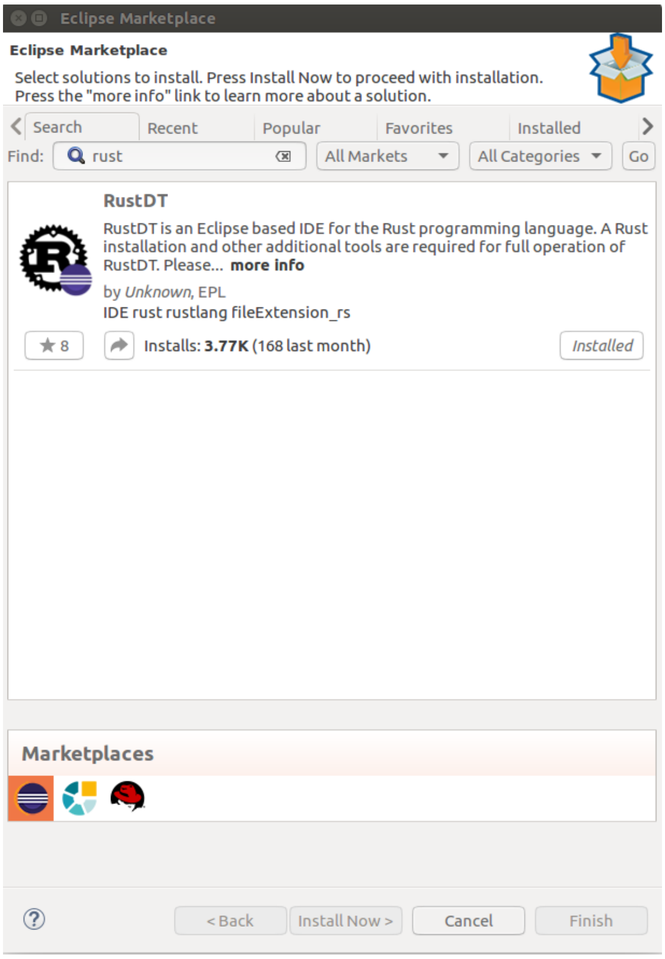
```go
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello, playground")
}
```

```
<terminated> <exit code: 0> myProjTest - test [Go Application] /home/yinghua/Desktop/apps/eclipse-workspace/myProjTest/bin/test (Sep 11, 2017 10:19:
Hello, playground
```

Figure A.8: Test Go compilation in Eclipse IDE

Test Go compilation with simple Hello Playground program, the setup process is successful if the Go program is compile and run correctly.

## A.5 RustDT plugin for Eclipse IDE installation



Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

RustDT

RustDT is an Eclipse based IDE for the Rust programming language. A Rust installation and other additional tools are required for full operation of RustDT. Please... **more info**

by *Unknown*, EPL

IDE rust rustlang fileExtension_rs

Installs: **3.77K** (168 last month)

FIGURE A.9: Test Go compilation in Eclipse IDE

Open Eclipse Marketplace similar to step in Appendix A.4.1 to A.4.7. Search the marketplace by type "Rust" in search bar and press Go button to search for tools. Press install now to proceed with installation. The setup process is similar with Goclipse installation process, once the installation and setup is done. The program will compile and run successfully.

## A.6 Linux command for PostgreSQL database installation

```

===========================================
Step 1 - Install postgreSQL in command line
===========================================

yinghua@yinghua-NL8C:~$ sudo apt-get update
yinghua@yinghua-NL8C:~$ sudo apt-get install postgresql postgresql-contrib
[sudo] password for yinghua:

===========================================
Step 2 - Create database for FYP1
===========================================
postgres=# create database fyp1;
CREATE DATABASE

postgres=# \l
List of databases
Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
fyp1      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
|          |          |             |             | postgres=CTc/postgres
template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
|          |          |             |             | postgres=CTc/postgres
(4 rows)

===========================================
Step 3 - Initial login with postgres user into psql
===========================================
yinghua@yinghua-NL8C:~$ sudo -i -u postgres psql
psql (9.5.7)
Type "help" for help.

======================================================
Step 4 - Add myself as new user for PostgreSQL with Superuser access
======================================================
yinghua@yinghua-NL8C:~/Documents/FYP/Postcode-data/uk-postcodes-master$ sudo -i -u postgres psql fyp1
[sudo] password for yinghua:
psql (9.5.7)
Type "help" for help.

postgres@yinghua-NL8C:~$ createuser -P -s -e yinghua
Enter password for new role:
Enter it again:
CREATE ROLE yinghua PASSWORD 'md5eec308d944ffa817c37ee6230b0c98eb' SUPERUSER CREATEDB CREATEROLE INHERIT
     LOGIN;

======================================================
Step 5 - List all the user in PostgreSQL
======================================================
postgres=# \du
List of roles
Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
yinghua   | Superuser, Create role, Create DB


===========================================
Step 6 - Connect FYP1 Database
===========================================
postgres=# \c fyp1
You are now connected to database "fyp1" as user "postgres".

===============================================
Step 7 - Check whether there are tables in FYP1 database
===============================================
fyp1=# \dt
No relations found.
```

Listing A.3: Linux command for PostgreSQL database installation

Install PostgreSQL database with command line using APT package. After the installation is success, create new user for new database in PostgreSQL.

# Appendix B

# Data Validation

## B.1 Introduction

The devil advocation test is conducted to ensure obtained raw CSV data are clean and useful. The test is conduced to ensure:-

1. The number of commas in each records should match number of columns in database.
2. Raw data from CSV should match the column's data type in database for data importation and preparation.
3. Review and check uniqueness of data in each columns and row.

## B.2 Match number of commas with database columns

```

===================
1. connect to database
===================

yinghua@yinghua:~$ psql fyp1;
psql (9.5.8)
Type "help" for help.

==================================
2. create table without one column
==================================

fyp1=# create table subject_test ( ukprn int not null, providername varchar(100) not null, region varchar
     (100) not null, subject varchar(50) not null, sex varchar(30) not null, yearaftergraduation varchar
     (30) not null, grads varchar(10) null default null, unmatched varchar(20) null default null, matched
     varchar(20) null default null, activityNotCaptured varchar(20) default null, nosustdest varchar(20)
     null default null, sustemponly varchar(20) null default null, sustemp varchar(20) null default null,
     sustempfsorboth varchar(20) null default null, earningsinclude varchar(20) null default null,
     lowerannearn varchar(20) null default null, medianannearn varchar(20) null default null, upperannearn
     varchar(20) null default null, polargrpone varchar(20) null default null, polargrponeincluded varchar
     (20) null default null, prattband varchar(20) null default null);

========================================================================
3. Terminal return error complains extra data after last expected column
========================================================================

fyp1=# \copy subject_test from 'institution-subject-data.csv' with header csv;

ERROR:  extra data after last expected column

CONTEXT:  COPY subject_test, line 2: "10000291,Anglia Ruskin University,East,Agriculture & related subjects,
     Female,1,30,x,x,x,x,x,x,x,20,9..."
```

Listing B.1: Match number of commas with database columns

In this section, PostgreSQL query is executed on a terminal to check the number of commas match the number of columns possesses in the table. We will purposely remove one column during table creation and try to import all rows of data into PostgreSQL database.

The terminal will return an error and complains data could not insert into the table because a column is expected during importation process.

```

========================================================================
4. Add new column into tables and import data successfully
========================================================================

fyp1=# alter table subject_test add column prattincluded varchar(20) null default null;
fyp1=# \copy subject_test from 'institution-subject-data.csv' with header csv;
COPY 32706
```

Listing B.2: Identify completeness in datasets.

Ultimately, the CSV raw data will import successfully only if the count of commas match the counts of columns in table.

## B.3 Identify correctness and suitability of data types

```

======================
Step 1. connect to database
======================

yinghua@yinghua:~$ psql fyp1;
psql (9.5.8)
Type "help" for help.

==================================
Step 2. create companydata table
==================================

fyp1=# create table companydata ( CompanyName varchar(160) null default null, CompanyNumber varchar(8) not
     null primary key, CareOf varchar(100) null, POBOX varchar(10) null, AddressLine1 varchar(300) null,
     AddressLine2 varchar(300) null, PostTown varchar(50) null, County varchar(50) null, Country varchar
     (50) null, PostCode varchar(20) null, CompanyCategory varchar(100) not null, CompanyStatus varchar(70)
      not null, CountryOfOrigin varchar(50) not null, DissolutionDate date null default null,
     IncorporationDate date null default null, AccountingRefDay int null, AccountingRefMonth int null
     default 0, Account_NextDueDate date null default null, Account_LastMadeUpdate date null default null,
     AccountCategory varchar(30) null, Return_NextDueDate date null default null, Return_LastMadeUpDate
     date null default null, NumMortChanges int null, NumMortOutstanding int null, NumMortPartSatisfied int
      null, NumMortSatisfied int null, SICCode1 varchar(170) null, SICCode2 varchar(170) null, SICCode3
     varchar(170) null, SICCode4 varchar(170) null, NumGenPartners int not null, NumLimPartners int not
     null, URI varchar(47) not null, pn1_CONDate date null default null, pn1_CompanyName varchar(160) null,
      pn2_CONDate date null default null, pn2_CompanyName varchar(160) null, pn3_CONDate date null default
     null, pn3_CompanyName varchar(160) null, pn4_CONDate date null default null, pn4_CompanyName varchar
     (160) null, pn5_CONDate date null default null, pn5_CompanyName varchar(160) null, pn6_CONDate date
     null default null, pn6_CompanyName varchar(160) null, pn7_CONDate date null default null,
     pn7_CompanyName varchar(160) null, pn8_CONDate date null default null, pn8_CompanyName varchar(160)
     null, pn9_CONDate date null default null, pn9_CompanyName varchar(160) null, pn10_CONDate date null
     default null, pn10_CompanyName varchar(160) null, ConfStmtNextDueDate date null default null,
     ConfStmtLastMadeUpDate date null default null);
CREATE TABLE

===========================================
Step 3 - Import data into companydata table
===========================================
fyp1=# \copy companydata from 'Basic-Company-Data-Full.csv' with header csv;

=================================================================================
Step 4 - Terminal return error because double quotes are not allow to insert into date datatypes.
=================================================================================
ERROR:  invalid input syntax for type date: ""
CONTEXT:  COPY companydata, line 2, column dissolutiondate: ""
```

Listing B.3: Identify correctness of data types

In this section, PostgreSQL query is executed on a terminal to check the suitability and correctness of data types during data importation from CSV files to PostgreSQL database.

The terminal will return an error and because double quotes are not allow to insert into "date" datatypes. It is caused by the NULL values in company CSV raw data is generated with double quotes and unable to insert them into "date" data types.

```

=====================================
Step 5 - Remove null value with double quotes for data insertion on DATE DATATYPE
=====================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ sed 's/""//g' Basic-Company-Data-Full.csv > Full.csv

=====================================
Step 6 - Import data into companydata table
=====================================
fyp1=# \copy companydata from 'Full.csv' with header csv;
COPY 4077979
```

Listing B.4: Remove null values with double quotes in CSV raw data

As the meaning of null values with double quotes and without quotes are the same. To resolve this problem, *seq* command is required produce new files by remove null values with double quotes stores in each columns. The CSV raw data will import successfully if every columns of data match table's column data types.

## B.4 Identify row and column uniqueness in each raw data

Data redundancy and duplication is an inevitable phenomenon found in million of data obtained from on-line sources. Unintentional duplication of records created from data warehouse are hardly avoided. Therefore, the uniqueness of data has to be check in every row and columns for conduct data de-duplication in Phase 2.

### B.4.1 Identify row uniqueness

```
=====================
Step 1. connect to database
=====================

yinghua@yinghua:~$ psql fyp1;
psql (9.5.8)
Type "help" for help.

fyp1#Data redundancy and duplication is an inevitable phenomenon found in million of data obtained from on-line sources. Unintentional duplication of records created from the data warehouse ’s hard to be avoided. Therefore, the uniqueness of data has to be check in every row and columns for conduct data de-duplication in Phase 2.

=====================================================
Step 2 - Verify duplicates row in company data tables
=====================================================
fyp1=# select (companydata.*)::text, count(*) from companydata group by companydata.* having count(*) > 1;

 companydata | count
-------------+-------
(0 rows)

=====================================================
Step 3 - Verify duplicates row in subject data tables
=====================================================
fyp1=# select (leo.*)::text, count(*) from leo group by leo.* having count(*) > 1;
  leo    | count
---------+-------
(0 rows)

=====================================================
Step 4 - Verify duplicates row in LEO data tables
=====================================================
fyp1=# select (leo.*)::text, count(*) from leo group by leo.* having count(*) > 1;
  leo    | count
---------+-------
(0 rows)

=====================================================
Step 5: Verify duplicates row in NSPL data table
=====================================================
fyp1=# select (nspl.*)::text, count(*) from nspl group by nspl.* having count(*) > 1;
 nspl | count
------+-------
(0 rows)
```

Listing B.5: Identify row uniqueness

In this section, PostgreSQL query is executed on a terminal to identify duplicates row found in every table. The result shows that there is no row duplication occurs between rows.

### B.4.2 Identify column uniqueness

```
=====================
Step 1. connect to database
=====================

yinghua@yinghua:~$ psql fyp1;
psql (9.5.8)
Type "help" for help.

==============================
Step 2. List structure of table
==============================

fyp1=# \d+ leo

Table "public.leo"
Column              |          Type          |            Modifiers            | Storage  |
--------------------+------------------------+---------------------------------+----------+
ukprn               | integer                | not null                        | plain    |
providername        | character varying(100) | not null                        | extended |
region              | character varying(100) | not null                        | extended |
subject             | character varying(50)  | not null                        | extended |
sex                 | character varying(30)  | not null                        | extended |
yearaftergraduation | character varying(30)  | not null                        | extended |
grads               | character varying(10)  | default NULL::character varying | extended |
unmatched           | character varying(20)  | default NULL::character varying | extended |

(more columns are not shown.....)

==============================================================
Step 3. Check duplication of data in selected columns
==============================================================

fyp1=# select ukprn, providername, region, count(*) from leo group by ukprn, providername, region having
     count(*) > 1;

==============================================================
Step 4. The duplication of columns with rows are return
==============================================================

ukprn    |                    providername                     |          region          | count
----------+-----------------------------------------------------+--------------------------+-------
10007775 | Queen Mary University of London                     | London                   |   207
10007792 | The University of Exeter                            | South West               |   207
10003324 | The Institute of Cancer Research                    | London                   |   207
10007784 | University College London                           | London                   |   207
10003957 | Liverpool John Moores University                    | North West               |   207
10000886 | The University of Brighton                          | South East               |   207
10007816 | The Royal Central School of Speech and Drama        | London                   |   207
10002681 | Glasgow School of Art                               | Scotland                 |   207
10005545 | Royal Agricultural University                       | South West               |   207
10037449 | University of St Mark and St John                   | South West               |   207
10007144 | The University of East London                       | London                   |   207
10007161 | Teesside University                                 | North East               |   207
10007713 | York St John University                             | Yorkshire and the Humber |   207
10003863 | Leeds Trinity University                            | Yorkshire and the Humber |   207

(more duplication data found in columns are not shown......)
```

Listing B.6: Identify column uniqueness

In this section, PostgreSQL query is executed on a terminal to identify duplicates data found in specific columns. The result shows the count of duplication data found in selected columns and lists out in tabular form. This method is proved to be able to identify data duplication occurs within a column.

# Appendix C

# Golang programming for import CSV into PostgreSQL database

## C.1 Introduction

The Go Programming Language possess package csv to reads and write comma-separated values (CSV) files. The package will automatically ignore whitespace, blank lines and delimits commas to read data. In addition, the language also contains a driver to perform CRUD transaction on PostgreSQL database.

The program below imports 100 rows of company data, LEO data and NSPL data from CSV files to PostgreSQL database. Five columns of data are selected from each file to import into this program as proof of concept in this project. The tables will be created in PostgreSQL database before the program is executed.

### C.1.1 LEO table for data importation

```

-- File: fyp1-leo.sql
-- Author: Chai Ying Hua
-- Database: psql (PostgreSQL) 9.5.8

-- ======================================
-- CHANGES IN V1.1(Sun Aug 27. 2017)
--      Create leo table for phase 1 to import data
-- ======================================

create table go_subject  (
        ukprn int not null,
        providername varchar(100) not null,
        region varchar(100) not null,
        subject varchar(50) not null,
        sex varchar(30) not null
);
```

Listing C.1: PostgreSQL query for LEO table creation.

### C.1.2 NSPL table for data importation

```

-- File: fyp1-nspl.sql
-- Author: Chai Ying Hua
-- Database: psql (PostgreSQL) 9.5.8

-- ======================================
-- CHANGES IN V1.1(Mon Sep 4. 2017)
--      Create nspl table for phase 1 to import data
-- ======================================

create table go_nspl (
        postcode1 varchar(15) not null,
        postcode2 varchar(15) not null primary key,
        date_introduce varchar(10) not null,
        usertype int not null,
        position_quality int not null
)
```

Listing C.2: PostgreSQL query for NSPL table creation.

### C.1.3 LEO table for data importation

```

-- File: fyp1-company.sql
-- Author: Chai Ying Hua
-- Database: psql (PostgreSQL) 9.5.8

-- =========================================================================
-- CHANGES IN V1.1(Sun Aug 27. 2017)
--      Create companydata table for phase 1 to import data
-- =========================================================================

create table go_company (
        CompanyName varchar(160) null default null,
        CompanyNumber varchar(8) not null primary key,
        CompanyCategory varchar(100) not null,
        CompanyStatus varchar(70) not null
        CountryOfOrigin varchar(50) not null
);
```

Listing C.3: PostgreSQL query for Company table creation.

### C.1.4 Source code of Go program

```

package main

import (
        "bufio"
        "database/sql"
        "encoding/csv"
        "fmt"
        "io"
        "os"
        "strconv"

        _ "github.com/lib/pq"
)

const (
        DB_USER                       = "yinghua"
        DB_PASSWORD                   = "123"
        DB_NAME                       = "fyp1"
        COMPANY_FILE_DIRECTORY string = "/home/yinghua/Documents/FYP-data/company-data/company-data-full.csv"
        LEO_FILE_DIRECTORY     string = "/home/yinghua/Documents/FYP-data/subject-data/institution-subject-data.csv"
        NSPL_FILE_DIRECTORY    string = "/home/yinghua/Documents/FYP-data/postcode-data/UK-NSPL.csv"
)

type CompanyData struct {
        name     string
        number   string
        category string
        status   string
        country  string
}

type LEOData struct {
        ukprn   int
        name    string
        region  string
        subject string
        sex     string
}

type NSPLData struct {
        postcode1      string
        postcode2      string
        date_introduce string
        usertype       int
        pos_quality    int
}

var db *sql.DB

//==================================================
//function to check error and print error messages
//==================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//==================================================
// initialize connection to database
//==================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        psqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "psql open")
        db = psqldb

}

//==================================================
// Import company data
//==================================================
func importCompanyData() {

        var sStmt string = "insert into go_company values ($1, $2, $3, $4, $5)"

        stmt, err := db.Prepare(sStmt)
        checkErr(err, "Prepare Stmt")

        // Open CSV files
        csvFile, err := os.Open(COMPANY_FILE_DIRECTORY)
        checkErr(err, "Open CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                record, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }

                company := CompanyData{
                        name:     record[0],
                        number:   record[1],
                        category: record[10],
                        status:   record[11],
                        country:  record[12],
                }

                stmt.Exec(company.name, company.number, company.category, company.status, company.country)
                checkErr(err, "Company Data importation")
        }
}

//====================================================
// Import LEO data
//====================================================
func importSubjectData() {

        var sStmt string = "insert into go_subject values ($1, $2, $3, $4, $5)"

        stmt, err := db.Prepare(sStmt)
        checkErr(err, "Prepare Subject Stmt")

        csvFile, err := os.Open(LEO_FILE_DIRECTORY)
        checkErr(err, "Open LEO CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                record, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }

                integer, err := strconv.Atoi(record[0])
                checkErr(err, "Convert UKRPN to Integer")

                subject := LEOData{
                        ukprn:   integer,
                        name:    record[1],
                        region:  record[2],
                        subject: record[3],
                        sex:     record[4],
                }

                        stmt.Exec(subject.ukprn, subject.name, subject.region, subject.subject, subject.sex)
                        checkErr(err, "Subject Data importation")
                }
}

//====================================================
// Import NSPL data
//====================================================
func importNSPLData() {

        var sStmt string = "insert into go_nspl values ($1, $2, $3, $4, $5)"

        stmt, err := db.Prepare(sStmt)
        checkErr(err, "Prepare Postcode Stmt")

        csvFile, err := os.Open(NSPL_FILE_DIRECTORY)
        checkErr(err, "Open Postcode CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                record, err := reader.Read()

                // skipped the first line
                if i == 0 {
                continue
                }

                // Stop at EOF.
                if err == io.EOF {
                break
                }

                userInt, err := strconv.Atoi(record[4])
                checkErr(err, "Convert Usertype to Integer")

                posInt, err := strconv.Atoi(record[7])
                checkErr(err, "Convert Usertype to Integer")

                postcode := NSPLData {
                postcode1:      record[0],
                postcode2:      record[1],
                date_introduce: record[3],
                usertype:       userInt,
                pos_quality:    posInt,
                }

                stmt.Exec(postcode.postcode1, postcode.postcode2, postcode.date_introduce, postcode.usertype, postcode.pos_quality)
                checkErr(err, "Postcode Data importation")
        }
}

func main() {

        initDB()
        importCompanyData()
        importSubjectData()
        importNSPLData()

}

/**

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build import-csv-psql.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run import-csv-psql.go

real    0m3.647s
user    0m0.328s
sys     0m0.096s
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$

**/
```

Listing C.4: Source code of Go program

# Appendix D

# Sequential and concurrent programming with Golang on PostgreSQL database retrieval.

## D.1 Golang Sequential Program Source Code

```

package main

        import (
        "database/sql"
        "fmt"
        "time"

        _ "github.com/lib/pq"
)

const (
        DB_USER     = "yinghua"
        DB_PASSWORD = "123"
        DB_NAME     = "fyp1"
)

var db *sql.DB

//=====================================================
//function to check error and print error messages
//=====================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//=====================================================
// initialize connection with database
//=====================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        psqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "Initialize database")
        db = psqldb

}

//=====================================================
// retrieve data from company table in postgres
//=====================================================
func retrieveCompanyData() {
```

```go

        fmt.Println("Start retrieve company data from database ... ")
        start := time.Now()

        time.Sleep(time.Second * 2)

        rows, err := db.Query("SELECT c.companyname, c.companynumber, c.companycategory, c.companystatus, c.countryoforigin FROM companydata AS c ORDER BY c.companynumber limit 100;")
        checkErr(err, "Query Company DB rows")

        var (
                companyname     string
                companynumber   string
                companycategory string
                companystatus   string
                countryoforigin string
        )

        for rows.Next() {
                err = rows.Scan(&companyname, &companynumber, &companycategory, &companystatus, &countryoforigin)
                checkErr(err, "Read company data rows")
                //fmt.Printf("%8v %3v %6v %6v %6v\n", companyname, companynumber, companycategory, companystatus, countryoforigin)
        }

        fmt.Println("Data retrieval of company data SUCCESS! ")
        fmt.Printf("%.8fs elapsed\n\n", time.Since(start).Seconds())

}

//==================================================
// retrieve data from postcode table in postgres
//==================================================
func retrievePostcodeData() {

        fmt.Println("Start retrieve postcode data from database ... ")
        start := time.Now()

        time.Sleep(time.Second * 2)

        rows, err := db.Query("SELECT postcode1, postcode2, date_introduce, usertype, position_quality FROM go_nspl LIMIT 50")
        checkErr(err, "Query Postcode DB rows")

        var (
                postcode1        string
                postcode2        string
                date_introduce   string
                usertype         int
                position_quality int
        )

        for rows.Next() {
                err = rows.Scan(&postcode1, &postcode2, &date_introduce, &usertype, &position_quality)
                checkErr(err, "Read postcode data rows")
                //fmt.Printf("%6v %8v %6v %6v %6v\n", postcode1, postcode2, date_introduce, usertype, position_quality)
        }

        fmt.Print("Data retrieval of postcode data SUCCESS! ")
        fmt.Printf("%.8fs elapsed\n\n", time.Since(start).Seconds())

}

//==================================================
// retrieve data from subject table in postgres
//==================================================
func retrieveSubjectData() {

        fmt.Println("Start retrieve LEO data from database ... ")
        start := time.Now()

        time.Sleep(time.Second * 2)

        rows, err := db.Query("SELECT ukprn, providername, region, subject, sex FROM go_subject LIMIT 50")
        checkErr(err, "Query subject DB rows")

        var (
                ukprn   int
                name    string
                region  string
                subject string
                sex     string
        )

        for rows.Next() {
                err = rows.Scan(&ukprn, &name, &region, &subject, &sex)
                checkErr(err, "Read subject data rows")
                //fmt.Printf("%6v %8v %6v %6v %6v\n", ukprn, name, region, subject, sex)
        }

        fmt.Print("Data retrieval of subject data SUCCESS! ")
        fmt.Printf(" %.8fs elapsed\n\n", time.Since(start).Seconds())

}

//==================================================
// Main function
//==================================================
func main() {

        // get the time before execution
        start := time.Now()

        initDB()
        retrieveCompanyData()
        retrievePostcodeData()
        retrieveSubjectData()

        // print the time after execution
        fmt.Printf("Total execution %.5fs elapsed\n", time.Since(start).Seconds())

}

/**

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build sequential-psql.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run sequential-psql.go
Start retrieve company data from database ...
Data retrieval of company data SUCCESS!
2.00721985s elapsed

Start retrieve postcode data from database ...
Data retrieval of postcode data SUCCESS!
2.00144933s elapsed

Start retrieve LEO data from database ...
Data retrieval of subject data SUCCESS!
2.00131415s elapsed

Total execution 6.01005s elapsed

real    0m6.252s
user    0m0.272s
sys             0m0.032s


**/
```

Listing D.1: Golang Sequential Program Source Code

### D.1.1 Golang Concurrent Program Source Code

```

package main

import (
        "database/sql"
        "fmt"
        "time"

        _ "github.com/lib/pq"
)

//==================================================
// database information
//==================================================
const (
        DB_USER     = "yinghua"
        DB_PASSWORD = "123"
        DB_NAME     = "fyp1"
)

var (
        db          *sql.DB
        numChannels int = 3
)

//==================================================
// function to check error and print error messages
//==================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//==================================================
// initialize connection with database
//==================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        psqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "Initialize database")
        db = psqldb

}

//==================================================
// retrieve company data store in postgres database
//==================================================
func retrieveCompanyData(ch_company chan string) {

        fmt.Println("Start retrieve company data from database ... ")
        start := time.Now()

        time.Sleep(time.Second * 2)

        rows, err := db.Query("SELECT c.companyname, c.companynumber, c.companycategory, c.companystatus, c.
    countryoforigin FROM companydata AS c ORDER BY c.companynumber limit 100;")
        checkErr(err, "Query Company DB rows")

        var (
                companyname     string
                companynumber   string
                companycategory string
                companystatus   string
                countryoforigin string
        )

        for rows.Next() {
                err = rows.Scan(&companyname, &companynumber, &companycategory, &companystatus, &
    countryoforigin)
                checkErr(err, "Read company data rows")
                //fmt.Printf("%8v %3v %6v %6v %6v\n", companyname, companynumber, companycategory,
    companystatus, countryoforigin)
        }

        fmt.Printf("%.8fs elapsed\n", time.Since(start).Seconds())
        ch_company <- "Retrieval of company data success. \n"
}

//==================================================
// retrieve postcode data store in postgres database
//==================================================
func retrievePostcodeData(ch_postcode chan string) {

        fmt.Println("Start retrieve postcode data from database ... ")
        start := time.Now()

        time.Sleep(time.Second * 2)

        rows, err := db.Query("SELECT postcode1, postcode2, date_introduce, usertype, position_quality FROM go_nspl LIMIT 50")
        checkErr(err, "Query Postcode DB rows")

        var (
                postcode1        string
                postcode2        string
                date_introduce   string
                usertype         int
                position_quality int
        )

        for rows.Next() {
                err = rows.Scan(&postcode1, &postcode2, &date_introduce, &usertype, &position_quality)
                checkErr(err, "Read postcode data rows")
                //fmt.Printf("%6v %8v %6v %6v %6v\n", postcode1, postcode2, date_introduce, usertype, position_quality)
        }

        fmt.Printf("%.8fs elapsed\n", time.Since(start).Seconds())
                ch_postcode <- "Retrieval of postcode success. \n"
        }

        //==================================================
        // retrieve subject data store in postgres database
        //==================================================
        func retrieveSubjectData(ch_subject chan string) {

                fmt.Println("Start retrieve LEO data from database ... ")
                start := time.Now()

                time.Sleep(time.Second * 2)

                rows, err := db.Query("SELECT ukprn, providername, region, subject, sex FROM go_subject LIMIT 50")
                checkErr(err, "Query subject DB rows")

                var (
                        ukprn   int
                        name    string
                        region  string
                        subject string
                        sex     string
                )

                for rows.Next() {
                        err = rows.Scan(&ukprn, &name, &region, &subject, &sex)
                        checkErr(err, "Read subject data rows")
                        //fmt.Printf("%6v %8v %6v %6v %6v\n", ukprn, name, region, subject, sex)
                }

                fmt.Printf("%.8fs elapsed\n", time.Since(start).Seconds())
                        ch_subject <- "Retrieval of subject data success. \n"
                }

                // select function
                func goSelect(ch_company, ch_subject, ch_postcode chan string) {

                for i := 0; i < numChannels; i++ {

                        select {
                        case msg1 := <-ch_postcode:
                                fmt.Println(msg1)
                        case msg2 := <-ch_company:
                                fmt.Println(msg2)
                        case msg3 := <-ch_subject:
                                fmt.Println(msg3)

                }

        }
}

//==================================================
// Main function
//==================================================
func main() {

        // make three channel for three functions
        ch_company := make(chan string)
        ch_subject := make(chan string)
        ch_postcode := make(chan string)

        // get the time before execution
        start := time.Now()

        initDB()

        //go routines
        go retrieveCompanyData(ch_company)
        go retrieveSubjectData(ch_subject)
        go retrievePostcodeData(ch_postcode)

        goSelect(ch_company, ch_subject, ch_postcode)

        // obtain the time after execution
        fmt.Printf("Total execution %.5fs elapsed\n", time.Since(start).Seconds())

}

/**

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build concurrent-psql.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run concurrent-psql.go
Start retrieve postcode data from database ...
Start retrieve company data from database ...
Start retrieve LEO data from database ...
2.00615007s elapsed
Retrieval of subject data success.

2.00661550s elapsed
Retrieval of postcode success.

2.00745319s elapsed
Retrieval of company data success.

Total execution 2.00754s elapsed

real    0m2.268s
user    0m0.244s
sys             0m0.076s



**/

)
```

Listing D.2: Golang Concurrent Program Source Code

# Appendix E

# Sequential and concurrent programming with Golang on reading CSV file

## E.1 Golang Sequential Program Source Code

```

package main

import (
        "bufio"
        "database/sql"
        "encoding/csv"
        "fmt"
        "io"
        "os"
        "time"

        _ "github.com/lib/pq"
)

const (
        DB_USER                       = "yinghua"
        DB_PASSWORD                   = "123"
        DB_NAME                       = "fyp1"
        COMPANY_FILE_DIRECTORY string = "/home/yinghua/Documents/FYP-data/company-data/company-data-full.csv
     "
        LEO_FILE_DIRECTORY     string = "/home/yinghua/Documents/FYP-data/subject-data/institution-subject-
     data.csv"
        NSPL_FILE_DIRECTORY    string = "/home/yinghua/Documents/FYP-data/postcode-data/UK-NSPL.csv"
)

var db *sql.DB

// function to check error and print error messages
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

func read_CompanyCSV() {

        fmt.Println("Start reading 100 row Company CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(COMPANY_FILE_DIRECTORY)
        checkErr(err, "Open CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }

        }

        fmt.Println("Finish reading Company CSV data")

}

func read_LEOCSV() {

        fmt.Println("Start reading 100 row LEO CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(LEO_FILE_DIRECTORY)
        checkErr(err, "Open LEO CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        fmt.Println("Finish readying LEO CSV data")

}

func read_NSPLCSV() {

        fmt.Println("Start reading 100 row NSPL CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(NSPL_FILE_DIRECTORY)
        checkErr(err, "Open Postcode CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

        // skipped the first line
                if i == 0 {
                        continue
                }

        // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        fmt.Println("Finish readying LEO CSV data")

}

func main() {

        // get the time before execution
        start := time.Now()

        read_CompanyCSV()
        read_LEOCSV()
        read_NSPLCSV()

        // obtain the time after execution
        fmt.Printf("Total execution %.5fs elapsed\n", time.Since(start).Seconds())

}

/**

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build sequential-read-csv.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run sequential-read-csv.go
Start reading 100 row Company CSV data
Finish reading Company CSV data
Start reading 100 row LEO CSV data
Finish readying LEO CSV data
Start reading 100 row NSPL CSV data
Finish readying LEO CSV data
Total execution 6.00823s elapsed

real    0m6.285s
user    0m0.316s
sys             0m0.056s
**/
```

Listing E.1: Golang Sequential Program Source Code

### E.1.1 Golang Concurrent Program Source Code

```

package main

import (
        "bufio"
        "database/sql"
        "encoding/csv"
        "fmt"
        "io"
        "os"
        "time"

        _ "github.com/lib/pq"
)

const (
        DB_USER                        = "yinghua"
        DB_PASSWORD                    = "123"
        DB_NAME                        = "fyp1"
        COMPANY_FILE_DIRECTORY string = "/home/yinghua/Documents/FYP-data/company-data/company-data-full.csv"
        LEO_FILE_DIRECTORY     string = "/home/yinghua/Documents/FYP-data/subject-data/institution-subject-data.csv"
        NSPL_FILE_DIRECTORY    string = "/home/yinghua/Documents/FYP-data/postcode-data/UK-NSPL.csv"
)

var (
        db          *sql.DB
        numChannels int = 3
)

// function to check error and print error messages
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

func read_CompanyCSV(ch_company chan string) {

        fmt.Println("Start reading 100 row Company CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(COMPANY_FILE_DIRECTORY)
        checkErr(err, "Open CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        ch_company <- "Finish readying LEO CSV data"

}

func read_LEOCSV(ch_leo chan string) {

        fmt.Println("Start reading 100 row LEO CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(LEO_FILE_DIRECTORY)
        checkErr(err, "Open LEO CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        ch_leo <- "Finish reading LEO CSV data"

}

func read_NSPLCSV(ch_nspl chan string) {

        fmt.Println("Start reading 100 row NSPL CSV data")

        time.Sleep(time.Second * 2)

        csvFile, err := os.Open(NSPL_FILE_DIRECTORY)
        checkErr(err, "Open Postcode CSV")

        defer csvFile.Close()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for i := 0; i <= 100; i++ {
                _, err := reader.Read()

                // skipped the first line
                if i == 0 {
                        continue
                }

                // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        ch_nspl <- "Finish reading NSPL CSV data"
}

// select function
func goSelect(ch_company, ch_leo, ch_nspl chan string) {

        for i := 0; i < numChannels; i++ {

                select {
                        case msg1 := <-ch_leo:
                                fmt.Println(msg1)
                        case msg2 := <-ch_company:
                        fmt.Println(msg2)
                                case msg3 := <-ch_nspl:
                        fmt.Println(msg3)

                }

        }
}

func main() {

        // make three channel for three functions
        ch_company := make(chan string)
        ch_leo := make(chan string)
        ch_nspl := make(chan string)

        // get the time before execution
        start := time.Now()

        go read_CompanyCSV(ch_company)
        go read_LEOCSV(ch_leo)
        go read_NSPLCSV(ch_nspl)

        goSelect(ch_company, ch_leo, ch_nspl)

        // obtain the time after execution
        fmt.Printf("Total execution %.5fs elapsed\n", time.Since(start).Seconds())

}

/**

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build concurrent-read-csv.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run concurrent-read-csv.go
Start reading 100 row NSPL CSV data
Start reading 100 row Company CSV data
Start reading 100 row LEO CSV data
Finish reading LEO CSV data
Finish reading NSPL CSV data
Finish readying LEO CSV data
Total execution 2.00376s elapsed

real    0m2.243s
user    0m0.264s
sys             0m0.044s

**/
```

Listing E.2: Golang Concurrent Program Source Code

# Appendix F

# Result of Sequential and concurrent programming with Golang on process CSV

## F.1 Linux command for Go program execution

```
==============================================================
Step 1 - Build sequential-read-csv.go
==============================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build sequential-read-csv.go

==============================================================
Step 2 - Execute sequential-read-csv.go program
==============================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run sequential-read-csv.go
Start reading 100 row Company CSV data
Finish reading Company CSV data
Start reading 100 row LEO CSV data
Finish readying LEO CSV data
Start reading 100 row NSPL CSV data
Finish readying LEO CSV data
Total execution 6.00823s elapsed

real    0m6.285s
user    0m0.316s
sys     0m0.056s

==============================================================
Step 3 - Build concurrent-read-csv.go
==============================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build concurrent-read-csv.go

==============================================================
Step 4 - Execute concurrent-read-csv.go program
==============================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run concurrent-read-csv.go
Start reading 100 row NSPL CSV data
Start reading 100 row Company CSV data
Start reading 100 row LEO CSV data
Finish reading LEO CSV data
Finish reading NSPL CSV data
Finish readying LEO CSV data
Total execution 2.00376s elapsed

real    0m2.243s
user    0m0.264s
sys     0m0.044s
```

Listing F.1: Linux command for Go program execution

## F.2 Result of Golang programming on process CSV

| **Elapsed Time** | **sequential-read-csv.go** | **concurrent-read-csv.go** |
|---|---|---|
| real | 6.285s | 2.243s |
| user | 0.316s | 0.264s |
| sys | 0.056s | 0.044s |

Table F.1: Result of Golang programming on process CSV raw data

# Appendix G

# Result of Sequential and concurrent programming with Golang on process PostgreSQL database.

## G.1 Linux command for Go program execution

```

================================================================
Step 1 - Build sequential-psql.go
================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build sequential-psql.go

================================================================
Step 2 - Execute sequential-psql.go program
================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run sequential-psql.go
Start retrieve company data from database ...
Data retrieval of company data SUCCESS!
2.00721985s elapsed

Start retrieve postcode data from database ...
Data retrieval of postcode data SUCCESS!
2.00144933s elapsed

Start retrieve LEO data from database ...
Data retrieval of subject data SUCCESS!
2.00131415s elapsed

Total execution 6.01005s elapsed

real    0m6.252s
user    0m0.272s
sys     0m0.032s

================================================================
Step 3 - Build concurrent-psql.go
================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build concurrent-psql.go

================================================================
Step 4 - Execute concurrent-psql.go program
================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run concurrent-psql.go
Start retrieve postcode data from database ...
Start retrieve company data from database ...
Start retrieve LEO data from database ...
```

```
2.00615007s elapsed
Retrieval of subject data success.

2.00661550s elapsed
Retrieval of postcode success.

2.00745319s elapsed
Retrieval of company data success.

Total execution 2.00754s elapsed

real    0m2.268s
user    0m0.244s
sys     0m0.076s
```

Listing G.1: Linux command for Go program execution

## G.2 Result of Golang programming on process PostgreSQL database

| **Elapsed Time** | **sequential-psql.go** | **concurrent-psql.go** |
|---|---|---|
| real | 6.252s | 2.268s |
| user | 0.272s | 0.244s |
| sys | 0.032s | 0.076s |

Table G.1: Result of Golang programming on PostgreSQL database

# Appendix H

# Result of import data from CSV file to PostgreSQL database with Golang

## H.1 Linux command for import data

```
=====================================
Step 1 - Connect to FYP1 database
=====================================

yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ psql fyp1;
psql (9.5.8)
Type "help" for help.

fyp1=#

=====================================
Step 2 - Check number of tables
=====================================
fyp1=# \d
List of relations
Schema |    Name     | Type  |  Owner
--------+-------------+-------+---------
public | companydata | table | yinghua
public | leo         | table | yinghua
public | nspl        | table | yinghua
(3 rows)

==================================================================
Step 3 - Create go_company table ready for importation
==================================================================
fyp1=# create table go_company (companyname varchar(160) null default null, companynumber varchar(8) not null primary key, companycategory varchar(100) not null, companystatus varchar(70) not null, countryoforigin varchar(50) not null );
CREATE TABLE

==================================================================
Step 4 - Create go_subject table ready for importation
==================================================================
fyp1=# create table go_subject (ukprn int not null, providername varchar(100) not null, region varchar(100) not null, subject varchar(50) not null, sex varchar(30) not null );
CREATE TABLE

==================================================================
Step 5 - Create go_nspl table ready for importation
==================================================================
fyp1=# create table go_nspl (postcode1 varchar(15) not null, postcode2 varchar(15) not null primary key, date_introduce varchar(10) not null,usertype int not null, position_quality int not null);

==================================================================
Step 6 - Check number of data in each respective table
```

```
==================================================================
fyp1=# \d
List of relations
Schema |    Name     | Type  |  Owner
--------+-------------+-------+---------
public | companydata | table | yinghua
public | go_company  | table | yinghua
public | go_nspl     | table | yinghua
public | go_subject  | table | yinghua
public | leo         | table | yinghua
public | nspl        | table | yinghua
(6 rows)

fyp1=# select count(*) from go_company;
count
-------
0
(1 row)

fyp1=# select count(*) from go_nspl;
count
-------
0
(1 row)

fyp1=# select count(*) from go_subject;
count
-------
0
(1 row)

==================================================================
Step 7 - List all the Go files
==================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ ls -l
total 33084
-rwxrwxr-x 1 yinghua yinghua 4903560 Sep 16 23:10 concurrent-psql
-rw-rw-r-- 1 yinghua yinghua    5487 Sep 17 23:25 concurrent-psql.go
-rwxrwxr-x 1 yinghua yinghua 4724204 Sep 16 23:13 concurrent-read-csv
-rw-rw-r-- 1 yinghua yinghua    3571 Sep 16 23:13 concurrent-read-csv.go
-rwxrwxr-x 1 yinghua yinghua 4858407 Sep 17 23:01 import-csv-psql
-rw-rw-r-- 1 yinghua yinghua    5146 Sep 17 23:02 import-csv-psql.go
-rwxrwxr-x 1 yinghua yinghua 4895323 Sep 16 23:09 sequential-psql
-rw-rw-r-- 1 yinghua yinghua    4728 Sep 17 23:20 sequential-psql.go
-rwxrwxr-x 1 yinghua yinghua 4720029 Sep 16 23:12 sequential-read-csv
-rw-rw-r-- 1 yinghua yinghua    3002 Sep 16 23:12 sequential-read-csv.go

===============================================================================
Step 8 - Build and run import-csv-psql.go to import data from CSV to PostgreSQL
===============================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ go build import-csv-psql.go
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ time go run import-csv-psql.go

real    0m3.622s
user    0m0.312s
sys     0m0.088s

===============================================================================
Step 9 - Connect to database and verified whether the importation is success
===============================================================================
yinghua@yinghua:~/Desktop/apps/eclipse-workspace/FYP1/src/postgres-process$ psql fyp1;
psql (9.5.8)
Type "help" for help.

fyp1=# select count(*) from go_company;
count
-------
100
(1 row)

fyp1=# select count(*) from go_nspl;
count
-------
100
(1 row)

fyp1=# select count(*) from go_subject;
count
-------
100
(1 row)
```

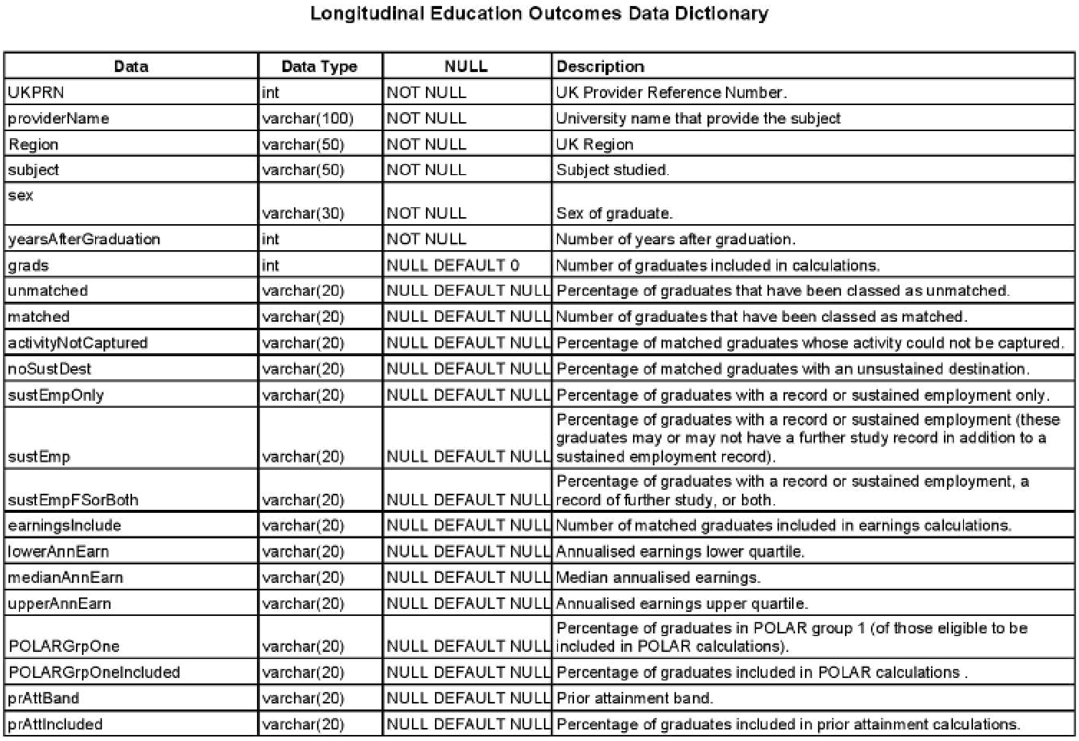
Listing H.1: Linux command for import data

# Appendix I

# Data Collection

## I.1 Data Dictionary of Raw Datasets

### I.1.1 Phase 1 Longitudinal Education Outcomes (LEO) Data Dictionary

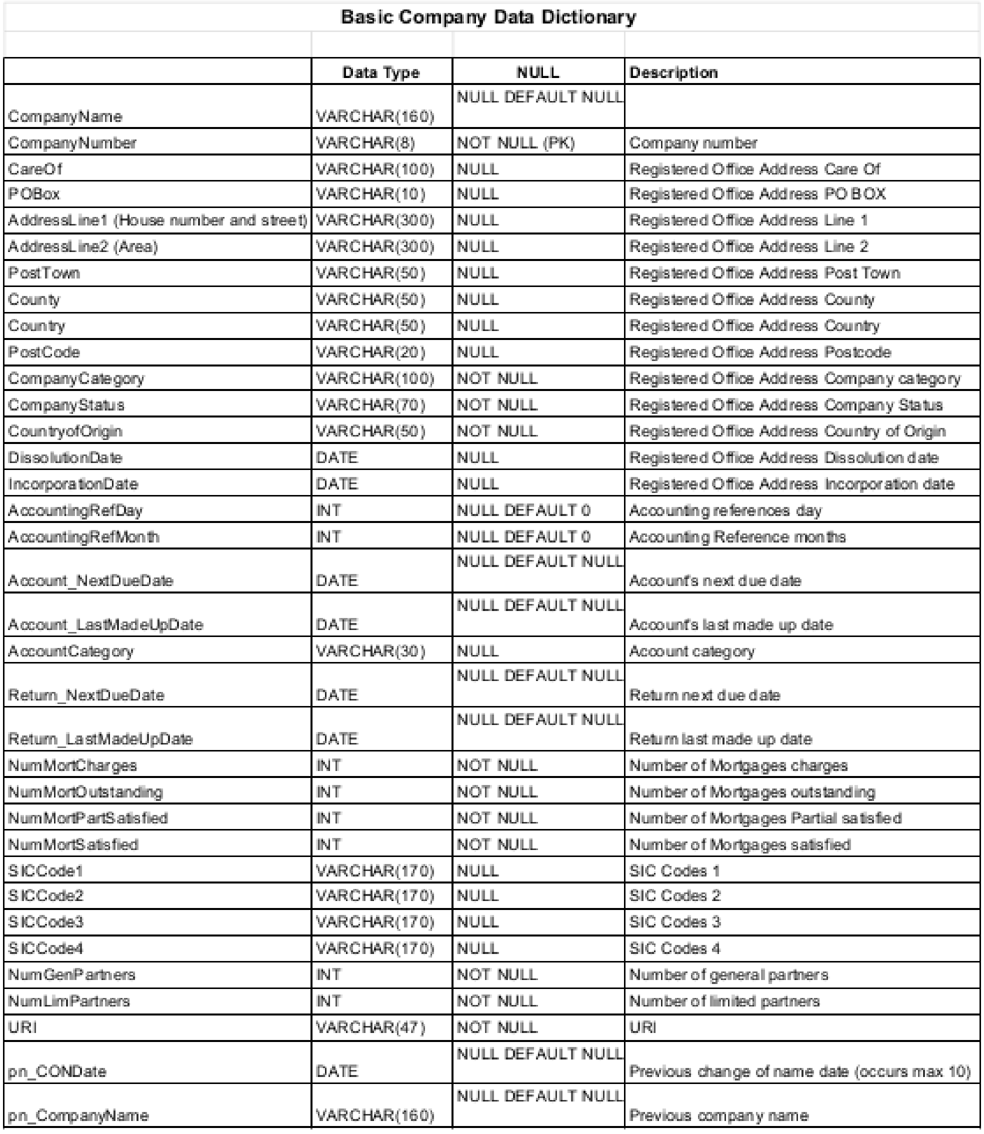


**Longitudinal Education Outcomes Data Dictionary**

| Data | Data Type | NULL | Description |
|---|---|---|---|
| UKPRN | int | NOT NULL | UK Provider Reference Number. |
| providerName | varchar(100) | NOT NULL | University name that provide the subject |
| Region | varchar(50) | NOT NULL | UK Region |
| subject | varchar(50) | NOT NULL | Subject studied. |
| sex | varchar(30) | NOT NULL | Sex of graduate. |
| yearsAfterGraduation | int | NOT NULL | Number of years after graduation. |
| grads | int | NULL DEFAULT 0 | Number of graduates included in calculations. |
| unmatched | varchar(20) | NULL DEFAULT NULL | Percentage of graduates that have been classed as unmatched. |
| matched | varchar(20) | NULL DEFAULT NULL | Number of graduates that have been classed as matched. |
| activityNotCaptured | varchar(20) | NULL DEFAULT NULL | Percentage of matched graduates whose activity could not be captured. |
| noSustDest | varchar(20) | NULL DEFAULT NULL | Percentage of matched graduates with an unsustained destination. |
| sustEmpOnly | varchar(20) | NULL DEFAULT NULL | Percentage of graduates with a record or sustained employment only. |
| sustEmp | varchar(20) | NULL DEFAULT NULL | Percentage of graduates with a record or sustained employment (these graduates may or may not have a further study record in addition to a sustained employment record). |
| sustEmpFSorBoth | varchar(20) | NULL DEFAULT NULL | Percentage of graduates with a record or sustained employment, a record of further study, or both. |
| earningsInclude | varchar(20) | NULL DEFAULT NULL | Number of matched graduates included in earnings calculations. |
| lowerAnnEarn | varchar(20) | NULL DEFAULT NULL | Annualised earnings lower quartile. |
| medianAnnEarn | varchar(20) | NULL DEFAULT NULL | Median annualised earnings. |
| upperAnnEarn | varchar(20) | NULL DEFAULT NULL | Annualised earnings upper quartile. |
| POLARGrpOne | varchar(20) | NULL DEFAULT NULL | Percentage of graduates in POLAR group 1 (of those eligible to be included in POLAR calculations). |
| POLARGrpOneIncluded | varchar(20) | NULL DEFAULT NULL | Percentage of graduates included in POLAR calculations . |
| prAttBand | varchar(20) | NULL DEFAULT NULL | Prior attainment band. |
| prAttIncluded | varchar(20) | NULL DEFAULT NULL | Percentage of graduates included in prior attainment calculations. |

Figure I.1: Phase 1 Longitudinal Education Outcomes (LEO) Data Dictionary

### I.1.2 Phase 1 Company Data Dictionary

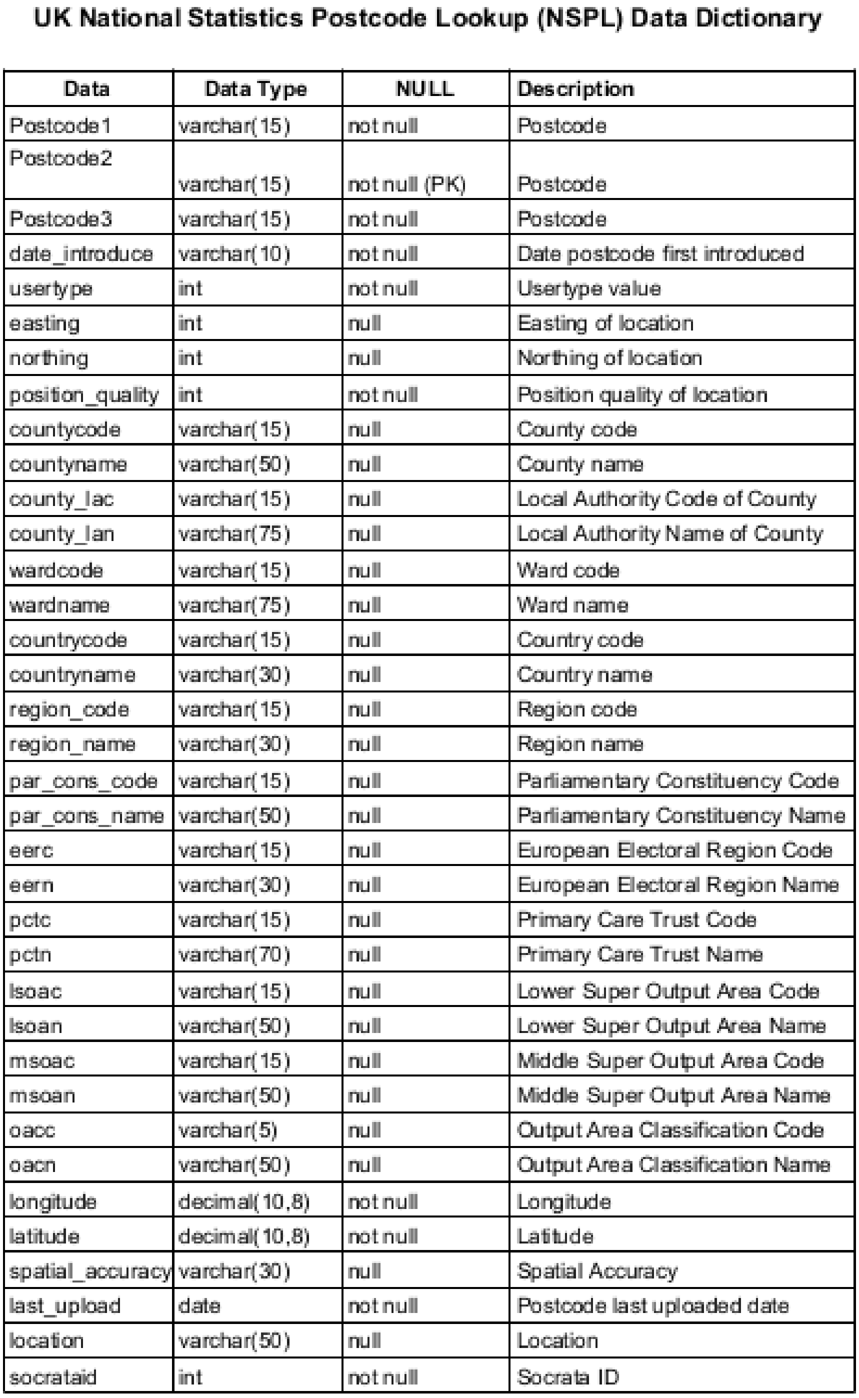


**Basic Company Data Dictionary**

| | Data Type | NULL | Description |
|---|---|---|---|
| CompanyName | VARCHAR(160) | NULL DEFAULT NULL | |
| CompanyNumber | VARCHAR(8) | NOT NULL (PK) | Company number |
| CareOf | VARCHAR(100) | NULL | Registered Office Address Care Of |
| POBox | VARCHAR(10) | NULL | Registered Office Address PO BOX |
| AddressLine1 (House number and street) | VARCHAR(300) | NULL | Registered Office Address Line 1 |
| AddressLine2 (Area) | VARCHAR(300) | NULL | Registered Office Address Line 2 |
| PostTown | VARCHAR(50) | NULL | Registered Office Address Post Town |
| County | VARCHAR(50) | NULL | Registered Office Address County |
| Country | VARCHAR(50) | NULL | Registered Office Address Country |
| PostCode | VARCHAR(20) | NULL | Registered Office Address Postcode |
| CompanyCategory | VARCHAR(100) | NOT NULL | Registered Office Address Company category |
| CompanyStatus | VARCHAR(70) | NOT NULL | Registered Office Address Company Status |
| CountryofOrigin | VARCHAR(50) | NOT NULL | Registered Office Address Country of Origin |
| DissolutionDate | DATE | NULL | Registered Office Address Dissolution date |
| IncorporationDate | DATE | NULL | Registered Office Address Incorporation date |
| AccountingRefDay | INT | NULL DEFAULT 0 | Accounting references day |
| AccountingRefMonth | INT | NULL DEFAULT 0 | Accounting Reference months |
| Account_NextDueDate | DATE | NULL DEFAULT NULL | Account's next due date |
| Account_LastMadeUpDate | DATE | NULL DEFAULT NULL | Account's last made up date |
| AccountCategory | VARCHAR(30) | NULL | Account category |
| Return_NextDueDate | DATE | NULL DEFAULT NULL | Return next due date |
| Return_LastMadeUpDate | DATE | NULL DEFAULT NULL | Return last made up date |
| NumMortCharges | INT | NOT NULL | Number of Mortgages charges |
| NumMortOutstanding | INT | NOT NULL | Number of Mortgages outstanding |
| NumMortPartSatisfied | INT | NOT NULL | Number of Mortgages Partial satisfied |
| NumMortSatisfied | INT | NOT NULL | Number of Mortgages satisfied |
| SICCode1 | VARCHAR(170) | NULL | SIC Codes 1 |
| SICCode2 | VARCHAR(170) | NULL | SIC Codes 2 |
| SICCode3 | VARCHAR(170) | NULL | SIC Codes 3 |
| SICCode4 | VARCHAR(170) | NULL | SIC Codes 4 |
| NumGenPartners | INT | NOT NULL | Number of general partners |
| NumLimPartners | INT | NOT NULL | Number of limited partners |
| URI | VARCHAR(47) | NOT NULL | URI |
| pn_CONDate | DATE | NULL DEFAULT NULL | Previous change of name date (occurs max 10) |
| pn_CompanyName | VARCHAR(160) | NULL DEFAULT NULL | Previous company name |

Figure I.2: Phase 1 Company Data Dictionary

### I.1.3 Phase 1 National Statistics Postcode Lookup (NSPL) Data Dictionary



**UK National Statistics Postcode Lookup (NSPL) Data Dictionary**

| Data | Data Type | NULL | Description |
|---|---|---|---|
| Postcode1 | varchar(15) | not null | Postcode |
| Postcode2 | varchar(15) | not null (PK) | Postcode |
| Postcode3 | varchar(15) | not null | Postcode |
| date_introduce | varchar(10) | not null | Date postcode first introduced |
| usertype | int | not null | Usertype value |
| easting | int | null | Easting of location |
| northing | int | null | Northing of location |
| position_quality | int | not null | Position quality of location |
| countycode | varchar(15) | null | County code |
| countyname | varchar(50) | null | County name |
| county_lac | varchar(15) | null | Local Authority Code of County |
| county_lan | varchar(75) | null | Local Authority Name of County |
| wardcode | varchar(15) | null | Ward code |
| wardname | varchar(75) | null | Ward name |
| countrycode | varchar(15) | null | Country code |
| countryname | varchar(30) | null | Country name |
| region_code | varchar(15) | null | Region code |
| region_name | varchar(30) | null | Region name |
| par_cons_code | varchar(15) | null | Parliamentary Constituency Code |
| par_cons_name | varchar(50) | null | Parliamentary Constituency Name |
| eerc | varchar(15) | null | European Electoral Region Code |
| eern | varchar(30) | null | European Electoral Region Name |
| pctc | varchar(15) | null | Primary Care Trust Code |
| pctn | varchar(70) | null | Primary Care Trust Name |
| lsoac | varchar(15) | null | Lower Super Output Area Code |
| lsoan | varchar(50) | null | Lower Super Output Area Name |
| msoac | varchar(15) | null | Middle Super Output Area Code |
| msoan | varchar(50) | null | Middle Super Output Area Name |
| oacc | varchar(5) | null | Output Area Classification Code |
| oacn | varchar(50) | null | Output Area Classification Name |
| longitude | decimal(10,8) | not null | Longitude |
| latitude | decimal(10,8) | not null | Latitude |
| spatial_accuracy | varchar(30) | null | Spatial Accuracy |
| last_upload | date | not null | Postcode last uploaded date |
| location | varchar(50) | null | Location |
| socrataid | int | not null | Socrata ID |

Figure I.3: Phase 1 National Statistics Postcode Lookup (NSPL) Data Dictionary

# Appendix J

# Data Encoding

## J.1 Dirty Records Found in Company Datasets.

```
=========================================================================
List first three rows of data in company-data.csv file for display purposes
=========================================================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ head -3 company-data.csv

"! LTD","08209948","","","METROHOUSE 57 PEPPER ROAD","HUNSLET","LEEDS","YORKSHIRE","","LS10 2RU","Private
     Limited Company","Active","United Kingdom","","11/09/2012","30","9","30/06/2018","30/09/2016","DORMANT
     ","09/10/2016","11/09/2015","0","0","0","0","99999 - Dormant Company","","","","0","0","http://
     business.data.gov.uk/id/company
     /08209948","","","","","","","","","","","","","","","","","","","","25/09/2019","11/09/2016"

"!BIG IMPACT GRAPHICS LIMITED","07382019","","","335 ROSDEN HOUSE","372 OLD STREET","LONDON","","","EC1V 9AV
     ","Private Limited Company","Active","United Kingdom
     ","","21/09/2010","30","9","30/06/2018","30/09/2016","DORMANT
     ","19/10/2016","21/09/2015","0","0","0","0","59112 - Video production activities","59113 - Television
     programme production activities","74100 - specialised design activities","74202 - Other specialist
     photography","0","0","http://business.data.gov.uk/id/company
     /07382019","","","","","","","","","","","","","","","","","","","","05/10/2019","21/09/2016"

"!NSPIRED LTD","SC421617","","","26 POLMUIR ROAD","","ABERDEEN","","UNITED KINGDOM","AB11 7SY","Private
     Limited Company","Active","United Kingdom","","11/04/2012","30","3","30/12/2017","30/03/2016","TOTAL
     EXEMPTION SMALL","09/05/2017","11/04/2016","0","0","0","0","70229 - Management consultancy activities
     other than financial management","","","","0","0","http://business.data.gov.uk/id/company/SC421617
     ","","","","","","","","","","","","","","","","","","","","","25/04/2020","11/04/2017"
```

Listing J.1: Three rows of data in Company CSV datasets

Listing J.1 display first three rows of data found in company CSV datasets, the double quotes are found in empty values (,””,””,””,) will result in storing as *String* into PostgreSQL database and caused data inconsistency. Therefore, Data encoding is performed to eliminate double quotes (””) found in empty values to prevent incompatible data types for data handling.

## J.2 Data encoding with stream editor.

```
=======================================
Step 1 - Date on running data encoding
=======================================
yinghua@yinghua-NL8C:~$ date
Sun Aug 27 01:33:00 MYT 2017

===========================================================
Step 2 - The specification of Operating System environment
===========================================================
yinghua@yinghua-NL8C:~$ uname -a
Linux yinghua-NL8C 4.10.0-32-generic #36~16.04.1-Ubuntu SMP Wed Aug 9 09:19:02 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux

===============================================
Step 3 - Change Directory to CSV file location
===============================================
yinghua@yinghua-NL8C:~$ cd ~/Documents/FYP/Basic-Company-Data/

===============================================
Step 4 - List files in directory
===============================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ ls -al
drwxrwxr-x 5 yinghua yinghua       4096 Feb  7 15:09 .
drwxrwxr-x 5 yinghua yinghua       4096 Sep  8 00:16 ..
-rw-r--r-- 1 yinghua yinghua 1980210686 Sep  1 07:00 company-data.csv <-- Input file for encoding

===========================================================================
Step 5 - Remove null value with double quotes for data encoding
---------------------------------------------------------------------------
sed                             = Stream Editor
’s/""//g’                       = Regular expression to eliminate double quotes in empty field
company-data.csv                = Input file
>                               = Redirection operation
company-data-full.csv           = Output file
===========================================================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ sed ’s/""//g’ company-data.csv > company-data-full.csv

===================================================================
Step 6 - The encoded file is processed and stored in same directory
===================================================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ ls -al
drwxrwxr-x 5 yinghua yinghua       4096 Feb  7 15:09 .
drwxrwxr-x 5 yinghua yinghua       4096 Sep  8 00:16 ..
-rw-r--r-- 1 yinghua yinghua 1980210686 Sep  1 07:00 company-data.csv <-- Input file for encoding
-rw-rw-r-- 1 yinghua yinghua 1751741578 Sep  1 11:39 company-data-full.csv <-- Encoded file
```

Listing J.2: Execution of data encoding with stream editor.

The combination of Linux commands is executed to display the data encoding operations. According to Step 5 in Listing J.2, *company-data.csv* is consume as input file and process with text substitution to eliminate double quotes according to regular expression provided. The execution will redirect *company-data-full.csv* as output file and stored into the same directory as shown in Step 6.

## J.3 View Records in Encoded Company Datasets

```
=======================================================================
List first three rows of data in company-data-full.csv file for display purposes
=======================================================================
yinghua@yinghua-NL8C:~/Documents/FYP/Basic-Company-Data$ head -3 company-data-full.csv

"! LTD","08209948",,,"METROHOUSE 57 PEPPER ROAD","HUNSLET","LEEDS","YORKSHIRE",,"LS10 2RU","Private Limited
     Company","Active","United Kingdom",,"11/09/2012","30","9","30/06/2018","30/09/2016","DORMANT
     ","09/10/2016","11/09/2015","0","0","0","0","99999 - Dormant Company",,,,"0","0","http://business.data
     .gov.uk/id/company/08209948",,,,,,,,,,,,,,,,,,,,,"25/09/2019","11/09/2016"

"!BIG IMPACT GRAPHICS LIMITED","07382019",,,"335 ROSDEN HOUSE","372 OLD STREET","LONDON",,,"EC1V 9AV","
     Private Limited Company","Active","United Kingdom",,"21/09/2010","30","9","30/06/2018","30/09/2016","
     DORMANT","19/10/2016","21/09/2015","0","0","0","0","59112 - Video production activities","59113 -
     Television programme production activities","74100 - specialised design activities","74202 - Other
     specialist photography","0","0","http://business.data.gov.uk/id/company
     /07382019",,,,,,,,,,,,,,,,,,,,,"05/10/2019","21/09/2016"

"!NSPIRED LTD","SC421617",,,"26 POLMUIR ROAD",,"ABERDEEN",,"UNITED KINGDOM","AB11 7SY","Private Limited
     Company","Active","United Kingdom",,"11/04/2012","30","3","30/12/2017","30/03/2016","TOTAL EXEMPTION
     SMALL","09/05/2017","11/04/2016","0","0","0","0","70229 - Management consultancy activities other than
      financial management",,,,"0","0","http://business.data.gov.uk/id/company/SC421617
     ",,,,,,,,,,,,,,,,,,,,,"25/04/2020","11/04/2017"
```

Listing J.3: Three rows of data in Encoded Company Datasets

Listing J.3 display first three rows of data found in encoded company CSV datasets, the double quotes found in empty values are eliminated and removed after the file is encoded. The operation is successful and data consistency is maintained with data encoding activities. The encoded data is safe to be processed by other activities such as data transformation and data parsing.

# Appendix K

# Data Transformation

## K.1 Validate Line Counts in Original Datasets.

```
1  =======================================================================================
2  Step 1 - Change directory to company data location and counts number of lines in file.
3  =======================================================================================
4  yinghua@yinghua:~$ cd ~/Documents/FYP1/FYP-data/company-data/
5  yinghua@yinghua:~/Documents/FYP1/FYP-data/company-data$ wc -l company-data-full.csv
6  3595702 company-data-full.csv                   <-- line count of company datasets
7  
8  =======================================================================================
9  Step 2 - Change directory to NSPL data location and counts number of lines in file.
10 =======================================================================================
11 yinghua@yinghua:~$ cd ~/Documents/FYP1/FYP-data/postcode-data/
12 yinghua@yinghua:~/Documents/FYP1/FYP-data/postcode-data$ wc -l UK-NSPL.csv
13 1754883 UK-NSPL.csv                             <-- line count of NSPL datasets
14 
15 =======================================================================================
16 Step 3 - Change directory to LEO data location and counts number of lines in file.
17 =======================================================================================
18 yinghua@yinghua:~$ cd ~/Documents/FYP1/FYP-data/subject-data/
19 yinghua@yinghua:~/Documents/FYP1/FYP-data/subject-data$ wc -l institution-subject-data.csv
20 32707 institution-subject-data.csv              <-- line count of LEO datasets
```

LISTING K.1: Validate lines counts in CSV datasets.

The **number of lines** in each datasets are required to be recorded before these data are transform and import into PostgreSQL database. This step is conducted to prevent loss of data after the data transformation process and execution failure can be quickly observed during the process. Row 6, 13 and 20 in Listing K.1 show the line counts of each datasets with *wc* commands.

## K.2 PL/pgSQL's scripts for Data Transformation.

### K.2.1 NSPL data transformation script.

```
================================================
Step 1 - Drop the previous created table for demonstration
================================================
drop table nspl_rawdata;

==========================================================================================
Step 2 - Use DDL to define attribute's data types and table for data transformation purpose
==========================================================================================
create table nspl_rawdata (
postcode1               varchar(15)     not null,
postcode2               varchar(15)     not null primary key,
postcode3               varchar(15)     not null,
date_introduce          varchar(10)     not null,
usertype                int             not null,
easting                 int             null default 0,
northing                int             null default 0,
position_quality        int             not null,
countycode              varchar(15)     null default 'Undefined',
countyname              varchar(50)     null default 'Undefined',
county_lac              varchar(15)     null default 'Undefined',
county_lan              varchar(75)     null default 'Undefined',
wardcode                varchar(15)     null default 'Undefined',
wardname                varchar(75)     null default 'Undefined',
countrycode             varchar(15)     null default 'Undefined',
countryname             varchar(30)     null default 'Undefined',
region_code             varchar(15)     null default 'Undefined',
region_name             varchar(30)     null default 'Undefined',
par_cons_code           varchar(15)     null default 'Undefined',
par_cons_name           varchar(50)     null default 'Undefined',
eerc                    varchar(15)     null default 'Undefined',
eern                    varchar(30)     null default 'Undefined',
pctc                    varchar(15)     null default 'Undefined',
pctn                    varchar(70)     null default 'Undefined',
isoac                   varchar(15)     null default 'Undefined',
isoan                   varchar(50)     null default 'Undefined',
msoac                   varchar(15)     null default 'Undefined',
msoan                   varchar(50)     null default 'Undefined',
oacc                    varchar(5)      null default '---',
oacn                    varchar(50)     null default 'Undefined',
longitude               real            not null,
latitude                real            not null,
spatial_accuracy        varchar(30)     null default 'Undefined',
last_upload             date            not null,
location                varchar(50)     null default 'Undefined',
socrataid               int             not null
);

===============================================================================
Step 3 - Perform data transformation execution
-------------------------------------------------------------------------------
\copy                   = Transform data from CSV into PostgreSQL database
nspl_rawdata            = The destination table of data transformation
'/home/yinghua/Documents/FYP1/FYP-data/postcode-data/UK-NSPL.csv' = The directory of raw data
with header csv         = Define the format of migration
===============================================================================
\copy nspl_rawdata from '/home/yinghua/Documents/FYP1/FYP-data/postcode-data/UK-NSPL.csv' with header csv;
```

LISTING K.2: PL/pgSQL's scripts for NSPL data transformation.

The PL/pgSQL script for NSPL data transformation is written to create database entity with well-defined data types for each attributes as shown in Listing K.2. Afterwards, the data transformation is executed to extract CSV data and import into destination table created on Step 2.

### K.2.2 Company data transformation script.

```
==========================================================
Step 1 - Drop the previous created table for demonstration
==========================================================
drop table company_rawdata;

===========================================================================================
Step 2 - Use DDL to define attribute's data types and table for data transformation purpose
===========================================================================================
create table company_rawdata (
CompanyName             varchar(160) null default 'Undefined',
CompanyNumber           varchar(8) not null,
CareOf                  varchar(100) null default 'Undefined',
POBox                   varchar(10) null default 'Undefined',
AddressLine1            varchar(300) null default 'Undefined',
AddressLine2            varchar(300) null default 'Undefined',
PostTown                varchar(50) null default 'Undefined',
County                  varchar(50) null default 'Undefined',
Country                 varchar(50) null default 'Undefined',
PostCode                varchar(20) null default 'Undefined',
CompanyCategory         varchar(100) not null,
CompanyStatus           varchar(70) not null,
CountryOfOrigin         varchar(50) not null,
DissolutionDate         varchar(20) null default '3000-01-01',
IncorporationDate       varchar(20) null default '3000-01-01',
AccountingRefDay        int null default 0,
AccountingRefMonth      int null default 0,
Account_NextDueDate     varchar(20) null default '3000-01-01',
Account_LastMadeUpdate  varchar(20) null default '3000-01-01',
AccountCategory         varchar(30) null default 'Undefined',
Return_NextDueDate      varchar(20) null default '3000-01-01',
Return_LastMadeUpDate   varchar(20) null default '3000-01-01',
NumMortCharges          int not null,
NumMortOutstanding      int not null,
NumMortPartSatisfied    int not null,
NumMortSatisfied        int not null,
SICCode1                varchar(170) null default 'Undefined',
SICCode2                varchar(170) null default 'Undefined',
SICCode3                varchar(170) null default 'Undefined',
SICCode4                varchar(170) null default 'Undefined',
NumGenPartners          int not null,
NumLimPartners          int not null,
URI                     varchar(47) not null,
pn1_CONDate             varchar(20) null default '3000-01-01',
pn1_CompanyName         varchar(160) null default 'Undefined',
pn2_CONDate             varchar(20) null default '3000-01-01',
pn2_CompanyName         varchar(160) null default 'Undefined',
pn3_CONDate             varchar(20) null default '3000-01-01',
pn3_CompanyName         varchar(160) null default 'Undefined',
pn4_CONDate             varchar(20) null default '3000-01-01',
pn4_CompanyName         varchar(160) null default 'Undefined',
pn5_CONDate             varchar(20) null default '3000-01-01',
pn5_CompanyName         varchar(160) null default 'Undefined',
pn6_CONDate             varchar(20) null default '3000-01-01',
pn6_CompanyName         varchar(160) null default 'Undefined',
pn7_CONDate             varchar(20) null default '3000-01-01',
pn7_CompanyName         varchar(160) null default 'Undefined',
pn8_CONDate             varchar(20) null default '3000-01-01',
pn8_CompanyName         varchar(160) null default 'Undefined',
pn9_CONDate             varchar(20) null default '3000-01-01',
pn9_CompanyName         varchar(160) null default 'Undefined',
pn10_CONDate            varchar(20) null default '3000-01-01',
pn10_CompanyName        varchar(160) null default 'Undefined',
ConfStmtNextDueDate     varchar(20) default '3000-01-01',
ConfStmtLastMadeUpDate  varchar(20) default '3000-01-01'
);

==================================================================================
Step 3 - Perform data transformation execution
==================================================================================
\copy company_rawdata from '/home/yinghua/Documents/FYP1/FYP-data/company-data/company-data-full.csv' with header csv;
```

Listing K.3: PL/pgSQL's scripts for Company data transformation.

The PL/pgSQL script for Company data transformation is written to create database entity with well-defined data types for each attributes as shown in Listing K.3. Afterwards, the data transformation is executed to extract CSV data and import into destination table created on Step 2.

### K.2.3 LEO data transformation script.

```
==========================================================
Step 1 - Drop the previous created table for demonstration
==========================================================
drop table leo_rawdata;

===========================================================================================
Step 2 - Use DDL to define attribute's data types and table for data transformation purpose
===========================================================================================
create table leo_rawdata (

ukprn                   int         not null,
providername            varchar(100) not null,
region                  varchar(50) not null,
subject                 varchar(50) not null,
sex                     varchar(30) not null,
yearaftergraduation     int         not null,
grads                   varchar(10) null default null,
unmatched               varchar(20) null default null,
matched                 varchar(20) null default null,
activityNotCaptured     varchar(20) null default null,
nosustdest              varchar(20) null default null,
sustemponly             varchar(20) null default null,
sustemp                 varchar(20) null default null,
sustempfsorboth         varchar(20) null default null,
earningsinclude         varchar(20) null default null,
lowerannearn            varchar(20) null default null,
medianannearn           varchar(20) null default null,
upperannearn            varchar(20) null default null,
polargrpone             varchar(20) null default null,
polargrponeincluded     varchar(20) null default null,
prattband               varchar(20) null default null,
prattincluded           varchar(20) null default null

);

==================================================================================
Step 3 - Perform data transformation execution
==================================================================================
\copy leo_rawdata from '/home/yinghua/Documents/FYP1/FYP-data/subject-data/institution-subject-data.csv'
     with header csv;
```

Listing K.4: PL/pgSQL's scripts for LEO data transformation.

The PL/pgSQL script for LEO data transformation is written to create database entity with well-defined data types for each attributes as shown in Listing K.4. Afterwards, the data transformation is executed to extract CSV data and import into destination table created on Step 2.

## K.3 Data Transformation execution.

### K.3.1 NSPL data transformation execution.

```
 1  ======================================================================================
 2  Step 1 - Change to contain postcode raw data directory and check the location of scripts
 3  ======================================================================================
 4  yinghua@yinghua:~$ cd ~/gitRepo/final-year-project/FYP2-Database-Queries/postcode-database-queries
 5  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/postcode-database-queries$ ls -al
 6  total 44
 7  drwxrwxr-x 2 yinghua yinghua 4096 Feb  7 16:22 .
 8  drwxrwxr-x 5 yinghua yinghua 4096 Jan 29 22:58 ..
 9  -rw-rw-r-- 1 yinghua yinghua 4264 Feb  7 16:22 01_yinghua_raw_postcode_DDL.sql     <- This script
10  -rw-rw-r-- 1 yinghua yinghua 7554 Jan 17 15:48 02_yinghua_normalized_NSPL_DDL.sql
11  -rw-rw-r-- 1 yinghua yinghua 5164 Jan 14 18:06 03_yinghua_insert_NSPL_table.sql
12  -rw-rw-r-- 1 yinghua yinghua 1252 Jan 13 22:06 postcode_format.sql
13  -rw-rw-r-- 1 yinghua yinghua 2224 Jan 15 14:54 test2.sql
14  -rw-rw-r-- 1 yinghua yinghua 3416 Jan 14 18:29 test.sql
15
16  ======================================================================================
17  Step 2 - Execution of data transformation scripts
18  ======================================================================================
19  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/postcode-database-queries$ psql -U
         yinghua -d postcode -a -f 01_yinghua_raw_postcode_DDL.sql
20
21  (output too much not shown...)
22
23  ======================================================================================
24  Step 3 - Connect to postcode database
25  ======================================================================================
26  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/postcode-database-queries$ psql postcode;
27  psql (9.5.10)
28  Type "help" for help.
29
30  postcode=#
31
32  ======================================================================================
33  Step 4 - Select number of rows of table in database after data transformation
34  ======================================================================================
35  postcode=# select distinct count(*) from nspl_rawdata;
36  count
37  ---------
38  1754882
39  (1 row)
```

Listing K.5: Execution of PL/pgSQL's scripts for NSPL data transformation.

The execution of NSPL data transformation scripts stated in Section K.2.1 is performed in Step 2 (Row 16-22) at Listing K.5.

The command required username (yinghua), database (postcode) and script name (01_yinghua_raw_postcode_DDL.sql) as parameter to execute the script for security and control access purposes.

Once the execution is complete, the number of row in destination table is verified against the number of lines in postcode dataset (performed in Section K.1). The postcode data transformation is success because the data is not missing and import successfully without errors.

## K.3.2 Company data transformation execution.

```
 1  =====================================================================================
 2  Step 1 - Change to contain company raw data directory and check the location of scripts
 3  =====================================================================================
 4  yinghua@yinghua:~$ cd ~/gitRepo/final-year-project/FYP2-Database-Queries/company-database-queries
 5  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/company-database-queries$ ls -al
 6  total 32
 7  drwxrwxr-x 2 yinghua yinghua 4096 Jan 29 22:57 .
 8  drwxrwxr-x 5 yinghua yinghua 4096 Jan 29 22:58 ..
 9  -rw-rw-r-- 1 yinghua yinghua 3427 Jan 27 11:42 00_yinghua_company_csv_db_migration.sql  <- This script
10  -rw-rw-r-- 1 yinghua yinghua 2883 Jan 27 11:46 01_yinghua_create_company_table.sql
11  -rw-rw-r-- 1 yinghua yinghua 6923 Jan 28 16:43 02_yinghua_normalized_company_DDL.sql
12  -rw-rw-r-- 1 yinghua yinghua 3221 Jan 28 15:59 03_yinghua_insert_normalized_table_DML.sql
13  -rw-rw-r-- 1 yinghua yinghua  365 Jan 20 01:59 session_run.txt
14
15  =====================================================================================
16  Step 2 - Execution of data transformation scripts
17  =====================================================================================
18  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/company-database-queries$ psql -U yinghua
          -d company -a -f 00_yinghua_company_csv_db_migration.sql
19
20  (output too much not shown...)
21
22  =====================================================================================
23  Step 3 - Connect to company database
24  =====================================================================================
25  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/company-database-queries$ psql company;
26  psql (9.5.10)
27  Type "help" for help.
28
29  company=#
30
31  =====================================================================================
32  Step 4 - Select number of rows of table in database after data transformation
33  =====================================================================================
34  company=# select distinct count(*) from company_rawdata;
35  count
36  ---------
37  3595702
38  (1 row)
```

Listing K.6: Execution of PL/pgSQL's scripts for Company data transformation.

The execution of company data transformation scripts stated in Section K.2.2 is performed in Step 2 (Row 15-20) at Listing K.6.

The command required username (yinghua), database (company) and script name (00_yinghua_company_csv_db_migration.sql) as parameter to execute the script for security and control access purposes.

Once the execution is complete, the number of row in destination table is verified against the number of lines in company dataset (performed in Section K.1). The company data transformation is success because the data is not missing and import successfully without errors.

### K.3.3 LEO data transformation execution.

```
 1  ===================================================================================
 2  Step 1 - Change to contain education raw data directory and check the location of scripts
 3  ===================================================================================
 4  yinghua@yinghua:~$ cd ~/gitRepo/final-year-project/FYP2-Database-Queries/education-database-queries
 5  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/education-database-queries$ ls -al
 6  total 28
 7  drwxrwxr-x 2 yinghua yinghua 4096 Jan 28 14:34 .
 8  drwxrwxr-x 5 yinghua yinghua 4096 Jan 29 22:58 ..
 9  -rw-rw-r-- 1 yinghua yinghua 1721 Jan  5 14:03 01_yinghua_raw_leo_table_DDL.sql          <- This script
10  -rw-rw-r-- 1 yinghua yinghua 4703 Jan 12 11:16 02_yinghua_normalized_leo_table_DDL.sql
11  -rw-rw-r-- 1 yinghua yinghua 3292 Jan 10 01:56 03_yinghua_insert_leo_table_DML.sql
12  -rw-rw-r-- 1 yinghua yinghua 2425 Jan 12 11:22 04_yinghua_leo_data_migration.sql
13
14
15  ===================================================================================
16  Step 2 - Execution of data transformation scripts
17  ===================================================================================
18  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/education-database-queries$ psql -U
         yinghua -d education -a -f 01_yinghua_raw_leo_table_DDL.sql
19
20  (output too much not shown...)
21
22  ===================================================================================
23  Step 3 - Connect to company database
24  ===================================================================================
25  yinghua@yinghua:~/gitRepo/final-year-project/FYP2-Database-Queries/education-database-queries$ psql
         education;
26  psql (9.5.10)
27  Type "help" for help.
28
29  education=#
30
31  ===================================================================================
32  Step 4 - Select number of rows of table in database after data transformation
33  ===================================================================================
34  education=# select distinct count(*) from leo_rawdata;
35  count
36  -------
37  32706
38  (1 row)
```

LISTING K.7: Execution of PL/pgSQL's scripts for LEO data transformation.

The execution of LEO data transformation scripts stated in Section K.2.3 is performed in Step 2 (Row 15-20) at Listing K.7. The command required username (yinghua), database (education) and script name (01_yinghua_raw_leo_table_DDL.sql) as parameter to execute the script for security and control access purposes.

Once the execution is complete, the number of row in destination table is verified against the number of lines in LEO dataset (performed in Section K.1). The LEO data transformation is success because the data is not missing and import successfully without errors.

# Appendix L

# Data Retrieval

## L.1 Go program for CSV file data retrieval.

### L.1.1 Go Sequential program source codes.

```go

package main

import (
        "bufio"
        "encoding/csv"
        "fmt"
        "io"
        "os"
        "time"

        _ "github.com/lib/pq"
)


func retrieve_without_channel(directory string, indicator string) {

        fmt.Printf("BEGIN retrieve data from %s files. \n", indicator);

        csvFile, err := os.Open(directory)
        checkErr(err, "Open CSV")

        defer csvFile.Close()

        // get the time before execution
        start := time.Now()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for {
                _, err := reader.Read()

                // Stop at EOF.
                if err == io.EOF {
                        break
                }

        }

        // obtain the time after execution
        fmt.Printf("FINISH retrieve all rows of data from %s files with %.5fs seconds. \n", indicator, time.Since(start).Seconds())

}

func sequential_csv() {

        // get the time before execution
        start := time.Now()

        retrieve_without_channel(LEO_DIRECTORY, LEO_INDICATOR);
        retrieve_without_channel(COMPANY_DIRECTORY, COMPANY_INDICATOR);
        retrieve_without_channel(NSPL_DIRECTORY, NSPL_INDICATOR);

        // obtain the time after execution
        fmt.Printf("%.5fs seconds on retrieve all the data SEQUENTIALLY. \n", time.Since(start).Seconds())
}

/**

yinghua@yinghua:~/gitRepo/go-read-csv/src/main$ go build *.go
yinghua@yinghua:~/gitRepo/go-read-csv/src/main$ time go run *.go

BEGIN retrieve data from subject files.
FINISH retrieve all rows of data from subject files with 0.09179 seconds.
BEGIN retrieve data from company files.
FINISH retrieve all rows of data from company files with 32.64937 seconds.
BEGIN retrieve data from postcode files.
FINISH retrieve all rows of data from postcode files with 13.07156 seconds.
45.81286s seconds on retrieve all the data SEQUENTIALLY.

real    0m46.050s
user    0m46.651s
sys     0m0.612s

**/
```

Listing L.1: Go sequential program source codes. (sequential-csv.go)

Listing L.1 shows the source code of Go programming language based application that retrieve all rows of data from NSPL, company and LEO datasets in sequential manner. The program will open the each raw datasets stored in predefine directory and began to read all lines of records in the CSV file. Ultimately, the execution time will be display and recorded for comparison in result and discussion.

### L.1.2 Go Concurrent program source codes.

```
package main

import (
        "bufio"
        "encoding/csv"
        "fmt"
        "io"
        "os"
        "time"

        _ "github.com/lib/pq"
)

===================================================================
Function that retrieve data with Goroutune and passed into Gochannel
===================================================================
func retrieve_data_with_channel(directory string, indicator string, msg chan string) {

        fmt.Printf("BEGIN retrieve data from %s files. \n", indicator);

        csvFile , err := os.Open(directory)
        checkErr(err, "Open CSV")

        defer csvFile.Close()

        // get the time before execution
        start := time.Now()

        // Create a new reader.
        reader := csv.NewReader(bufio.NewReader(csvFile))

        for {

                _ , err := reader.Read()

                // Stop at EOF.
                if err == io.EOF {
                        break
                }
        }

        // obtain the time after execution
        fmt.Printf("FINISH retrieve all rows of data from %s files with %.5fs seconds.", indicator, time.Since(start).Seconds())
        msg <- " "

}

=============================================
Select function that receive Goroutine message
=============================================
func goSelect(ch_company, ch_leo, ch_nspl chan string) {


        for i := 0; i < 3; i++ {

                select {
                case msg1 := <-ch_leo:
                        fmt.Println(msg1)
                case msg2 := <-ch_company:
                        fmt.Println(msg2)
                case msg3 := <-ch_nspl:
                        fmt.Println(msg3)
                }
        }
}

=============================================
This function read all CSV data concurrently
=============================================
func concurrent_csv() {

        // get the time before execution
        start := time.Now()

        // make three channel for three functions
        ch_company := make(chan string)
        ch_leo := make(chan string)
        ch_nspl := make(chan string)


        go retrieve_data_with_channel(LEO_DIRECTORY, LEO_INDICATOR, ch_leo);
        go retrieve_data_with_channel(COMPANY_DIRECTORY, COMPANY_INDICATOR, ch_company);
        go retrieve_data_with_channel(NSPL_DIRECTORY, NSPL_INDICATOR, ch_nspl);

        goSelect(ch_company, ch_leo, ch_nspl)

        // obtain the time after execution
        fmt.Printf("T%.5fs seconds on retrieve all the data CONCURRENTLY. \n", time.Since(start).Seconds())
}

/**

BEGIN retrieve data from postcode files.
BEGIN retrieve data from subject files.
BEGIN retrieve data from company files.
FINISH retrieve all rows of data from subject  files with 0.12362 seconds.
FINISH retrieve all rows of data from postcode files with 15.21926 seconds.
FINISH retrieve all rows of data from company  files with 36.22334 seconds.
36.22355 seconds on retrieve all the data CONCURRENTLY.

real    0m36.478s
user    0m52.337s
sys     0m0.719s

**/
```

Listing L.2: Go concurrent program source codes. (concurrent-csv.go)

Listing L.2 shows the source code of Go programming language based application that retrieve all rows of data from NSPL, company and LEO datasets in concurrent manner. Three *Goroutines* is created and each Gorountine is assigned by a job (function) to complete the job. *GoSelect* is used to receive the thread that completed the process and update the state of specific operations.

The program will open each raw datasets stored in predefine directory simultaneously and began to read all lines of records in the CSV file concurrently. Ultimately, the execution time will be display and recorded for comparison in result and discussion.

## L.2 Go program for PostgreSQL database retrieval with ORM.

In this project, we developed our own Object Relational Mapping (ORM) tools to convert data into object model for data handling and manipulation. *Struct* is created to define as *object* that contain characteristic and attributes of elements and ready to be mapped by data retrieved from PostgreSQL database.

Therefore, **NSPL struct**, **Company struct** and **LEO struct** are created with separate file in each program.

### L.2.1 NSPL struct

```

============================
// 36 columns 1754882 rows
============================
type Nspl struct {
postcode1               string
postcode2               string
postcode3               string
date_introduce          string
usertype                int  // 5

easting                 sql.NullInt64
northing                sql.NullInt64
position_quality        int
countycode              sql.NullString
countyname              sql.NullString // 10

county_lac              sql.NullString
county_lan              sql.NullString
wardcode                sql.NullString
wardname                sql.NullString
countrycode             sql.NullString // 15

countryname             sql.NullString
region_code             sql.NullString
region_name             sql.NullString
par_cons_code           sql.NullString
par_cons_name           sql.NullString // 20

eerc                    sql.NullString
eern                    sql.NullString
pctc                    sql.NullString
pctn                    sql.NullString
isoac                   sql.NullString // 25

isoan                   sql.NullString
msoac                   sql.NullString
msoan                   sql.NullString
oacc                    sql.NullString
oacn                    sql.NullString
longitude               float64         // 31

latitude                float64
spatial_accuracy        sql.NullString
last_upload             string
location                sql.NullString
socrataid               int             // 36

}
```

Listing L.3: Source code for NSPL struct. (nspl.go)

### L.2.2 Company struct

```
======================
3595702 rows 55 columns
======================
type Company struct {

        name                            sql.NullString
        number                          string
        careOf                          sql.NullString
        poBox                           sql.NullString
        addressLine1                    sql.NullString // 5

        addressLine2                    sql.NullString
        postTown                        sql.NullString
        county                          sql.NullString
        country                         sql.NullString
        postcode                        sql.NullString // 10

        category                        string
        status                          string
        countryOfOrigin                 string
        dissolution_date                sql.NullString
        incorporate_date                sql.NullString // 15

        accounting_refDay               sql.NullInt64
        accounting_refMonth             sql.NullInt64
        account_nextDueDate             sql.NullString
        account_lastMadeUpdate          sql.NullString
        account_category                sql.NullString // 20

        return_nextDueDate              sql.NullString
        return_lastMadeUpdate           sql.NullString
        num_MortChanges                 int64
        num_MortOutstanding             int64
        num_MortPartSatisfied           int64 // 25

        num_MortSatisfied               int64
        siccode1                        sql.NullString
        siccode2                        sql.NullString
        siccode3                        sql.NullString
        siccode4                        sql.NullString // 30

        num_genPartner                  int
        num_limPartner                  int
        uri                             string
        pn1_condate                     sql.NullString
        pn1_companydate                 sql.NullString // 35

        pn2_condate                     sql.NullString
        pn2_companydate                 sql.NullString
        pn3_condate                     sql.NullString
        pn3_companydate                 sql.NullString
        pn4_condate                     sql.NullString // 40

        pn4_companydate                 sql.NullString
        pn5_condate                     sql.NullString
        pn5_companydate                 sql.NullString
        pn6_condate                     sql.NullString
        pn6_companydate                 sql.NullString // 45

        pn7_condate                     sql.NullString
        pn7_companydate                 sql.NullString
        pn8_condate                     sql.NullString
        pn8_companydate                 sql.NullString
        pn9_condate                     sql.NullString // 50

        pn9_companydate                 sql.NullString
        pn10_condate                    sql.NullString
        pn10_companydate                sql.NullString
        conf_stmtNextDueDate            sql.NullString
        conf_stmtLastMadeUpdate         sql.NullString // 55
}
```

Listing L.4: Source code for Company struct. (company.go)

### L.2.3 LEO struct

```
======================
32706 rows 22 columns
======================
type Leo struct {

        ukprn                           int
        providername                    string
        region                          string
        subject                         string
        sex                             string // 5

        yearAfterGraduation             string
        grads                           sql.NullString
        unmatched                       sql.NullString
        matched                         sql.NullString
        activitynocaptured              sql.NullString //10

        nosustdest                      sql.NullString
        sustemponly                     sql.NullString
        sustemp                         sql.NullString
        sustempfsorboth                 sql.NullString
        earningsinclude                 sql.NullString //15

        lowerannearn                    sql.NullString
        medianannearn                   sql.NullString
        upperannearn                    sql.NullString
        polargrpone                     sql.NullString
        polargrponeincluded             sql.NullString //20

        prattband                       sql.NullString
        prattincluded                   sql.NullString //22
}
```

Listing L.5: Source code for LEO struct. (leo.go)

Listing L.2, L.3 and L.4 shows the source code of NSPL, Company and LEO struct created in Go ORM program. Table below explain the specification of types conversion and choice data type used in these struct.

| **Data type in PostgreSQL** | **Data type in Go** | **Specification** |
|---|---|---|
| INTEGER(10) | int | store signed 32 bits integer. |
| BIGINT | int64 | store signed 64 bits integer. |
| VARCHAR | string | store alphanumeric and alphabets. |
| INT or BIGINT | sql.NullInt64 | store NULL values or 64 bits integer. |
| VARCHAR | sql.NullString | store NULL values or string. |
| REAL or DECIMAL | float64 | store signed 64 bit decimal. |

Table L.1: Data type specification in Go programming language

It is essential to understand and declared valid data types for object relational mapping to prevent type errors and data corruption. The attributes of each struct are declared and defined with correct data types for data conversion.

### L.2.4 Go sequential program source code

#### L.2.4.1 Company data retrieval function

```
 1 ==========================================================================
 2 Retrieving 3595702 rows of data from PostgreSQL database in sequential manner
 3 ==========================================================================
 4 func retrieve_company() {
 5
 6         fmt.Println("BEGIN retrieve data from companydata database.")
 7
 8         // get the time before execution
 9         start := time.Now()
10
11
12         rows, err := db.Query("SELECT * FROM companydata;")
13
14         checkErr(err, "Error on query DB")
15
16         for rows.Next() {
17
18                 var c Company
19
20                 err = rows.Scan(&c.name, &c.number, &c.careOf, &c.poBox, &c.addressLine1,
21                 &c.addressLine2, &c.postTown, &c.county, &c.country, &c.postcode,
22                 &c.category, &c.status, &c.countryOfOrigin, &c.dissolution_date, &c.incorporate_date,
23                 &c.accounting_refDay, &c.accounting_refMonth, &c.account_nextDueDate, &c.
      account_lastMadeUpdate, &c.account_category,
24                 &c.return_nextDueDate, &c.return_lastMadeUpdate, &c.num_MortChanges, &c.
      num_MortOutstanding, &c.num_MortPartSatisfied,
25                 &c.num_MortSatisfied, &c.siccode1, &c.siccode2, &c.siccode3, &c.siccode4,
26                 &c.num_genPartner, &c.num_limPartner, &c.uri, &c.pn1_condate, &c.pn1_companydate,
27                 &c.pn2_condate, &c.pn2_companydate, &c.pn3_condate, &c.pn3_companydate, &c.pn4_condate,
28                 &c.pn4_companydate,&c.pn5_condate, &c.pn5_companydate, &c.pn6_condate, &c.pn6_companydate,
29                 &c.pn7_condate, &c.pn7_companydate, &c.pn8_condate, &c.pn8_companydate, &c.pn9_condate,
30                 &c.pn9_companydate, &c.pn10_condate, &c.pn10_companydate, &c.conf_stmtNextDueDate, &c.
      conf_stmtLastMadeUpdate)
31                 checkErr(err, "Read company data rows,")
32
33                 //                      fmt.Printf("%+v\n", c)
34         }
35
36         // obtain the time after execution
37         fmt.Printf("FINISH retrieve all rows of data from company database with %.5fs seconds. \n", time.
      Since(start).Seconds())
38
39 }
```

Listing L.6: Function for company data retrieval. (retrieve_company.go)

Listing L.6 shows the source code of company data retrieval function that SELECT 3595702 rows of company data from PostgreSQL database in **sequential** manner. The function will establish connection with database and perform transaction to retrieve all rows of data and map into the object declared (refer row 16-34).

The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 37). The results will be tabulated and discussed in results and finding section.

### L.2.4.2 NSPL data retrieval function

```
1  =============================================================================
2  Retrieving 1754882 rows of data from PostgreSQL database in sequential manner
3  =============================================================================
4  func retrieve_nspl() {
5  
6          fmt.Println("BEGIN retrieve data from nspl database.")
7  
8          // get the time before execution
9          start := time.Now()
10 
11         rows, err := db.Query("SELECT * FROM nspl;")
12 
13         checkErr(err, "Error on query DB")
14 
15         for rows.Next() {
16 
17                 var n Nspl
18 
19                 err = rows.Scan(&n.postcode1, &n.postcode2, &n.postcode3, &n.date_introduce, &n.usertype,
20                 &n.easting, &n.northing, &n.position_quality, &n.countrycode, &n.countryname,
21                 &n.county_lac, &n.county_lan, &n.wardcode, &n.wardname, &n.countrycode,
22                 &n.countryname, &n.region_code, &n.region_name, &n.par_cons_code, &n.par_cons_name,
23                 &n.eerc, &n.eern, &n.pctc, &n.pctn, &n.isoac, &n.isoan,
24                 &n.msoac, &n.msoan, &n.oacc, &n.oacn, &n.longitude,
25                 &n.latitude, &n.spatial_accuracy, &n.last_upload, &n.location, &n.socrataid)
26                 checkErr(err, "Read company data rows,")
27 
28                 //                      fmt.Printf("%+v\n", n)
29         }
30 
31         fmt.Printf("FINISH retrieve all rows of data from nspl database with %.5fs seconds. \n", time.Since(
       start).Seconds())
32 }
```

Listing L.7: Function for NSPL data retrieval. (retrieve_nspl.go)

Listing L.7 shows the source code of NSPL data retrieval function that SELECT 1754882 rows of company data from PostgreSQL database in **sequential** manner. The function will establish connection with database and perform transaction to retrieve all rows of data and map into the object declared (refer row 15-29).

The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 31). The results will be tabulated and discussed in results and finding section.

### L.2.4.3 LEO data retrieval function

```
1   ============================================================================
2   Retrieving 32706 rows of data from PostgreSQL database in sequential manner
3   ============================================================================
4   func retrieve_leo() {
5   
6           fmt.Println("BEGIN retrieve data from leo database.")
7   
8           // get the time before execution
9           start := time.Now()
10          rows, err := db.Query("SELECT * FROM leo;")
11  
12          checkErr(err, "Error on query DB")
13  
14          for rows.Next() {
15  
16                  var l Leo
17  
18                  err = rows.Scan(&l.ukprn, &l.providername, &l.region, &l.subject, &l.sex,
19                  &l.yearAfterGraduation, &l.grads, &l.unmatched, &l.matched, &l.activitynocaptured,
20                  &l.nosustdest, &l.sustemponly, &l.sustemp, &l.sustempfsorboth, &l.earningsinclude,
21                  &l.lowerannearn, &l.medianannearn, &l.upperannearn, &l.polargrpone, &l.polargrponeincluded,
22                  &l.prattband, &l.prattincluded)
23                  checkErr(err, "Read LEO data rows,")
24  
25                  //                      fmt.Printf("%+v\n", l)
26          }
27  
28          fmt.Printf("FINISH retrieve all rows of data from leo database with %.5fs seconds. \n", time.Since(
        start).Seconds())
29  
30  }
```

Listing L.8: Function for LEO data retrieval. (retrieve_leo.go)

Listing L.8 shows the source code of LEO data retrieval function that SELECT 32706 rows of company data from PostgreSQL database in **sequential** manner. The function will establish connection with database and perform transaction to retrieve all rows of data and map into the object declared (refer row 14-26).

The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 28). The results will be tabulated and discussed in results and finding section.

#### L.2.4.4 Main function

```
package main

import (
        "fmt"
        "database/sql"
        "time"

        _ "github.com/jinzhu/gorm/dialects/postgres"
        _ "github.com/lib/pq"
)

const (
        DB_USER     = "yinghua"
        DB_PASSWORD = "123"
        DB_NAME     = "fyp1"
)

var db *sql.DB

//==================================================
//function to check error and print error messages
//==================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//==================================================
// initialize connection with database
//==================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        sqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "Initialize database")
        db = sqldb

}

//============================================================
Retrieve all data from PostgreSQL database in sequential manner
//============================================================
func sequential_read() {

        // get the time before execution
        start := time.Now()

        initDB()
        retrieve_company()
        retrieve_leo()
        retrieve_nspl()

        // obtain the time after execution
        fmt.Printf("%.5fs seconds on retrieve all the data from database SEQUENTIALLY. \n", time.Since(start
    ).Seconds())

}

func main() {
        sequential_read()
}

/**

yinghua@yinghua:~/gitRepo/go-read-psql/src/main$ go build *.go
yinghua@yinghua:~/gitRepo/go-read-psql/src/main$ time go run *.go

BEGIN retrieve data from companydata database.
FINISH retrieve all rows of data from companydata database with 39.87781s seconds.
BEGIN retrieve data from leo database.
FINISH retrieve all rows of data from leo database with 0.22304s seconds.
BEGIN retrieve data from nspl database.
FINISH retrieve all rows of data from nspl database with 11.96392s seconds.
52.06485s seconds on retrieve all the data from database SEQUENTIALLY.

real    0m52.358s
user    0m53.685s
sys     0m1.533s

**/
```

Listing L.9: Main function for sequential execution. (main.go)

Listing L.9 shows the source code for main function of Go programming language based PostgreSQL database retrieval program. The main function is where **a program start its execution**. When the program is compiled and executed, main() will call sequential_read() function to initiate data retrieval operation from three tables sequentially (refer row 60).

The program will first establish connection to PostgreSQL database with user, password and database name provided. Then, it will began to retrieve data from company table, LEO table and follow by NSPL table (refer row 51-53) by calling three functions shown in Listing L.6, L.7 and L.8. The total execution time of entire program will be display and print on terminal (refer row 56).

The result obtained will be tabulated and discussed.

## L.2.5 Go concurrent program source code

### L.2.5.1 Company data retrieval function

```
1  =============================================================================
2  Retrieving 3595702 rows of data from PostgreSQL database in concurrent manner
3  =============================================================================
4  func retrieve_company_with_channel(msg chan string) {
5
6          fmt.Println("BEGIN retrieve data from companydata database.")
7
8          // get the time before execution
9          start := time.Now()
10
11
12         rows , err := db.Query("SELECT * FROM companydata;")
13
14         checkErr(err , "Error on query DB")
15
16         for rows.Next() {
17
18                 var c Company
19
20                 err = rows.Scan(&c.name , &c.number , &c.careOf , &c.poBox , &c.addressLine1 ,
21                 &c.addressLine2 , &c.postTown , &c.county , &c.country , &c.postcode ,
22                 &c.category , &c.status ,  &c.countryOfOrigin ,  &c.dissolution_date ,  &c.incorporate_date ,
23                 &c.accounting_refDay ,  &c.accounting_refMonth ,  &c.account_nextDueDate ,  &c.
       account_lastMadeUpdate ,  &c.account_category ,
24                 &c.return_nextDueDate ,  &c.return_lastMadeUpdate ,  &c.num_MortChanges ,  &c.
       num_MortOutstanding ,  &c.num_MortPartSatisfied ,
25                 &c.num_MortSatisfied ,  &c.siccode1 ,  &c.siccode2 ,  &c.siccode3 ,  &c.siccode4 ,
26                 &c.num_genPartner , &c.num_limPartner , &c.uri , &c.pn1_condate , &c.pn1_companydate ,
27                 &c.pn2_condate , &c.pn2_companydate , &c.pn3_condate , &c.pn3_companydate , &c.pn4_condate ,
28                 &c.pn4_companydate ,&c.pn5_condate , &c.pn5_companydate , &c.pn6_condate , &c.pn6_companydate ,
29                 &c.pn7_condate , &c.pn7_companydate , &c.pn8_condate , &c.pn8_companydate , &c.pn9_condate ,
30                 &c.pn9_companydate , &c.pn10_condate , &c.pn10_companydate , &c.conf_stmtNextDueDate , &c.
       conf_stmtLastMadeUpdate)
31                 checkErr(err , "Read company data rows,")
32
33                 //                      fmt.Printf("%+v\n", c)
34         }
35
36         // obtain the time after execution
37         fmt.Printf("FINISH retrieve all rows of data from companydata database with %.5fs seconds. ", time.
       Since(start).Seconds())
38         msg <- " "
39
40 }
```

Listing L.10: Function for company data retrieval. (retrieve_company.go)

Listing L.10 shows the source code of company data retrieval function that SELECT 3595702 rows of company data from PostgreSQL database in **concurrent** manner. The function possess one parameter to allow *Gochannel* to be assigned for concurrent execution.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 16-34). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 37). The results will be tabulated and discussed in results and finding section.

### L.2.5.2 NSPL data retrieval function

```
1  ========================================================================
2  Retrieving 1754882 rows of data from PostgreSQL database in concurrent manner
3  ========================================================================
4  func retrieve_nspl_with_channel(msg chan string) {
5
6          fmt.Println("BEGIN retrieve data from nspl database.")
7
8          // get the time before execution
9          start := time.Now()
10
11         rows, err := db.Query("SELECT * FROM nspl;")
12
13         checkErr(err, "Error on query DB")
14
15         for rows.Next() {
16
17                 var n Nspl
18
19                 err = rows.Scan(&n.postcode1, &n.postcode2, &n.postcode3, &n.date_introduce, &n.usertype,
20                 &n.easting, &n.northing, &n.position_quality, &n.countrycode, &n.countryname,
21                 &n.county_lac, &n.county_lan, &n.wardcode, &n.wardname, &n.countrycode,
22                 &n.countryname, &n.region_code, &n.region_name, &n.par_cons_code, &n.par_cons_name,
23                 &n.eerc, &n.eern, &n.pctc, &n.pctn, &n.isoac, &n.isoan,
24                 &n.msoac, &n.msoan, &n.oacc, &n.oacn, &n.longitude,
25                 &n.latitude, &n.spatial_accuracy, &n.last_upload, &n.location, &n.socrataid)
26                 checkErr(err, "Read company data rows,")
27
28                 //                      fmt.Printf("%+v\n", n)
29         }
30
31         fmt.Printf("FINISH retrieve all rows of data from nspl database with %.5fs seconds. ", time.Since(
       start).Seconds())
32         msg <- " "
33
34 }
```

Listing L.11: Function for NSPL data retrieval. (retrieve_nspl.go)

Listing L.11 shows the source code of NSPL data retrieval function that SELECT 1754882 rows of NSPL data from PostgreSQL database in **concurrent** manner. The function possess one parameter to allow *Gochannel* to be assigned for concurrent execution.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 15-29). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 31). The results will be tabulated and discussed in results and finding section.

### L.2.5.3 LEO data retrieval function

```
1  ========================================================================
2  Retrieving 32706 rows of data from PostgreSQL database in concurrent manner
3  ========================================================================
4  func retrieve_leo_with_channel(msg chan string) {
5
6          fmt.Println("BEGIN retrieve data from leo database.")
7
8          // get the time before execution
9          start := time.Now()
10         rows, err := db.Query("SELECT * FROM leo;")
11
12         checkErr(err, "Error on query DB")
13
14         for rows.Next() {
15
16                 var l Leo
17
18                 err = rows.Scan(&l.ukprn, &l.providername, &l.region, &l.subject, &l.sex,
19                 &l.yearAfterGraduation, &l.grads, &l.unmatched, &l.matched, &l.activitynocaptured,
20                 &l.nosustdest, &l.sustemponly, &l.sustemp, &l.sustempfsorboth, &l.earningsinclude,
21                 &l.lowerannearn, &l.medianannearn, &l.upperannearn, &l.polargrpone, &l.polargrponeincluded,
22                 &l.prattband, &l.prattincluded)
23                 checkErr(err, "Read LEO data rows,")
24
25                 //                      fmt.Printf("%+v\n", l)
26         }
27
28         fmt.Printf("FINISH retrieve all rows of data from leo database with %.5fs seconds. ", time.Since(
       start).Seconds())
29         msg <- " "
30
31 }
```

Listing L.12: Function for LEO data retrieval. (retrieve_leo.go)

Listing L.12 shows the source code of NSPL data retrieval function that SELECT 32706 rows of LEO data from PostgreSQL database in **concurrent** manner. The function possess one parameter to allow *Gochannel* to be assigned for concurrent execution.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 14-26). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 28). The results will be tabulated and discussed in results and finding section.

### L.2.5.4 Main function

```
package main

import (
        "fmt"
        "database/sql"
        "time"

        _ "github.com/jinzhu/gorm/dialects/postgres"
        _ "github.com/lib/pq"
)

const (
        DB_USER     = "yinghua"
        DB_PASSWORD = "123"
        DB_NAME     = "fyp1"
)

var db *sql.DB

//==================================================
//function to check error and print error messages
//==================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//==================================================
// initialize connection with database
//==================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        sqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "Initialize database")
        db = sqldb

}

func goSelect(ch_company, ch_leo, ch_nspl chan string) {


        for i := 0; i < 3; i++ {
                select {
                case msg1 := <-ch_leo:
                        fmt.Println(msg1)
                case msg2 := <-ch_company:
                        fmt.Println(msg2)
                case msg3 := <-ch_nspl:
                        fmt.Println(msg3)

                }
        }
}

//============================================================
Retrieve all data from PostgreSQL database in concurrent manner
//============================================================
func concurrent_read() {
        // get the time before execution
        start := time.Now()

        initDB()

        // make three channel for three functions
        ch_company := make(chan string)
        ch_leo := make(chan string)
        ch_nspl := make(chan string)


        go retrieve_company_with_channel(ch_company);
        go retrieve_leo_with_channel(ch_leo);
        go retrieve_nspl_with_channel(ch_nspl);

        goSelect(ch_company, ch_leo, ch_nspl)

        // obtain the time after execution
        fmt.Printf("%.5fs seconds on retrieve all the data from database CONCURRENTLY. \n", time.Since(start).Seconds())
}

func main() {
        concurrent_read()
}
```

```
 86
 87  /**
 88
 89  yinghua@yinghua:~/gitRepo/go-read-psql/src/main$ go build *.go
 90  yinghua@yinghua:~/gitRepo/go-read-psql/src/main$ time go run *.go
 91  BEGIN retrieve data from nspl database.
 92  BEGIN retrieve data from companydata database.
 93  BEGIN retrieve data from leo database.
 94  FINISH retrieve all rows of data from leo database with 0.52910s seconds.
 95  FINISH retrieve all rows of data from nspl database with 14.52721s seconds.
 96  FINISH retrieve all rows of data from companydata database with 43.36509s seconds.
 97  43.36518s seconds on retrieve all the data from database CONCURRENTLY.
 98
 99  real    0m43.801s
100  user    0m59.145s
101  sys     0m1.631s
102
103  **/
```

Listing L.13: Main function for concurrent execution. (main.go)

Listing L.13 shows the source code for main function of Go programming language based PostgreSQL database retrieval program. The main function is where **a program start its execution**.

When the program is compiled and executed, main() will call concurrent_read() function to initiate data retrieval operation from three tables concurrently (refer row 84).

The program will first establish connection to PostgreSQL database with user, password and database name provided. Then, it will make three *Gochannels* ready to be parsed into each function (refer row 67-69). The functions shown in Listing L.10, L.11 and L.12 will be assigned into *Goroutines (A lightweight thread)* and parsed into the declared Gochannel to establish concurrent operation.

These function began to retrieve data from company table, LEO table and NSPL table concurrently. The Goselect is used to received the *Goroutines* and identify the state of each execution once the processed are finished. The total execution time of entire program will be display and print on terminal (refer row 80).

The result obtained will be tabulated and discussed.

## L.3 Rust program for CSV file data retrieval

### L.3.1 Rust Sequential program source codes.

```
extern crate csv;
use std::fs::File;

====================================
multiple producer, single consumer.
====================================
use std::sync::mpsc;

====================================
use time crate
====================================
extern crate time;
use time::PreciseTime;

const LEO_INDICATOR: &'static str = "subject";
const COMPANY_INDICATOR: &'static str = "company";
const NSPL_INDICATOR: &'static str = "postcode";
const LEO_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/subject-data/institution-subject-data.csv";
const COMPANY_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/company-data/company-data-full.csv";
const NSPL_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/postcode-data/UK-NSPL.csv";

=====================================
Function to retrieve data from CSV file
=====================================
fn retrieve_data(directory: &'static str, indicator: &'static str) -> u32 {

        println!("BEGIN retrieve data from {} files. ", indicator);

        // Parse the CSV reader and iterate over each record.
        let csv_file = File::open(directory).expect("Error open LEO file");

        let start = PreciseTime::now();
        let mut rdr = csv::Reader::from_reader(csv_file);

        for result in rdr.records() {

                let _record = result;
                //        println!("{:?}", record);
        }

                let end = PreciseTime::now();
                let duration = start.to(end);

                println!(
                "FINISH retrieve all rows of data from {} files with {} seconds.",
                indicator,
                duration
                );

        return 1;
}

==================================================================
Function that retrieve all rows of data from three raw CSV datasets.
==================================================================
fn sequential_read() {

        let start = PreciseTime::now();

        let _leo = retrieve_data(LEO_DIRECTORY, LEO_INDICATOR);
        let _company = retrieve_data(COMPANY_DIRECTORY, COMPANY_INDICATOR);
        let _nspl = retrieve_data(NSPL_DIRECTORY, NSPL_INDICATOR);

        let end = PreciseTime::now();
        let duration = start.to(end);

        println!(
        " {} seconds on retrieve all the data SEQUENTIALLY. ",
        duration
        );
}

fn main() {
        sequential_read();
}

/**

```

```
yinghua@yinghua:~/gitRepo/rs-read-csv$ cargo build
Compiling rs-read-csv v0.0.1 (file:///home/yinghua/gitRepo/rs-read-csv)
Finished dev [unoptimized + debuginfo] target(s) in 0.94 secs

yinghua@yinghua:~/gitRepo/rs-read-csv$ time cargo run
Finished dev [unoptimized + debuginfo] target(s) in 0.0 secs
Running ‘target/debug/rs-read-csv‘

BEGIN retrieve data from subject files.
FINISH retrieve all rows of data from subject files with 0.904617367 seconds.
BEGIN retrieve data from company files.
FINISH retrieve all rows of data from company files with 292.704881750 seconds.
BEGIN retrieve data from postcode files.
FINISH retrieve all rows of data from postcode files with 109.972792579 seconds.

403.582455002 seconds on retrieve all the data SEQUENTIALLY.

**/
```

Listing L.14: Rust sequential program source codes. (main.rs)

Listing L.14 shows the source code of Rust programming language based application that retrieve all rows of data from NSPL, company and LEO datasets in sequential manner. The program will open the each raw datasets stored in predefine directory and began to read all lines of records in the CSV file. Ultimately, the execution time will be display and recorded for comparison in result and discussion.

### L.3.2 Rust Concurrent program source codes.

```
extern crate csv;
use std::fs::File;

====================================
multiple producer, single consumer.
====================================
use std::sync::mpsc;

====================================
import for multithreading.
====================================
use std::thread;

====================================
use time crate
====================================
extern crate time;
use time::PreciseTime;

const LEO_INDICATOR: &'static str = "subject";
const COMPANY_INDICATOR: &'static str = "company";
const NSPL_INDICATOR: &'static str = "postcode";
const LEO_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/subject-data/institution-subject-data.csv";
const COMPANY_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/company-data/company-data-full.csv";
const NSPL_DIRECTORY: &'static str = "/home/yinghua/Documents/FYP1/FYP-data/postcode-data/UK-NSPL.csv";

=====================================
Function to retrieve data from CSV file
=====================================
fn retrieve_data(directory: &'static str, indicator: &'static str) -> u32 {

println!("BEGIN retrieve data from {} files. ", indicator);

// Parse the CSV reader and iterate over each record.
let csv_file = File::open(directory).expect("Error open LEO file");

let start = PreciseTime::now();
let mut rdr = csv::Reader::from_reader(csv_file);

for result in rdr.records() {

let _record = result;
//        println!("{:?}", record);
}

let end = PreciseTime::now();
let duration = start.to(end);

println!(
"FINISH retrieve all rows of data from {} files with {} seconds.",
indicator,
duration
);

return 1;
}

=================================================================================
Function that retrieve all rows of data from three raw CSV datasets in concurrent manner.
=================================================================================
fn concurrent_read() {

        let start = PreciseTime::now();

        // transmitter and receiver over the channel
        let (leo_tx, leo_rx) = mpsc::channel();
        let (company_tx, company_rx) = mpsc::channel();
        let (nspl_tx, nspl_rx) = mpsc::channel();

        thread::spawn(move || {
                let leo = retrieve_data(LEO_DIRECTORY, LEO_INDICATOR);
                leo_tx.send(leo).unwrap();
        });

        thread::spawn(move || {
                let company = retrieve_data(COMPANY_DIRECTORY, COMPANY_INDICATOR);
                company_tx.send(company).unwrap();
        });

        thread::spawn(move || {
                let nspl = retrieve_data(NSPL_DIRECTORY, NSPL_INDICATOR);
                nspl_tx.send(nspl).unwrap();
        });

        let leo_channel = leo_rx.recv().unwrap();
        let company_channel = company_rx.recv().unwrap();
        let nspl_channel = nspl_rx.recv().unwrap();

        let end = PreciseTime::now();
        let duration = start.to(end);

        println!(
        " {} seconds on retrieve all the data CONCURRENTLY. ",
        duration
        );

}

fn main() {
        concurrent_read();
}

/**

yinghua@yinghua:~/gitRepo/rs-read-csv$ cargo build
Compiling rs-read-csv v0.0.1 (file:///home/yinghua/gitRepo/rs-read-csv)
Finished dev [unoptimized + debuginfo] target(s) in 0.94 secs

yinghua@yinghua:~/gitRepo/rs-read-csv$ time cargo run

BEGIN retrieve data from subject files.
BEGIN retrieve data from postcode files.
BEGIN retrieve data from company files.
FINISH retrieve all rows of data from subject files with 1.038585794 seconds.
FINISH retrieve all rows of data from postcode files with 116.362977683 seconds.
FINISH retrieve all rows of data from company files with 314.530471492 seconds.

314.530967308 seconds on retrieve all the data CONCURRENTLY.

**/
```

Listing L.15: Rust concurrent program source codes. (main.rs)

Listing L.15 shows the source code of Rust programming language based application that retrieve all rows of data from NSPL, company and LEO datasets in concurrent manner. Three *threads* is created and each thread is assigned by a job (function) to complete the job. Three channel is declared used to receive the thread that completed the process and update the state of specific operations.

The program will open each raw datasets stored in predefine directory simultaneously and began to read all lines of records in the CSV file concurrently. Ultimately, the execution time will be display and recorded for comparison in result and discussion.

## L.4 Rust program for PostgreSQL database retrieval with ORM.

### L.4.1 NSPL struct

```

============================
// 36 columns 1754882 rows
============================
#[derive(Debug)]
struct Nspl {
        postcode1: String,
        postcode2: String,
        postcode3: String,
        date_introduce: String,
        user_type: i32,

        easting: Option<i32>,
        northing: Option<i32>,
        position_quality: i32,
        countycode: Option<String>,
        countyname: Option<String>,

        county_lac: Option<String>,
        county_lan: Option<String>,
        ward_code: Option<String>,
        ward_name: Option<String>,
        country_code: Option<String>,

        country_name: Option<String>,
        region_code: Option<String>,
        region_name: Option<String>,
        par_cons_code: Option<String>,
        par_cons_name: Option<String>,

        eerc: Option<String>,
        eern: Option<String>,
        pctc: Option<String>,
        pctn: Option<String>,
        isoac: Option<String>,

        isoan: Option<String>,
        msoac: Option<String>,
        msoan: Option<String>,
        oacc: Option<String>,
        oacn: Option<String>,

        longitude: f32,
        latitude: f32,
        spatial_accuracy: Option<String>,
        last_upload: NaiveDate,
        location: Option<String>,
        socrataid: i32,
}
```

Listing L.16: Source code for NSPL struct. (nspl.rs)

## L.4.2 Company struct

```
======================
3595702 rows 55 columns
======================
#[derive(Debug)]
struct Company {
        name: Option<String>,
        number: String,
        careof: Option<String>,
        po_box: Option<String>,
        address_line1: Option<String>,

        address_line2: Option<String>,
        post_town: Option<String>,
        county: Option<String>,
        country: Option<String>,
        post_code: Option<String>,

        company_category: String,
        company_status: String,
        county_of_origin: String,
        dissolution_date: Option<NaiveDate>,
        incorporation_date: Option<NaiveDate>,

        accounting_ref_day: Option<i32>,
        accounting_ref_month: Option<i32>,
        account_next_due_date: Option<NaiveDate>,
        account_last_made_update: Option<NaiveDate>,
        account_category: Option<String>,

        return_next_due_date: Option<NaiveDate>,
        return_last_made_update: Option<NaiveDate>,
        num_mort_changes: Option<i32>,
        num_mort_out_standing: Option<i32>,
        num_mort_part_satisfied: Option<i32>,

        num_mort_satisfied: Option<i32>,
        siccode1: Option<String>,
        siccode2: Option<String>,
        siccode3: Option<String>,
        siccode4: Option<String>,

        num_gen_partners: i32,
        num_lim_partners: i32,
        uri: String,
        pn1_condate: Option<NaiveDate>,
        pn1_companydate: Option<String>,

        pn2_condate: Option<NaiveDate>,
        pn2_companydate: Option<String>,
        pn3_condate: Option<NaiveDate>,
        pn3_companydate: Option<String>,
        pn4_condate: Option<NaiveDate>,

        pn4_companydate: Option<String>,
        pn5_condate: Option<NaiveDate>,
        pn5_companydate: Option<String>,
        pn6_condate: Option<NaiveDate>,
        pn6_companydate: Option<String>,

        pn7_condate: Option<NaiveDate>,
        pn7_companydate: Option<String>,
        pn8_condate: Option<NaiveDate>,
        pn8_companydate: Option<String>,
        pn9_condate: Option<NaiveDate>,

        pn9_companyname: Option<String>,
        pn10_condate: Option<NaiveDate>,
        pn10_companydate: Option<String>,
        conf_stmt_next_due_date: Option<NaiveDate>,
        conf_stmt_last_made_update: Option<NaiveDate>,
}
```

Listing L.17: Source code for Company struct. (company.rs)

### L.4.3 LEO struct

```
======================
32706 rows 22 columns
======================
#[derive(Debug)]
struct Leo {
        ukprn: i32,
        provider_name: String,
        region: String,
        subject: String,
        sex: String,

        year_after_graduation: String,
        grads: Option<String>,
        unmatched: Option<String>,
        matched: Option<String>,
        activity_not_captured: Option<String>,

        no_sust_dest: Option<String>,
        sus_temp_only: Option<String>,
        sus_temp: Option<String>,
        sus_tempfs_or_both: Option<String>,
        earnings_include: Option<String>,

        lower_ann_earn: Option<String>,
        median_ann_earn: Option<String>,
        upper_ann_earn: Option<String>,
        polar_gr_pone: Option<String>,
        polar_gr_pone_included: Option<String>,

        pr_att_band: Option<String>,
        pr_att_included: Option<String>,
}
```

Listing L.18: Source code for LEO struct. (leo.rs)

Listing L.16, L.17 and L.18 shows the source code of NSPL, Company and LEO struct created in Rust ORM program. Table below explain the specification of types conversion and choice data type used in these struct.

| **Data type in PostgreSQL** | **Data type in Rust** | **Specification** |
|---|---|---|
| INTEGER(10) | i32 | store signed 32 bits integer. |
| BIGINT | i64 | store signed 64 bits integer. |
| VARCHAR | String | store alphanumeric and alphabets. |
| INT | Option¡i32¿ | store NULL values or 32 bits integer. |
| VARCHAR | Option¡String¿ | store NULL values or string. |
| REAL | f32 | store signed 32 bit decimal. f |

Table L.2: Data type specification in Rust programming language

It is essential to understand and declared valid data types for object relational mapping to prevent type errors and data corruption. The attributes of each struct are declared and defined with correct data types for data conversion.

## L.4.4 Rust program source code

### L.4.4.1 Company data retrieval function

```
============================================================================
Retrieving 3595702 rows of data from PostgreSQL database
============================================================================
pub fn retrieve_company() {


        let db_url = "postgresql://yinghua:123@localhost:5432/fyp1";
        let conn = Connection::connect(db_url, TlsMode::None).unwrap();

        println!("BEGIN retrieve data from companydata database. ");
        let start = PreciseTime::now();


        for rows in &conn.query("SELECT * FROM companydata", &[]).unwrap() {
                let _company = Company {
                        name: rows.get(0),
                        number: rows.get(1),
                        careof: rows.get(2),
                        po_box: rows.get(3),
                        address_line1: rows.get(4),

                        address_line2: rows.get(5),
                        post_town: rows.get(6),
                        county: rows.get(7),
                        country: rows.get(8),
                        post_code: rows.get(9),

                        company_category: rows.get(10),
                        company_status: rows.get(11),
                        county_of_origin: rows.get(12),
                        dissolution_date: rows.get(13),
                        incorporation_date: rows.get(14),

                        accounting_ref_day: rows.get(15),
                        accounting_ref_month: rows.get(16),
                        account_next_due_date: rows.get(17),
                        account_last_made_update: rows.get(18),
                        account_category: rows.get(19),

                        return_next_due_date: rows.get(20),
                        return_last_made_update: rows.get(21),
                        num_mort_changes: rows.get(22),
                        num_mort_out_standing: rows.get(23),
                        num_mort_part_satisfied: rows.get(24),

                        num_mort_satisfied: rows.get(25),
                        siccode1: rows.get(26),
                        siccode2: rows.get(27),
                        siccode3: rows.get(28),
                        siccode4: rows.get(29),

                        num_gen_partners: rows.get(30),
                        num_lim_partners: rows.get(31),
                        uri: rows.get(32),
                        pn1_condate: rows.get(33),
                        pn1_companydate: rows.get(34),

                        pn2_condate: rows.get(35),
                        pn2_companydate: rows.get(36),
                        pn3_condate: rows.get(37),
                        pn3_companydate: rows.get(38),
                        pn4_condate: rows.get(39),

                        pn4_companydate: rows.get(40),
                        pn5_condate: rows.get(41),
                        pn5_companydate: rows.get(42),
                        pn6_condate: rows.get(43),
                        pn6_companydate: rows.get(44),

                        pn7_condate: rows.get(45),
                        pn7_companydate: rows.get(46),
                        pn8_condate: rows.get(47),
                        pn8_companydate: rows.get(48),
                        pn9_condate: rows.get(49),

                        pn9_companyname: rows.get(50),
                        pn10_condate: rows.get(51),
                        pn10_companydate: rows.get(52),
                        conf_stmt_next_due_date: rows.get(53),
                        conf_stmt_last_made_update: rows.get(54),
                };
82
83         //        println!("{:?}", company);
84         }
85
86         let end = PreciseTime::now();
87         let duration = start.to(end);
88
89         println!(
90         "FINISH retrieve all rows of data from companydata database with {} seconds.",
91         duration
92         );
93 }
```

Listing L.19: Function for company data retrieval. (company.rs)

Listing L.19 shows the source code of company data retrieval function that SELECT 3595702 rows of company data from PostgreSQL database in **concurrent** manner. The function is used for both sequential and concurrent execution for data retrieval in Rust program.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 14-83). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 89). The results will be tabulated and discussed in results and finding section.

### L.4.4.2 NSPL data retrieval function

```
=========================================================
Retrieving 1754882 rows of data from PostgreSQL database
=========================================================
pub fn retrieve_nspl() {

        let db_url = "postgresql://yinghua:123@localhost:5432/fyp1";
        let conn = Connection::connect(db_url, TlsMode::None).unwrap();

        println!("BEGIN retrieve data from nspl database. ");
        let start = PreciseTime::now();

        for rows in &conn.query("SELECT * FROM nspl", &[]).unwrap() {

                let _postcode = Nspl {
                        postcode1: rows.get(0),
                        postcode2: rows.get(1),
                        postcode3: rows.get(2),
                        date_introduce: rows.get(3),
                        user_type: rows.get(4),

                        easting: rows.get(5),
                        northing: rows.get(6),
                        position_quality: rows.get(7),
                        countycode: rows.get(8),
                        countyname: rows.get(9),

                        county_lac: rows.get(10),
                        county_lan: rows.get(11),
                        ward_code: rows.get(12),
                        ward_name: rows.get(13),
                        country_code: rows.get(14),

                        country_name: rows.get(15),
                        region_code: rows.get(16),
                        region_name: rows.get(17),
                        par_cons_code: rows.get(18),
                        par_cons_name: rows.get(19),

                        eerc: rows.get(20),
                        eern: rows.get(21),
                        pctc: rows.get(22),
                        pctn: rows.get(23),
                        isoac: rows.get(24),

                        isoan: rows.get(25),
                        msoac: rows.get(26),
                        msoan: rows.get(27),
                        oacc: rows.get(28),
                        oacn: rows.get(29),

                        longitude: rows.get(30),
                        latitude: rows.get(31),
                        spatial_accuracy: rows.get(32),
                        last_upload: rows.get(33),
                        location: rows.get(34),
                        socrataid: rows.get(35),
                };

        //        println!("{:?}", postcode);

        }

        let end = PreciseTime::now();
        let duration = start.to(end);
        println!(
        "FINISH retrieve all rows of data from nspl database with {} seconds.",
        duration
        );

}
```

Listing L.20: Function for NSPL data retrieval. (nspl.rs)

Listing L.20 shows the source code of company data retrieval function that SELECT 1754882 rows of NSPL data from PostgreSQL database in **concurrent** manner. The function is used for both sequential and concurrent execution for data retrieval in Rust program.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 15-56). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 65). The results will be tabulated and discussed in results and finding section.

### L.4.4.3 LEO data retrieval function

```
 1 =============================================================================
 2 Retrieving 32706 rows of data from PostgreSQL database in sequential manner
 3 =============================================================================
 4 pub fn retrieve_leo() {
 5 
 6         let db_url = "postgresql://yinghua:123@localhost:5432/fyp1";
 7         let conn = Connection::connect(db_url, TlsMode::None).unwrap();
 8 
 9         println!("BEGIN retrieve data from leo database. ");
10         let start = PreciseTime::now();
11 
12         for rows in &conn.query("SELECT * FROM leo", &[]).unwrap() {
13 
14                 let _subject = Leo {
15                         ukprn: rows.get(0),
16                         provider_name: rows.get(1),
17                         region: rows.get(2),
18                         subject: rows.get(3),
19                         sex: rows.get(4),
20 
21                         year_after_graduation: rows.get(5),
22                         grads: rows.get(6),
23                         unmatched: rows.get(7),
24                         matched: rows.get(8),
25                         activity_not_captured: rows.get(9),
26 
27                         no_sust_dest: rows.get(10),
28                         sus_temp_only: rows.get(11),
29                         sus_temp: rows.get(12),
30                         sus_tempfs_or_both: rows.get(13),
31                         earnings_include: rows.get(14),
32 
33                         lower_ann_earn: rows.get(15),
34                         median_ann_earn: rows.get(16),
35                         upper_ann_earn: rows.get(17),
36                         polar_gr_pone: rows.get(18),
37                         polar_gr_pone_included: rows.get(19),
38 
39                         pr_att_band: rows.get(20),
40                         pr_att_included: rows.get(21),
41                 };
42 
43                 //        println!("{:?}", subject);
44 
45         }
46 
47         let end = PreciseTime::now();
48         let duration = start.to(end);
49         println!(
50         "FINISH retrieve all rows of data from leo database with {} seconds.",
51         duration
52         );
53 }
```

Listing L.21: Function for LEO data retrieval. (leo.rs)

Listing L.21 shows the source code of company data retrieval function that SELECT 32706 rows of LEO data from PostgreSQL database in **concurrent** manner. The function is used for both sequential and concurrent execution for data retrieval in Rust program.

Other than that, the function will retrieve all rows of data and map into the object declared (refer row 15-40). The execution duration and outcomes will be display on the terminal to indicate the process is completed (refer row 49). The results will be tabulated and discussed in results and finding section.

### L.4.4.4 Main function

```
extern crate postgres;

================
use time crate
================
extern crate time;
extern crate chrono;
use time::PreciseTime;

====================================
multiple producer, single consumer.
====================================
use std::sync::mpsc;

====================================
import for multithreading execution
====================================
use std::thread;

mod company;
mod leo;
mod nspl;

===========================================================================
Function that retrieve all row of data from PostgreSQL in sequential manner
===========================================================================
fn sequential_read() {

        let start = PreciseTime::now();

        company::retrieve_company();
        leo::retrieve_leo();
        nspl::retrieve_nspl();

        let end = PreciseTime::now();
        let duration = start.to(end);

        println!(
        " {} seconds on retrieve all the data SEQUENTIALLY. ",
        duration
        );

}

===========================================================================
Function that retrieve all row of data from PostgreSQL in concurrent manner
===========================================================================
fn concurrent_read() {

        let start = PreciseTime::now();

        // transmitter and receiver over the channel
        let (leo_tx, leo_rx) = mpsc::channel();
        let (company_tx, company_rx) = mpsc::channel();
        let (nspl_tx, nspl_rx) = mpsc::channel();

        thread::spawn(move || {

        let company = company::retrieve_company();
                company_tx.send(company).unwrap();
        });

        thread::spawn(move || {

        let leo = leo::retrieve_leo();
                leo_tx.send(leo).unwrap();
        });

        thread::spawn(move || {

        let nspl = nspl::retrieve_nspl();
                nspl_tx.send(nspl).unwrap();
        });

        let _leo_channel = leo_rx.recv().unwrap();
        let _company_channel = company_rx.recv().unwrap();
        let _nspl_channel = nspl_rx.recv().unwrap();

        let end = PreciseTime::now();
        let duration = start.to(end);

        println!(
        " {} seconds on retrieve all the data CONCURRENTLY. ",
        duration
        );
}
```

```
 87
 88
 89  fn main() {
 90          concurrent_read();
 91          sequential_read();
 92  }
 93
 94  /**
 95
 96  yinghua@yinghua:~/gitRepo/rs-read-psql$ cargo build
 97  Finished dev [unoptimized + debuginfo] target(s) in 0.0 secs
 98  yinghua@yinghua:~/gitRepo/rs-read-psql$ time cargo run
 99  Compiling rs-read-psql v0.1.0 (file:///home/yinghua/gitRepo/rs-read-psql)
100
101  BEGIN retrieve data from nspl database.
102  BEGIN retrieve data from companydata database.
103  BEGIN retrieve data from leo database.
104  FINISH retrieve all rows of data from leo database with 0.789323246 seconds.
105  FINISH retrieve all rows of data from nspl database with 65.702471599 seconds.
106  FINISH retrieve all rows of data from companydata database with 181.387234079 seconds.
107  181.389403179 seconds on retrieve all the data CONCURRENTLY.
108
109  BEGIN retrieve data from companydata database.
110  FINISH retrieve all rows of data from companydata database with 172.584919465 seconds.
111  BEGIN retrieve data from leo database.
112  FINISH retrieve all rows of data from leo database with 0.720544494 seconds.
113  BEGIN retrieve data from nspl database.
114  FINISH retrieve all rows of data from nspl database with 60.442268738 seconds.
115  233.752923612 seconds on retrieve all the data SEQUENTIALLY.
116
117
118  **/
```

Listing L.22: Main function for sequential execution. (main.rs)

Listing L.22 shows the source code for main function of Rust programming language based PostgreSQL database retrieval program. The main function is where **a program start its execution**.

When the program is compiled and executed, main() will call both **concurrent_read()** and **sequential_read()**function to initiate data retrieval operation from three tables concurrently and sequentially (refer 90-92).

The concurrent function will first establish connection to PostgreSQL database with user, password and database name provided. Then, it will make three channels with *multiple producer and single consumer* that ready to be parsed into each function (refer row 52-55). The three function will be assigned into each *thread* and parsed into the declared channel to establish concurrent operation. The entire execution of this function will be display and print on terminal (refer row 82-84).

The sequential function will establish connection to PostgreSQL database with user, password and database name provided once again. Then, the three function began to retrieve data from company table, LEO table and NSPL table sequentially (refer row 31-33). The entire execution of this function will be display and print on terminal (refer row 38-41).

The result obtained will be tabulated, compared and discussed.

# Appendix M

# Data Definition Language (DDL).

## M.1 PL/pgSQL's DDL scripts for Postcode Normalized Table Creation.

```sql

-- File: 02_yinghua_normalized_NSPL_DDL.sql
-- Date: Sat Dec 6 16:02 MYT 2017
-- Author: Chai Ying Hua
-- Version: 1.0
-- Database: psql (PostgreSQL) 9.5.10
-- =========================================================
--  1. Drop table in Reverse order.
--  2. Create table in Proper order.
--  3. Verify whether all tables and sequences are created.
-- =========================================================

-- DROP TABLE IN REVERSE ORDER
DROP TABLE postcode_greek_coordinate            CASCADE;
DROP TABLE postcode_output_area_classification  CASCADE;
DROP TABLE postcode_middle_super_output_area    CASCADE;
DROP TABLE postcode_lower_super_output_area     CASCADE;
DROP TABLE postcode_primary_care_trust          CASCADE;
DROP TABLE postcode_euro_electoral_region       CASCADE;
DROP TABLE postcode_parliament_constituency     CASCADE;
DROP TABLE postcode_region                      CASCADE;
DROP TABLE postcode_country                     CASCADE;
DROP TABLE postcode_ward                        CASCADE;
DROP TABLE postcode_local_authority_county      CASCADE;
DROP TABLE postcode_county                      CASCADE;
DROP TABLE postcode_cartesian_coordinate        CASCADE;
DROP TABLE postcode_detail                      CASCADE;
DROP TABLE postcode                             CASCADE;

-- DROP SEQUENCE IN PROPER ORDER
DROP SEQUENCE seq_pos_detail_id         CASCADE;
DROP SEQUENCE seq_cart_coordinate_id    CASCADE;
DROP SEQUENCE seq_county_id             CASCADE;
DROP SEQUENCE seq_lac_id                CASCADE;
DROP SEQUENCE seq_ward_id               CASCADE;
DROP SEQUENCE seq_country_id            CASCADE;
DROP SEQUENCE seq_region_id             CASCADE;
DROP SEQUENCE seq_par_cons_id           CASCADE;
DROP SEQUENCE seq_eer_id                CASCADE;
DROP SEQUENCE seq_pct_id                CASCADE;
DROP SEQUENCE seq_lsoa_id               CASCADE;
DROP SEQUENCE seq_msoa_id               CASCADE;
DROP SEQUENCE seq_oac_id                CASCADE;
DROP SEQUENCE seq_greek_coordinate_id   CASCADE;
DROP SEQUENCE seq_pos_temp_id           CASCADE;

-- CREATE SEQUENCE IN REVERSE ORDER
CREATE SEQUENCE seq_pos_temp_id         MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_greek_coordinate_id MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_oac_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_msoa_id             MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_lsoa_id             MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_pct_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_eer_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_par_cons_id         MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_region_id           MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_country_id          MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_ward_id             MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_lac_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_county_id           MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_cart_coordinate_id  MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_pos_detail_id       MINVALUE 1 INCREMENT 1;

-- CREATE TABLE IN PROPER ORDER
create table postcode_greek_coordinate (
        pos_greek_coordinate_id         INT DEFAULT NEXTVAL (’seq_greek_coordinate_id’),
        pos_longitude                   REAL NOT NULL,
        pos_latitude                    REAL NOT NULL,
        PRIMARY KEY (pos_greek_coordinate_id)
);

create table postcode_output_area_classification (
        pos_oac_id                      INT DEFAULT NEXTVAL (’seq_oac_id’),
        pos_oac_code                    VARCHAR(5) NULL DEFAULT ’---’,
        pos_oac_name                    VARCHAR(50) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_oac_id)

);

create table postcode_middle_super_output_area (
        pos_msoa_id                     INT DEFAULT NEXTVAL (’seq_msoa_id’),
        pos_msoa_code                   VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_msoa_name                   VARCHAR(50) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_msoa_id)
);

create table postcode_lower_super_output_area (
        pos_lsoa_id                     INT DEFAULT NEXTVAL (’seq_lsoa_id’),
        pos_lsoa_code                   VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_lsoa_name                   VARCHAR(50) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_lsoa_id)

);

create table postcode_primary_care_trust (
        pos_pct_id                      INT DEFAULT NEXTVAL (’seq_pct_id’),
        pos_pct_code                    VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_pct_name                    VARCHAR(70) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_pct_id)
);

create table postcode_euro_electoral_region (
        pos_eer_id                      INT DEFAULT NEXTVAL (’seq_eer_id’),
        pos_eer_code                    VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_eer_name                    VARCHAR(30) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_eer_id)
);

create table postcode_parliament_constituency (
        pos_par_cons_id                 INT DEFAULT NEXTVAL (’seq_par_cons_id’),
        pos_par_cons_code               VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_par_cons_name               VARCHAR(75) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_par_cons_id)
);

create table postcode_region (
        pos_region_id                   INT DEFAULT NEXTVAL (’seq_region_id’),
        pos_region_code                 VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_region_name                 VARCHAR(50) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_region_id)
);

create table postcode_country (
        pos_country_id                  INT DEFAULT NEXTVAL (’seq_country_id’),
        pos_country_code                VARCHAR(30) NULL DEFAULT ’Undefined’,
        pos_country_name                VARCHAR(30) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_country_id)
);

create table postcode_ward (
        pos_ward_id                     INT DEFAULT NEXTVAL (’seq_ward_id’),
        pos_ward_code                   VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_ward_name                   VARCHAR(75) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_ward_id)
);

create table postcode_local_authority_county (
        pos_lac_id                      INT DEFAULT NEXTVAL (’seq_lac_id’),
        pos_lac_code                    VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_lac_name                    VARCHAR(75) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_lac_id)
);

create table postcode_county (
        pos_county_id                   INT DEFAULT NEXTVAL (’seq_county_id’),
        pos_county_code                 VARCHAR(15) NULL DEFAULT ’Undefined’,
        pos_county_name                 VARCHAR(75) NULL DEFAULT ’Undefined’,
        PRIMARY KEY (pos_county_id)
);

create table postcode_cartesian_coordinate (
        pos_cart_coordinate_id          INT DEFAULT NEXTVAL (’seq_cart_coordinate_id’),
        pos_easting                     INT NULL DEFAULT 0,
        pos_northing                    INT NULL DEFAULT 0,
        PRIMARY KEY (pos_cart_coordinate_id)
);

create table postcode_detail (
        pos_detail_id                   BIGINT DEFAULT NEXTVAL (’seq_pos_detail_id’),
        pos1                            VARCHAR(15) NOT NULL,
        pos2                            VARCHAR(15) NOT NULL,
        pos3                            VARCHAR(15) NOT NULL,
        pos_date_introduce              VARCHAR(10) NOT NULL,
        pos_usertype                    INT         NOT NULL,
        pos_cart_coordinate_id          INT         NOT NULL,
        position_quality                INT         NOT NULL,
        pos_spatial_accuracy            VARCHAR(30) NULL DEFAULT ’Undefined’,
        pos_location                    VARCHAR(50) NULL DEFAULT ’Undefined’,
        pos_socrataid                   INT         NOT NULL,
        pos_last_upload                 DATE        NOT NULL,
        PRIMARY KEY (pos_detail_id),
        FOREIGN KEY (pos_cart_coordinate_id) REFERENCES postcode_cartesian_coordinate (
    pos_cart_coordinate_id)
);

create table postcode (
        pos_detail_id                   INT REFERENCES postcode_detail (pos_detail_id),
        pos_county_id                   INT REFERENCES postcode_county (pos_county_id),
        pos_lac_id                      INT REFERENCES postcode_local_authority_county (pos_lac_id),
        pos_ward_id                     INT REFERENCES postcode_ward (pos_ward_id),
        pos_country_id                  INT REFERENCES postcode_country (pos_country_id),
        pos_region_id                   INT REFERENCES postcode_region (pos_region_id),
        pos_par_cons_id                 INT REFERENCES postcode_parliament_constituency (pos_par_cons_id),
        pos_eer_id                      INT REFERENCES postcode_euro_electoral_region (pos_eer_id),
        pos_pct_id                      INT REFERENCES postcode_primary_care_trust (pos_pct_id),
        pos_lsoa_id                     INT REFERENCES postcode_lower_super_output_area (pos_lsoa_id),
        pos_msoa_id                     INT REFERENCES postcode_middle_super_output_area (pos_msoa_id),
        pos_oac_id                      INT REFERENCES postcode_output_area_classification (pos_oac_id),
        pos_greek_coordinate_id         INT REFERENCES postcode_greek_coordinate (pos_greek_coordinate_id)
);

-- CHECK WHETHER ALL TABLE AND SEQUENCE ARE CREATED
\d+
```

Listing M.1: PL/pgSQL’s DDL scripts for Postcode Normalized Table Creation.

Listing M.1 show the PL/pgSQL’s Data Definition Language (DDL) scripts to create Postcode Normalized table based on database design shown in Section 3.6.2.3. All the table are defined with PRIMARY KEY (PK) and establish referencial integrity relationship amongs entity to form a good relational database design. Moreover, the data types are defined correctly with sufficient memory provided on each columns.

## M.2 PL/pgSQL's DDL scripts for Company Normalized Table Creation.

```

-- FILE: 02_yinghua_normalized_company_DDL.sql
-- DATE: Mon Jan 7 17:00 MYT 2018
-- AUTHOR: Chai Ying Hua
-- VERSION: 1.0
-- DATABASE: psql (PostgreSQL) 9.5.10
-- DESCRIPTION:
-- =======================================================
--
--      1. Drop previous created table in proper order.
--      2. Create sequence in proper order.
--      3. Drop sequence in reverse order.
--      4. Create table in reverse order for main table to reference foreign key
--      5. Check all the tables.
-- ========================================================

-- DROP TABLE IN REVERSE ORDER
DROP TABLE company_uri                  CASCADE;
DROP TABLE company_partnership          CASCADE;
DROP TABLE company_siccodes             CASCADE;
DROP TABLE company_mortgages            CASCADE;
DROP TABLE company_returns              CASCADE;
DROP TABLE company_account_category     CASCADE;
DROP TABLE company_account              CASCADE;
DROP TABLE company_previousname         CASCADE;
DROP TABLE company_conf_stmt            CASCADE;
DROP TABLE company_status               CASCADE;
DROP TABLE company_category             CASCADE;
DROP TABLE company_detail               CASCADE;
DROP TABLE company                      CASCADE;


-- DROP SEQUENCE IN PROPER ORDER
DROP SEQUENCE seq_detail_id;
DROP SEQUENCE seq_category_id;
DROP SEQUENCE seq_status_id;
DROP SEQUENCE seq_conf_stmt_id;
DROP SEQUENCE seq_pn_id;
DROP SEQUENCE seq_acc_id;
DROP SEQUENCE seq_acc_category_id;
DROP SEQUENCE seq_return_id;
DROP SEQUENCE seq_mort_id;
DROP SEQUENCE seq_sic_id;
DROP SEQUENCE seq_partnership_id;
DROP SEQUENCE seq_uri_id;

-- DROP SEQUENCE IN PROPER ORDER
CREATE SEQUENCE seq_uri_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_partnership_id      MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_sic_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_mort_id             MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_return_id           MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_acc_category_id     MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_acc_id              MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_pn_id               MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_conf_stmt_id        MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_status_id           MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_category_id         MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_detail_id           MINVALUE 1 INCREMENT 1;


-- CREATE TABLE IN PROPER ORDER
CREATE TABLE company_uri (
        com_uri_id                      INT DEFAULT NEXTVAL ('seq_uri_id') PRIMARY KEY,
        com_uri                         VARCHAR(47) NOT NULL
);

CREATE TABLE company_partnership (
        com_partnership_id              INT DEFAULT NEXTVAL ('seq_partnership_id') PRIMARY KEY,
        com_num_genpartners             INT NOT NULL,
        com_num_limpartners             INT NOT NULL
);

CREATE TABLE company_siccodes (
        com_sic_id                      INT DEFAULT NEXTVAL ('seq_sic_id') PRIMARY KEY,
        com_siccode1                    VARCHAR(170) NOT NULL,
        com_siccode2                    VARCHAR(170) NOT NULL,
        com_siccode3                    VARCHAR(170) NOT NULL,
        com_siccode4                    VARCHAR(170) NOT NULL
);

CREATE TABLE company_mortgages (
        com_mort_id                     INT DEFAULT NEXTVAL ('seq_mort_id') PRIMARY KEY,
        com_num_mortchanges             INT NOT NULL,
        com_num_mortoutstanding         INT NOT NULL,
        com_num_mortpartsatisfied       INT NOT NULL,
        com_num_mortsatisfied           INT NOT NULL
);

CREATE TABLE company_returns (
        com_return_id                   INT DEFAULT NEXTVAL ('seq_return_id') PRIMARY KEY,
        com_return_nextduedate          VARCHAR(50) NULL DEFAULT NULL,
        com_return_lastmadeupdate       VARCHAR(50) NULL DEFAULT NULL
);

CREATE TABLE company_account_category (
        com_acc_category_id             INT DEFAULT NEXTVAL ('seq_acc_category_id') PRIMARY KEY,
        com_acc_category                VARCHAR(100) NULL DEFAULT 'Undefined'
);

CREATE TABLE company_account (
        com_acc_id                      INT DEFAULT NEXTVAL ('seq_acc_id') PRIMARY KEY,
        com_acc_refday                  INT NULL DEFAULT 0,
        com_acc_refmonth                INT NULL DEFAULT 0,
        com_acc_nextduedate             VARCHAR(50) NULL DEFAULT NULL,
        com_acc_lastmadeupdate          VARCHAR(50) NULL DEFAULT NULL,
        com_acc_category_id             INT REFERENCES company_account_category (com_acc_category_id)
);

CREATE TABLE company_previousname (
        com_pn_id                       INT DEFAULT NEXTVAL ('seq_pn_id') PRIMARY KEY,
        com_pn1_condate                 VARCHAR(20) NOT NULL,
        com_pn1_companyname             VARCHAR(160) NOT NULL,
        com_pn2_condate                 VARCHAR(20) NOT NULL,
        com_pn2_companyname             VARCHAR(160) NOT NULL,
        com_pn3_condate                 VARCHAR(20) NOT NULL,
        com_pn3_companyname             VARCHAR(160) NOT NULL,
        com_pn4_condate                 VARCHAR(20) NOT NULL,
        com_pn4_companyname             VARCHAR(160) NOT NULL,
        com_pn5_condate                 VARCHAR(20) NOT NULL,
        com_pn5_companyname             VARCHAR(160) NOT NULL,
        com_pn6_condate                 VARCHAR(20) NOT NULL,
        com_pn6_companyname             VARCHAR(160) NOT NULL,
        com_pn7_condate                 VARCHAR(20) NOT NULL,
        com_pn7_companyname             VARCHAR(160) NOT NULL,
        com_pn8_condate                 VARCHAR(20) NOT NULL,
        com_pn8_companyname             VARCHAR(160) NOT NULL,
        com_pn9_condate                 VARCHAR(20) NOT NULL,
        com_pn9_companyname             VARCHAR(160) NOT NULL,
        com_pn10_condate                VARCHAR(20) NOT NULL,
        com_pn10_companyname            VARCHAR(160) NOT NULL

);

CREATE TABLE company_conf_stmt (
        com_conf_stmt_id                INT DEFAULT NEXTVAL ('seq_conf_stmt_id') PRIMARY KEY,
        com_conf_stmt_nextduedate       VARCHAR(50) NULL DEFAULT NULL,
        com_conf_stmt_lastmadeupdate    VARCHAR(50) NULL DEFAULT NULL
);

CREATE TABLE company_status (
        com_status_id                   INT DEFAULT NEXTVAL ('seq_status_id') PRIMARY KEY,
        com_status                      VARCHAR(70) NOT NULL DEFAULT 'Undefined'
);

CREATE TABLE company_category (
        com_category_id                 INT DEFAULT NEXTVAL ('seq_category_id') PRIMARY KEY,
        com_category                    VARCHAR(100) NOT NULL DEFAULT 'Undefined'
);

CREATE TABLE company_detail (
        com_detail_id                   INT DEFAULT NEXTVAL ('seq_detail_id') PRIMARY KEY,
        com_name                        VARCHAR(160) NULL DEFAULT 'Undefined',
        com_number                      VARCHAR(10)  NOT NULL,
        com_category_id                 INT          REFERENCES company_category (com_category_id),
        com_status_id                   INT          REFERENCES company_status (com_status_id),
);

CREATE TABLE company (
        com_detail_id                   INT REFERENCES company_detail (com_detail_id),
        com_dissolutiondate             VARCHAR(20) NOT NULL,
        com_incorporationdate           VARCHAR(20) NOT NULL,
        com_countryoforigin             VARCHAR(50) NOT NULL DEFAULT 'Undefined',
        com_careof                      VARCHAR(100) NULL DEFAULT 'Undefined',
        com_pobox                       VARCHAR(10)  NULL DEFAULT 'Undefined',
        com_addressline1                VARCHAR(300) NULL DEFAULT 'Undefined',
        com_addressline2                VARCHAR(300) NULL DEFAULT 'Undefined',
        com_posttown                    VARCHAR(50)  NULL DEFAULT 'Undefined',
        com_county                       VARCHAR(50)  NUListing M.1 show the PL/pgSQL's Data Definition
     Language (DDL) scripts to create Postcode Normalized table based on database design shown in Section
     3.6.2.3. All the table are defined with PRIMARY KEY (PK) and establish referencial integrity
     relationship amongs entity to form a good relational database design. Moreover, the data types are
     defined correctly with sufficient memory provided on each columns.

        \pagebreakLL DEFAULT 'Undefined',
        com_country                      VARCHAR(50)  NULL DEFAULT 'Undefined',
        com_postcode                     VARCHAR(20)  NULL DEFAULT 'Undefined',
        com_acc_id                       INT REFERENCES company_account (com_acc_id),
        com_return_id                    INT REFERENCES company_returns (com_return_id),
        com_mort_id                      INT REFERENCES company_mortgages (com_mort_id),
        com_sic_id                       INT REFERENCES company_siccodes (com_sic_id),
        com_partnership_id               INT REFERENCES company_partnership (com_partnership_id),
        com_uri_id                       INT REFERENCES company_uri (com_uri_id),
        com_pn_id                        INT REFERENCES company_previousname (com_pn_id),
        com_conf_stmt_id                 INT REFERENCES company_conf_stmt (com_conf_stmt_id)
    );
```

Listing M.2: PL/pgSQL's DDL scripts for Company Normalized Table Creation.

Listing M.2 show the PL/pgSQL's Data Definition Language (DDL) scripts to create Company Normalized table based on database design shown in Section 3.6.2.2. All the table are defined with PRIMARY KEY (PK) and establish referencial integrity relationship amongs entity to form a good relational database design. Moreover, the data types are defined correctly with sufficient memory provided on each columns.

## M.3 PL/pgSQL's DDL scripts for Education Normalized Table Creation.

```

-- File: 02_yinghua_create_normalized_leo_table.sql
-- Date: Fri Dec 5 14:04 MYT 2017
-- Author: Chai Ying Hua
-- Version: 1.0
-- Database: psql (PostgreSQL) 9.5.10
-- ======================================
--
--      1. Drop previous created table in proper order.
--      2. Create sequence in proper order.
--      3. Drop sequence in reverse order.
--      4. Create table in reverse order for main table to reference foreign key
--      5. Check all the tables.
-- ======================================

-- DROP TABLE IN PROPER ORDER


drop table leo_prior_attainment CASCADE;
drop table leo_polar CASCADE;
drop table leo_earning CASCADE;
drop table leo_sustain_employment CASCADE;
drop table leo_uncaptured CASCADE;
drop table leo_match CASCADE;
drop table leo_graduation CASCADE;
drop table leo_detail CASCADE;
drop table leo CASCADE;

-- DROP SEQUENCE IN PROPER ORDER
DROP SEQUENCE seq_leo_id;
DROP SEQUENCE seq_leo_detail_id;
DROP SEQUENCE seq_grads_id;
DROP SEQUENCE seq_match_id;
DROP SEQUENCE seq_uncaptured_id;
DROP SEQUENCE seq_sust_emp_id;
DROP SEQUENCE seq_earning_id;
DROP SEQUENCE seq_polar_id;
DROP SEQUENCE seq_pr_att_id;

-- CREATE SEQUENCE IN REVERSE ORDER
CREATE SEQUENCE seq_pr_att_id           MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_polar_id            MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_earning_id          MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_sust_emp_id         MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_uncaptured_id       MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_match_id            MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_grads_id            MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_leo_detail_id       MINVALUE 1 INCREMENT 1;
CREATE SEQUENCE seq_leo_id              MINVALUE 1 INCREMENT 1;

-- CREATE TABLE IN REVERSE ORDER
create table leo_prior_attainment (
        leo_pr_att_id           INT DEFAULT NEXTVAL ('seq_pr_att_id'),
        leo_pr_att_band         varchar(20) NOT NULL,
        leo_pr_att_included     varchar(20) NOT NULL,
        PRIMARY KEY (leo_pr_att_id)
);

create table leo_polar (
        leo_polar_id            INT DEFAULT NEXTVAL ('seq_polar_id'),
        leo_polar_grp_one       varchar(20) NOT NULL,
        leo_polar_grp_included  varchar(20) NOT NULL,
        PRIMARY KEY (leo_polar_id)
);

create table leo_earning (
        leo_earning_id          INT DEFAULT NEXTVAL ('seq_earning_id'),
        leo_earning_include     varchar(20) NOT NULL,
        leo_lower_ann_earn      varchar(20) NOT NULL,
        leo_median_ann_earn     varchar(20) NOT NULL,
        leo_upper_ann_earn      varchar(20) NOT NULL,
        PRIMARY KEY (leo_earning_id)
);

create table leo_sustain_employment (
        leo_sust_emp_id                 INT DEFAULT NEXTVAL ('seq_sust_emp_id'),
        leo_sust_emp_only               varchar(20) NOT NULL,
        leo_sust_emp                    varchar(20) NOT NULL,
        leo_sust_emp_fs_or_both         varchar(20) NOT NULL,
        PRIMARY KEY (leo_sust_emp_id)
);

```

```
create table leo_uncaptured (
        leo_uncaptured_id               INT DEFAULT NEXTVAL ('seq_uncaptured_id'),
        leo_activitynotcaptured         varchar(20) NOT NULL,
        leo_no_sust_dest                varchar(20) NOT NULL,
        PRIMARY KEY (leo_uncaptured_id)
);

create table leo_match (
        leo_match_id                    INT DEFAULT NEXTVAL ('seq_match_id'),
        leo_unmatched                   varchar(20) NOT NULL,
        leo_matched                     varchar(20) NOT NULL,
        PRIMARY KEY (leo_match_id)
);

create table leo_graduation (
        leo_grads_id                    INT DEFAULT NEXTVAL ('seq_grads_id'),
        leo_grad                        varchar(10) NOT NULL,
        PRIMARY KEY (leo_grads_id)
);

create table leo_detail (
        leo_detail_id                   INT DEFAULT NEXTVAL ('seq_leo_detail_id'),
        leo_ukprn                       int             NOT NULL,
        leo_providername                varchar(100)    NOT NULL,
        leo_region                      varchar(50)     NOT NULL,
        leo_subject                     varchar(50)     NOT NULL,
        leo_sex                         varchar(30)     NOT NULL,
        leo_yearaftergraduation         int             NOT NULL,
        PRIMARY KEY (leo_detail_id)
);

create table leo (
        leo_id                          INT DEFAULT NEXTVAL ('seq_leo_id'),
        leo_detail_id                   INT NOT NULL,
        leo_grads_id                    INT NOT NULL,
        leo_match_id                    INT NOT NULL,
        leo_uncaptured_id               INT NOT NULL,
        leo_sust_emp_id                 INT NOT NULL,
        leo_earning_id                  INT NOT NULL,
        leo_polar_id                    INT NOT NULL,
        leo_pr_att_id                   INT NOT NULL,

        FOREIGN KEY (leo_detail_id) REFERENCES leo_detail (leo_detail_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_grads_id) REFERENCES leo_graduation (leo_grads_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_match_id) REFERENCES leo_match (leo_match_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_uncaptured_id) REFERENCES leo_uncaptured (leo_uncaptured_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_sust_emp_id) REFERENCES leo_sustain_employment (leo_sust_emp_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_earning_id) REFERENCES leo_earning (leo_earning_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_polar_id) REFERENCES leo_polar (leo_polar_id) ON DELETE CASCADE,
        FOREIGN KEY (leo_pr_att_id) REFERENCES leo_prior_attainment (leo_pr_att_id) ON DELETE CASCADE
);

-- CHECK ALL THE TABLES
\dt
```

LISTING M.3: PL/pgSQL's DDL scripts for Education Normalized Table Creation.

Listing M.3 show the PL/pgSQL's Data Definition Language (DDL) scripts to create LEO Normalized table based on database design shown in Section 3.6.2.4. All the table are defined with PRIMARY KEY (PK) and establish referencial integrity relationship amongs entity to form a good relational database design. Moreover, the data types are defined correctly with sufficient memory provided on each columns.

## M.4 List of database relations

### M.4.1 List Relations of Postcode database

```
Schema |                Name                 |   Type   |  Owner  |    Size    |
      -------+-------------------------------------+----------+---------+------------+
public | nspl_rawdata                        | table    | yinghua | 1403 MB    |
public | postcode                            | table    | yinghua | 0 bytes    |
public | postcode_cartesian_coordinate       | table    | yinghua | 0 bytes    |
public | postcode_country                    | table    | yinghua | 0 bytes    |
public | postcode_county                     | table    | yinghua | 0 bytes    |
public | postcode_detail                     | table    | yinghua | 0 bytes    |
public | postcode_euro_electoral_region      | table    | yinghua | 8192 bytes |
public | postcode_greek_coordinate           | table    | yinghua | 0 bytes    |
public | postcode_local_authority_county     | table    | yinghua | 0 bytes    |
public | postcode_lower_super_output_area    | table    | yinghua | 0 bytes    |
public | postcode_middle_super_output_area   | table    | yinghua | 0 bytes    |
public | postcode_output_area_classification | table    | yinghua | 0 bytes    |
public | postcode_parliament_constituency    | table    | yinghua | 0 bytes    |
public | postcode_primary_care_trust         | table    | yinghua | 0 bytes    |
public | postcode_region                     | table    | yinghua | 0 bytes    |
public | postcode_ward                       | table    | yinghua | 0 bytes    |
public | seq_cart_coordinate_id              | sequence | yinghua | 8192 bytes |
public | seq_country_id                      | sequence | yinghua | 8192 bytes |
public | seq_county_id                       | sequence | yinghua | 8192 bytes |
public | seq_eer_id                          | sequence | yinghua | 8192 bytes |
public | seq_greek_coordinate_id             | sequence | yinghua | 8192 bytes |
public | seq_lac_id                          | sequence | yinghua | 8192 bytes |
public | seq_lsoa_id                         | sequence | yinghua | 8192 bytes |
public | seq_msoa_id                         | sequence | yinghua | 8192 bytes |
public | seq_oac_id                          | sequence | yinghua | 8192 bytes |
public | seq_par_cons_id                     | sequence | yinghua | 8192 bytes |
public | seq_pct_id                          | sequence | yinghua | 8192 bytes |
public | seq_pos_detail_id                   | sequence | yinghua | 8192 bytes |
public | seq_pos_form_id                     | sequence | yinghua | 8192 bytes |
public | seq_pos_temp_id                     | sequence | yinghua | 8192 bytes |
public | seq_region_id                       | sequence | yinghua | 8192 bytes |
public | seq_ward_id                         | sequence | yinghua | 8192 bytes |
(32 rows)
```

Listing M.4: List all relations in Postcode database.

Listing M.4 shows all the database relation found in Postcode database. The result shows the normalized entity are created and defined successfully based on Entity Relationship Diagram database design with PL/pgSQL's DDL scripts.

## M.4.2 List Relations of Company database

```
 Schema |           Name           |   Type   |  Owner  |    Size    |
--------+--------------------------+----------+---------+------------+
 public | company                  | table    | yinghua | 0 byte        |
 public | company_account          | table    | yinghua | 0 byte        |
 public | company_account_category | table    | yinghua | 0 byte        |
 public | company_category         | table    | yinghua | 0 byte        |
 public | company_conf_stmt        | table    | yinghua | 0 byte        |
 public | company_detail           | table    | yinghua | 0 byte        |
 public | company_mortgages        | table    | yinghua | 0 byte        |
 public | company_partnership      | table    | yinghua | 0 byte        |
 public | company_previousname     | table    | yinghua | 0 byte        |
 public | company_raw              | table    | yinghua | 1658 MB    |
 public | company_rawdata          | table    | yinghua | 2476 MB    |
 public | company_returns          | table    | yinghua | 0 byte        |
 public | company_siccodes         | table    | yinghua | 0 byte        |
 public | company_status           | table    | yinghua | 0 byte        |
 public | company_uri              | table    | yinghua | 0 byte        |
 public | seq_acc_category_id      | sequence | yinghua | 8192 bytes |
 public | seq_acc_id               | sequence | yinghua | 8192 bytes |
 public | seq_category_id          | sequence | yinghua | 8192 bytes |
 public | seq_conf_stmt_id         | sequence | yinghua | 8192 bytes |
 public | seq_detail_id            | sequence | yinghua | 8192 bytes |
 public | seq_mort_id              | sequence | yinghua | 8192 bytes |
 public | seq_partnership_id       | sequence | yinghua | 8192 bytes |
 public | seq_pn_id                | sequence | yinghua | 8192 bytes |
 public | seq_return_id            | sequence | yinghua | 8192 bytes |
 public | seq_sic_id               | sequence | yinghua | 8192 bytes |
 public | seq_status_id            | sequence | yinghua | 8192 bytes |
 public | seq_uri_id               | sequence | yinghua | 8192 bytes |
 (27 rows)
```

Listing M.5: List all relations in Company database.

Listing M.4 shows all the database relation found in Company database. The result shows the normalized entity are created and defined successfully based on Entity Relationship Diagram database design with PL/pgSQL's DDL scripts.

### M.4.3 List Relations of Education database

```
 Schema |          Name          |   Type   |  Owner  |    Size    | Description
--------+------------------------+----------+---------+------------+-------------
 public | leo                    | table    | yinghua | 0 byte     |
 public | leo_detail             | table    | yinghua | 0 byte     |
 public | leo_earning            | table    | yinghua | 0 byte     |
 public | leo_graduation         | table    | yinghua | 0 byte     |
 public | leo_match              | table    | yinghua | 0 byte     |
 public | leo_polar              | table    | yinghua | 0 byte     |
 public | leo_prior_attainment   | table    | yinghua | 0 byte     |
 public | leo_rawdata            | table    | yinghua | 0 byte     |
 public | leo_sustain_employment | table    | yinghua | 0 byte     |
 public | leo_uncaptured         | table    | yinghua | 0 byte     |
 public | seq_earning_id         | sequence | yinghua | 8192 bytes |
 public | seq_grads_id           | sequence | yinghua | 8192 bytes |
 public | seq_leo_detail_id      | sequence | yinghua | 8192 bytes |
 public | seq_leo_id             | sequence | yinghua | 8192 bytes |
 public | seq_match_id           | sequence | yinghua | 8192 bytes |
 public | seq_polar_id           | sequence | yinghua | 8192 bytes |
 public | seq_pr_att_id          | sequence | yinghua | 8192 bytes |
 public | seq_sust_emp_id        | sequence | yinghua | 8192 bytes |
 public | seq_uncaptured_id      | sequence | yinghua | 8192 bytes |
(19 rows)
```

Listing M.6: List all relations in Education database.

Listing M.4 shows all the database relation found in Education database. The result shows the normalized entity are created and defined successfully based on Entity Relationship Diagram database design with PL/pgSQL's DDL scripts.

# Appendix N

# Data Parser

## N.1 Go program based Data Cleaning Parser

### N.1.1 Function to clean and parse data

```
package main

import (
	"fmt"
	"strconv"
	"bufio"
	"encoding/csv"
	"io"
	"os"
	"time"
)

=========================================================================================================
Perform cleaning and parsing on data retrieved from CSV and import processed data into PostgreSQL database
=========================================================================================================
func importCSVtoDB() {

	start := time.Now()
	retrieveCSV()
	importDB()

	fmt.Printf("%.5fs seconds on cleaned 3595702 rows of company data. \n", time.Since(start).Seconds())

}

=============================================================================================================
Retrieve data 3595702 lines of data from CSV to eliminate NULL values and standardize data in specific columns
=============================================================================================================
func retrieveCSV() {
	csvFile, err := os.Open(COMPANY_FILE_DIRECTORY)
	checkErr(err, "Open CSV")

	defer csvFile.Close()

	// Create a new reader.
	reader := csv.NewReader(bufio.NewReader(csvFile))

	start := time.Now()

	for i := 0; i < ENTRIES; i++ {

		record, err := reader.Read()

		if i == 0 {
			continue
		}

		if i == 100000 {
			fmt.Println("Cleaned 100000 rows", time.Since(start).Seconds())
		} else if i == 500000 {
			fmt.Println("Cleaned 500000 rows", time.Since(start).Seconds())
		} else if i == 1000000 {
			fmt.Println("Cleaned 1000000 rows", time.Since(start).Seconds())
		} else if i == 2000000 {
			fmt.Println("Cleaned 2000000 rows", time.Since(start).Seconds())
		} else if i == 3000000 {
			fmt.Println("Cleaned 3000000 rows", time.Since(start).Seconds())
		} else if i == 4000000 {
			fmt.Println("Cleaned 4000000 rows", time.Since(start).Seconds())
		}

		// Stop at EOF.
		if err == io.EOF {
			break
		}

		int_mortchange, err := strconv.Atoi(record[22])
		checkErr(err, "convert mortchange value to integer")

		int_mortoutstanding, err  := strconv.Atoi(record[23])
		checkErr(err, "convert mortoutstanding value to integer")

		int_mortpartsatisfied, err := strconv.Atoi(record[24])
		checkErr(err, "convert mortpartsatisfied value to integer")

		int_mortsatisfied, err  := strconv.Atoi(record[25])
		checkErr(err, "convert mortsatisfied value to integer")

		int_genpartner, err := strconv.Atoi(record[30])
		checkErr(err, "convert genpartner value to integer")

		int_limpartner, err := strconv.Atoi(record[31])
		checkErr(err, "convert limpartner value to integer")


		company := company_rawdata{
			number: record[1],
			num_MortChanges: int_mortchange,
			num_MortOutstanding: int_mortoutstanding,
			num_MortPartSatisfied: int_mortpartsatisfied,
			num_MortSatisfied: int_mortsatisfied,
			num_genPartner: int_genpartner,
			num_limPartner: int_limpartner,
			uri: record[32],
		}

		company.category.Scan(record[10])
		if len(company.category.String) == 0 {
			company.category.String = "Undefined"
		}

		company.status.Scan(record[11])
		if len(company.status.String) == 0 {
			company.status.String = "Undefined"
		}

		company.countryOfOrigin.Scan(record[12])
		if len(company.countryOfOrigin.String) < 2 {
			company.countryOfOrigin.String = "Undefined"
		}

		company.name.Scan(record[0])
		if len(company.name.String) == 0 {
			company.name.String = "Undefined"
		}

		company.careOf.Scan(record[2])
		if len(company.careOf.String) == 0 {
			company.careOf.String = "Undefined"
		}

		company.poBox.Scan(record[3])
		if len(company.poBox.String) == 0 {
			company.poBox.String = "Undefined"
		}

		company.addressLine1.Scan(record[4])
		if len(company.addressLine1.String) == 0 {
			company.addressLine1.String = "Undefined"
		}

		company.addressLine2.Scan(record[5])
		if len(company.addressLine2.String) == 0 {
			company.addressLine2.String = "Undefined"
```

```
                }

                company.postTown.Scan(record[6])
                if len(company.postTown.String) == 0 {
                        company.postTown.String = "Undefined"
                }

                company.county.Scan(record[7])
                if len(company.county.String) == 0 {
                        company.county.String = "Undefined"
                }

                company.country.Scan(record[8])
                if len(company.country.String) == 0 {
                        company.country.String = "Undefined"
                }

                company.postcode.Scan(record[9])
                if len(company.postcode.String) == 0 {
                        company.postcode.String = "Undefined"
                }

                company.dissolution_date.Scan(record[13])
                if len(company.dissolution_date.String) == 0 {
                        company.dissolution_date.String = "01/01/3000"
                }

                company.incorporate_date.Scan(record[14])
                if len(company.dissolution_date.String) == 0 {
                        company.dissolution_date.String = "01/01/3000"
                }

                company.accounting_refDay.Scan(record[15])
                company.accounting_refMonth.Scan(record[16])

                company.account_nextDueDate.Scan(record[17])
                if len(company.account_nextDueDate.String) == 0 {
                        company.account_nextDueDate.String = "01/01/3000"
                }

                company.account_lastMadeUpdate.Scan(record[18])
                if len(company.account_lastMadeUpdate.String) == 0 {
                        company.account_lastMadeUpdate.String = "01/01/3000"
                }

                ......
                (Source code not fully display)
                (many if-else from record[19] to record[49] on data handling to eliminate NULL values ....)
                ......
                ......

                company.pn9_companyname.Scan(record[50])
                if len(company.pn9_companyname.String) == 0 {
                        company.pn9_companyname.String = "Undefined"
                }

                company.pn10_condat             companycategoryArray = append(companycategoryArray, company.category.String)
                companystatusArray = append(companystatusArray, company.status.String)
                countryoforiginArray = append(countryoforiginArray, company.countryOfOrigin.String)
                dissolutiondateArray = append(dissolutiondateArray, company.dissolution_date.String)
                incorporatedateArray = append(incorporatedateArray, company.incorporate_date.String) e.Scan(record[51])
                if len(company.pn10_condate.String) == 0 {
                        company.pn10_condate.String = "01/01/3000"
                }

                company.pn10_companyname.Scan(record[52])
                if len(company.pn10_companyname.String) == 0 {
                        company.pn10_companyname.String = "Undefined"
                }

                company.conf_stmtNextDueDate.Scan(record[53])
                if len(company.conf_stmtNextDueDate.String) == 0 {
                        company.conf_stmtNextDueDate.String = "01/01/3000"
                }

                company.conf_stmtLastMadeUpdate.Scan(record[54])
                if len(company.conf_stmtLastMadeUpdate.String) == 0 {
                        company.conf_stmtLastMadeUpdate.String = "01/01/3000"
                }

                companynameArray = append(companynameArray, company.name.String)
                companynumberArray = append(companynumberArray, company.number)
                careofArray = append(careofArray, company.careOf.String)
                poboxArray = append(poboxArray, company.poBox.String)
                addressline1Array = append(addressline1Array, company.addressLine1.String)

                addressline2Array = append(addressline2Array, company.addressLine2.String)
                posttownArray = append(posttownArray, company.postTown.String)
                countyArray = append(countyArray, company.county.String)
                countryArray = append(countryArray, company.country.String)
                postcodeArray = append(postcodeArray, company.postcode.String)


                                ......
                                (Source code not fully display)
                                (many appends of array for data standardization ....)
                                ......
                                ......

                pn9_companyname_Array = append(pn9_companyname_Array, company.pn9_companyname.String)
                pn10_condate_Array = append(pn10_condate_Array, company.pn10_condate.String)
                pn10_companyname_Array = append(pn10_companyname_Array, company.pn10_companyname.String)
                confstmtnextduedateArray = append(confstmtnextduedateArray, company.conf_stmtNextDueDate.
     String)
                confstmtlastmadeupdateArray = append(confstmtlastmadeupdateArray, company.
     conf_stmtLastMadeUpdate.String)

        }
}

================================================================
P/S: The source code is not fully display due to lack of space,
                to view full source code refer /go-import-company/retrieve-csv.go

================================================================
```

Listing N.1: Parse and cleaned data retrieved from CSV

Listing N.1 shows the source code for function on data retrieval from CSV and perform data cleaning and data standardization. The function possess control flow (if-else statement) to check the empty and missing fields in each records. If the record is found empty and missing, the program will replace a standard value to indicate the field is meaningful.

The information of data repairing in this program are as shown in table below:

| **Missing data retrieved from CSV file** | **Replaced data by Go program with standard value** |
|---|---|
| INTEGER(10) | 0 |
| DATE | ”01/01/3000” |
| VARCHAR | ”Undefined” |
| CHAR | ”—” |
| REAL | 0.0 |

Table N.1: Data repair on missing values.

After the data in specific columns is replaced and fixed, it will be stored into dedicated array await to be import into database.

### N.1.2 Function to import cleaned data into PostgreSQL database.

```
===========================================================================================================
Import cleaned data processed by retrieveCSV() into PostgreSQL database with Semaphore concurrent concepts.
===========================================================================================================
func importDB() {

        // Assigned 400000 Gorountines
        sem := make (chan bool, 400000)

        initDB()
        fmt.Println("Prepare to import data")

        var sStmt string = "INSERT INTO company_rawdata1 VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20, $21, $22, $23, $24, $25, $26, $27, $28, $29, $30, $31, $32, $33, $34, $35, $36, $37, $38, $39, $40, $41, $42, $43, $44, $45, $46, $47, $48, $49, $50, $51, $52, $53, $54, $55);"

        stmt, err := db.Prepare(sStmt)
        checkErr(err, "Prepare insert com_category")

        for i := len(companynameArray); i > 0; i-- {
                sem <- true
                go func () {
                        defer func () { <- sem } ()
                        _, err = stmt.Exec(companynameArray[i], companynumberArray[i], careofArray[i], poboxArray[i], addressline1Array[i],
                        addressline2Array[i], posttownArray[i], countyArray[i], countryArray[i], postcodeArray[i],
                        companycategoryArray[i], companystatusArray[i], countryoforiginArray[i], dissolutiondateArray[i], incorporatedateArray[i],
                        refdayArray[i], refmonthArray[i], a_nextduedateArray[i], a_lastmadeupdateArray[i], accountcategoryArray[i],
                        nextduedateArray[i], lastmadeupdateArray[i], mortchargesArray[i], mortoutstandingArray[i], mortpartsatisfiedArray[i],
                        mortsatisfiedArray[i], siccode1Array[i], siccode2Array[i], siccode3Array[i], siccode4Array[i],
                        genPartnerArray[i], limPartnerArray[i], uriArray[i], pn1_condate_Array[i], pn1_companyname_Array[i],
                        pn2_condate_Array[i], pn2_companyname_Array[i], pn3_condate_Array[i], pn3_companyname_Array[i], pn4_condate_Array[i],
                        pn4_companyname_Array[i], pn5_condate_Array[i], pn5_companyname_Array[i], pn6_condate_Array[i], pn6_companyname_Array[i],
                        pn7_condate_Array[i], pn7_companyname_Array[i], pn8_condate_Array[i], pn8_companyname_Array[i], pn9_condate_Array[i],
                        pn9_companyname_Array[i], pn10_condate_Array[i], pn10_companyname_Array[i], confstmtnextduedateArray[i], confstmtlastmadeupdateArray[i])

                        checkErr(err, "Company Data Importation")
                }()
        }
}
```

Listing N.2: Import cleaned data into PostgreSQL database

Listing N.2 shows the source code for function on data importation into PostgreSQL database. The function insert all the cleaned and processed data stores in array into table declared in PostgreSQL database.

400000 *Goroutines* is assigned to increase the execution process of data cleaning with *Semaphore* concurrent concepts. These Goroutines communicate with each other to perform importation of 3 millions row of data with 299 active connection available.

This program execution duration will be tabulated, compared and discussed.

## N.2 Data Consistency Verification

### N.2.1 Validate Company Data Completeness and Comformances

```
====================================
Step 1 - Connect to company database
====================================
yinghua@yinghua:~$ psql company;
psql (9.5.10)
Type "help" for help.

company=# \d+

============================================================================
Step 2 - Select some columns that contain NULL values before data is cleaned
============================================================================

company=# \d+
        Column         |          Type          |                Modifiers                | Storage  |
-----------------------+------------------------+-----------------------------------------+----------+
careof                 | character varying(100) | default 'Undefined'::character varying  | extended |
pobox                  | character varying(10)  | default 'Undefined'::character varying  | extended |
addressline1           | character varying(300) | default 'Undefined'::character varying  | extended |
addressline2           | character varying(300) | default 'Undefined'::character varying  | extended |
posttown               | character varying(50)  | default 'Undefined'::character varying  | extended |

============================================================================
Step 3 - Verify the completeness of selected columns
============================================================================
company=# select careof from company_rawdata where careof is null;
careof
--------
(0 rows)

company=# select pobox from company_rawdata where pobox is null;
pobox
--------
(0 rows)

company=# select addressline1 from company_rawdata where addressline1 is null;
addressline1
--------
(0 rows)

company=# select addressline2 from company_rawdata where addressline2 is null;
addressline2
--------
(0 rows)

company=# select posttown from company_rawdata where posttown is null;
posttown
--------
(0 rows)
```

Listing N.3: Import cleaned data into PostgreSQL database

In this section, we will verify the completeness of several columns to demonstrate the missing data and NULL values are eliminated with data parser.

# Appendix O

# Database Tuning

## O.1 Increase Max Concurrent Connection Limit

```
================================
Step 1 - Connect to any database
================================
yinghua@yinghua:~$ psql company;
psql (9.5.10)
Type "help" for help.

company=#

============================================
Step 2- Display the location of config file
============================================
company=# show config_file;
config_file
------------------------------------------
/etc/postgresql/9.5/main/postgresql.conf
(1 row)

==================================================================================
Step 3 - Close the database and login as root with admin privileges on Ubuntu OS
==================================================================================
yinghua@yinghua:~$ sudo su
[sudo] password for yinghua:
root@yinghua:/home/yinghua#

==================================================================================
Step 4 - Configure the value of Max Connection Limit in PostgreSQL Configuration file
==================================================================================
root@yinghua:/home/yinghua# sudo gedit /etc/postgresql/9.5/main/postgresql.conf

# -----------------------------
# PostgreSQL configuration file
# -----------------------------
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
#
# The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
#
# This file is read on server startup and when the server receives a SIGHUP
# signal.  If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, or use "pg_ctl reload".  Some
# parameters, which are marked below, require a server shutdown and restart to
```

```
51 # take effect.
52 #
53 # Any parameter can also be given as a command-line option to the server, e.g.,
54 # "postgres -c log_connections=on".  Some parameters can be changed at run time
55 # with the "SET" SQL command.
56 #
57 # Memory units:  kB = kilobytes        Time units:  ms  = milliseconds
58 #                MB = megabytes                     s   = seconds
59 #                GB = gigabytes                     min = minutes
60 #                TB = terabytes                     h   = hours
61 #                                                   d   = days
62
63 (.... other settings found in this configuration files)
64
65 #------------------------------------------------------------------------------
66 # CONNECTIONS AND AUTHENTICATION
67 #------------------------------------------------------------------------------
68
69 # - Connection Settings -
70
71 #listen_addresses = '*'                 # what IP address(es) to listen on;
72 # comma-separated list of addresses;
73 # defaults to 'localhost'; use '*' for all
74 # (change requires restart)
75 port = 5432                             # (change requires restart)
76 max_connections = 300                   # (change requires restart)  <========== Modify from 100 to 300
77
78 ===================================================================================
79 Step 5 - Restart the PostgreSQL database to update the changes
80 ==================================================================================
81 root@yinghua:/home/yinghua# /etc/init.d/postgresql restart
82 [ ok ] Restarting postgresql (via systemctl): postgresql.service.
```

Listing O.1: Increase Max Concurrent Connection Limit

Listing O.1 shows the detail procedure to increase the number of client to establish concurrent connection with PostgreSQL database. Step 1 and Step 2 is performed to identify the location of configuration file because the mentioned file is stored in different place depends on operating system.

After the location of configuration file is identified, it is required to login as root privileged on Ubuntu OS with administrator credential to perform any modification on Linux's file ownership (Step 3). Then, we open the configuration files with directory as input and increase the **max_connection** parameter from 100 to 300 (refer row 76). The modification requires restart of PostgreSQL database to update the changes.

## O.2 Increase Shared Buffer utilized by PostgreSQL Database

```
================================
Step 1 - Connect to any database
================================
yinghua@yinghua:~$ psql company;
psql (9.5.10)
Type "help" for help.

company=#

============================================
Step 2- Display the location of config file
============================================
company=# show config_file;
config_file
------------------------------------------
/etc/postgresql/9.5/main/postgresql.conf
(1 row)

===================================================================================
Step 3 - Close the database and login as root with admin privileges on Ubuntu OS
===================================================================================
yinghua@yinghua:~$ sudo su
[sudo] password for yinghua:
root@yinghua:/home/yinghua#

===================================================================================
Step 4 - Configure the value of Shared Buffer parameters in PostgreSQL Configuration file
===================================================================================
root@yinghua:/home/yinghua# sudo gedit /etc/postgresql/9.5/main/postgresql.conf

# -----------------------------
# PostgreSQL configuration file
# -----------------------------
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
#
# The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
#
# This file is read on server startup and when the server receives a SIGHUP
# signal.  If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, or use "pg_ctl reload".  Some
# parameters, which are marked below, require a server shutdown and restart to
# take effect.
#
# Any parameter can also be given as a command-line option to the server, e.g.,
# "postgres -c log_connections=on".  Some parameters can be changed at run time
# with the "SET" SQL command.
#
# Memory units:  kB = kilobytes        Time units:  ms  = milliseconds
#                MB = megabytes                     s   = seconds
#                GB = gigabytes                     min = minutes
#                TB = terabytes                     h   = hours
#                                                   d   = days

(.... other settings found in this configuration files)

#-------------------------------------------------------------------------------
# RESOURCE USAGE (except WAL)
#-------------------------------------------------------------------------------

# - Memory -

shared_buffers = 256MB                  # min 128kB              <========== Modify from 128MB to 256MB
                                                        # (change requires restart)
#hutilized by PostgreSQL Databaseuge_pages = try                        # on, off, or try
                                                        # (change requires restart)
#temp_buffers = 8MB                     # min 800kB
#max_prepared_transactions = 0          # zero disables the feature
                                                # (change requires restart)
                                                # Caution: it is not advisable to set
     max_prepared_transactions nonzero unless
                                                  # you actively intend to use prepared transactions.
#work_mem = 4MB                         # min 64kB
#maintenance_work_mem = 64MB            # min 1MB
```

```
82 #autovacuum_work_mem = -1              # min 1MB, or -1 to use maintenance_work_mem
83 #max_stack_depth = 2MB                # min 100kB
84 dynamic_shared_memory_type = posix     # the default is the first option
85                                                          # supported by the operating system:
86                                                          #   posix
87                                                          #   sysv
88                                                          #   windows
89                                                          #   mmap
90                                                          # use none to disable dynamic shared memory
91                                                          # (change requires restart)
92
93 ============================================================================
94 Step 5 - Restart the PostgreSQL database to update the changes
95 ============================================================================
96 root@yinghua:/home/yinghua# /etc/init.d/postgresql restart
97 [ ok ] Restarting postgresql (via systemctl): postgresql.service.
```

Listing O.2: Increase Shared Buffer utilized by PostgreSQL Database

Listing O.2 shows the detail procedure to increase the number of shared buffer utilized by PostgreSQL database. Step 1 and Step 2 is performed to identify the location of configuration file because the mentioned file is stored in different place depends on operating system.

After the location of configuration file is identified, it is required to login as root privileged on Ubuntu OS with administrator credential to perform any modification on Linux's file ownership (Step 3). Then, we open the configuration files with directory as input and increase the **shared_buffer** parameter from 126MB to 256MB (refer row 71). The modification requires restart of PostgreSQL database to update the changes.

## O.3 Increase maximum size of shared memory segment.

```
1  ===================================================================================
2  Step 1 - Login as root with admin privileges on Ubuntu OS
3  ===================================================================================
4  yinghua@yinghua:~$ sudo su
5  [sudo] password for yinghua:
6  root@yinghua:/home/yinghua#
7
8  =================================================================================================
9  Step 2 - Add the value of maximum size of shared memory segment into Ubuntu System Configuration
10 =================================================================================================
11 root@yinghua:/home/yinghua# sudo gedit /etc/sysctl.conf
12 kernel.shmmax=26000000000       <========== Add this line, it is equal to 26GB
13
14 ====================================================================================
15 Step 3 - Restart the PostgreSQL database to update the changes
16 ===================================================================================
17 root@yinghua:/home/yinghua# /etc/init.d/postgresql restart
18 [ ok ] Restarting postgresql (via systemctl): postgresql.service.
```

Listing O.3: Increase maximum size of shared memory segment.

Listing O.3 shows the detail procedure to increase the maximum size of memory segment shared to PostgreSQL database.

The operation required to login as root privileged on Ubuntu OS with administrator credential to perform any modification on Linux's file ownership (Step 3). We will configure and modifies the attributes of system kernels to allocate extra memory for PostgreSQL database to perform transaction. The **sysctl.conf** file is opened and **shared_buffer** parameter with 26,000,000,000B (26GB) is added at the end of files (refer row 12). The modification requires restart of PostgreSQL database to update the changes.

# Appendix P

# Data Migration

## P.1 PL/pgSQL's DML Script for Data Migration.

### P.1.1 Script for Education Normalized Database Migration.

```

-- File: 03_yinghua_insert_leo_table_DML.sql
-- Date: Mon Dec 8 10:10 MYT 2017
-- Author: Chai Ying Hua
-- Version: 1.0
-- Database: psql (PostgreSQL) 9.5.10
-- ====================================================================
-- (Version 1.0 Change: 8 Dec 2017)
--      1. Delete all data in reverse order.
--      2. Migrate all data from raw table into normalized lookup table.
-- ====================================================================


-- DELETE ALL DATA FROM TABLE IN REVERSE ORDER
DELETE FROM leo_prior_attainment        WHERE TRUE;
DELETE FROM leo_polar                   WHERE TRUE;
DELETE FROM leo_earning                 WHERE TRUE;
DELETE FROM leo_sustain_employment      WHERE TRUE;
DELETE FROM leo_uncaptured              WHERE TRUE;
DELETE FROM leo_match                   WHERE TRUE;
DELETE FROM leo_graduation              WHERE TRUE;
DELETE FROM leo_detail                  WHERE TRUE;
DELETE FROM leo                         WHERE TRUE;

-- SELECT UNIQUE DATA FROM RAW TABLE AND INSERT INTO NORMALIZED DATA.

----------------------------------------
-- LEO_PRIOR_ATTAINMENT TABLE MIGRATION
-- ROW COUNTS: 2139
----------------------------------------
INSERT INTO leo_prior_attainment (leo_pr_att_band,leo_pr_att_included)
        SELECT DISTINCT prattband, prattincluded FROM leo_rawdata;

----------------------------------------
-- LEO_POLAR TABLE MIGRATION
-- ROW COUNTS: 6793
----------------------------------------
INSERT INTO leo_polar (leo_polar_grp_one,leo_polar_grp_included)
        SELECT DISTINCT polargrpone, polargrponeincluded FROM leo_rawdata;

----------------------------------------
-- LEO_EARNING TABLE MIGRATION
-- ROW COUNTS: 14372
----------------------------------------
INSERT INTO leo_earning (leo_earning_include,leo_lower_ann_earn,leo_median_ann_earn,leo_upper_ann_earn)
        SELECT DISTINCT earningsinclude, lowerannearn, medianannearn, upperannearn FROM leo_rawdata;

----------------------------------------
-- LEO_SUSTAIN_EMPLOYMENT TABLE MIGRATION
-- ROW COUNTS: 6192
----------------------------------------
INSERT INTO leo_sustain_employment (leo_sust_emp_only,leo_sust_emp,leo_sust_emp_fs_or_both)
        SELECT DISTINCT sustemponly, sustemp, sustempfsorboth FROM leo_rawdata;

----------------------------------------
-- LEO_UNCAPTURED TABLE MIGRATION
-- ROW COUNTS: 6283
----------------------------------------
INSERT INTO leo_uncaptured (leo_activitynotcaptured,leo_no_sust_dest)
        SELECT DISTINCT activitynotcaptured, nosustdest FROM leo_rawdata;

----------------------------------------
-- LEO_MATCH TABLE MIGRATION
-- ROW COUNTS: 3992
----------------------------------------
INSERT INTO leo_match (leo_unmatched,leo_matched)
        SELECT DISTINCT unmatched, matched FROM leo_rawdata;

----------------------------------------
-- LEO_GRADUATION TABLE MIGRATION
-- ROW COUNTS: 195
----------------------------------------
INSERT INTO leo_graduation (leo_grad)
        SELECT DISTINCT grads FROM leo_rawdata;

----------------------------------------
-- LEO_DETAIL TABLE MIGRATION
-- ROW COUNTS: 32706            <- SAME COUNT WITH RAWDATA
----------------------------------------
INSERT INTO leo_detail (leo_ukprn, leo_providername, leo_region, leo_subject, leo_sex, leo_yearaftergraduation)
        SELECT DISTINCT ukprn, providername, region, subject, sex, yearaftergraduation FROM leo_rawdata;

----------------------------------------
-- LEO TABLE MIGRATION
-- ROW COUNTS: 32706            <- SAME COUNT WITH RAWDATA
----------------------------------------
INSERT INTO leo (leo_detail_id, leo_grads_id, leo_match_id, leo_uncaptured_id, leo_sust_emp_id, leo_earning_id, leo_polar_id, leo_pr_att_id)
        SELECT leo_detail_id, leo_grads_id, leo_match_id, leo_uncaptured_id, leo_sust_emp_id, leo_earning_id, leo_polar_id, leo_pr_att_id
        FROM leo_rawdata AS rawdata
        JOIN leo_detail     AS detail
                ON      detail.leo_ukprn = rawdata.ukprn
                AND     detail.leo_providername = rawdata.providername
                AND     detail.leo_region = rawdata.region
                AND     detail.leo_subject = rawdata.subject
                AND     detail.leo_sex = rawdata.sex
                AND     detail.leo_yearaftergraduation = rawdata.yearaftergraduation
        JOIN leo_graduation AS grad
                ON      grad.leo_grad = rawdata.grads
        JOIN leo_match      AS match
                ON      match.leo_unmatched = rawdata.unmatched
                AND     match.leo_matched   = rawdata.matched
        JOIN leo_uncaptured AS uncaptured
                ON      uncaptured.leo_activitynotcaptured = rawdata.activitynotcaptured
                AND     uncaptured.leo_no_sust_dest        = rawdata.nosustdest
        JOIN leo_sustain_employment       AS sustemp
                ON      sustemp.leo_sust_emp_only       = rawdata.sustemponly
                AND     sustemp.leo_sust_emp            = rawdata.sustemp
                AND     sustemp.leo_sust_emp_fs_or_both = rawdata.sustempfsorboth
        JOIN leo_earning    AS earning
                ON      earning.leo_earning_include     = rawdata.earningsinclude
                AND     earning.leo_lower_ann_earn      = rawdata.lowerannearn
                AND     earning.leo_median_ann_earn     = rawdata.medianannearn
                AND     earning.leo_upper_ann_earn      = rawdata.upperannearn
        JOIN leo_polar      AS polar
                ON      polar.leo_polar_grp_one         = rawdata.polargrpone
                AND     polar.leo_polar_grp_included    = rawdata.polargrponeincluded
        JOIN leo_prior_attainment     AS pa
                ON      pa.leo_pr_att_band              = rawdata.prattband
                AND     pa.leo_pr_att_included          = rawdata.prattincluded;

----------------
-- END SCRIPT --
----------------
```

Listing P.1: PL/pgSQL's DML Script for Education Normalized Database Migration.

### P.1.2 Script for Postcode Normalized Database Migration.

```
-- File: 03_yinghua_insert_NSPL_table.sql
-- Date: Fri Jan 12 16:02 MYT 2018
-- Author: Chai Ying Hua
-- Version: 1.0
-- Database: psql (PostgreSQL) 9.5.10
-- =========================================================

DELETE FROM postcode_greek_coordinate;
DELETE FROM postcode_output_area_classification;
DELETE FROM postcode_middle_super_output_area;
DELETE FROM postcode_lower_super_output_area;
DELETE FROM postcode_primary_care_trust;
DELETE FROM postcode_euro_electoral_region;
DELETE FROM postcode_parliament_constituency;
DELETE FROM postcode_region;
DELETE FROM postcode_country;
DELETE FROM postcode_ward;
DELETE FROM postcode_local_authority_county;
DELETE FROM postcode_county;
DELETE FROM postcode_cartesian_coordinate;


-- SELECT UNIQUE DATA FROM RAW TABLE AND INSERT INTO NORMALIZED DATA.
----------------------------------------
-- POSTCODE_GREEK_COORDINATE TABLE MIGRATION
-- ROW COUNTS: 1664728
----------------------------------------
INSERT INTO postcode_greek_coordinate (pos_longitude, pos_latitude)
        SELECT DISTINCT longitude, latitude FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_AREA_OUTPUT_CLASSIFICATION TABLE MIGRATION
-- ROW COUNTS: 77
----------------------------------------
INSERT INTO postcode_output_area_classification (pos_oac_code, pos_oac_name)
        SELECT DISTINCT oacc, oacn FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_MIDDLE_SUPER_OUTPUT_AREA TABLE MIGRATION
-- ROW COUNTS: 8484
----------------------------------------
INSERT INTO postcode_middle_super_output_area (pos_msoa_code, pos_msoa_name)
        SELECT DISTINCT msoac, msoan FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_LOWER_SUPER_OUTPUT_AREA TABLE MIGRATION
-- ROW COUNTS: 42460
----------------------------------------
INSERT INTO postcode_lower_super_output_area (pos_lsoa_code, pos_lsoa_name)
        SELECT DISTINCT isoac, isoan FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_PRIMARY_CARE_TRUST TABLE MIGRATION
-- ROW COUNTS:  200
----------------------------------------
INSERT INTO postcode_primary_care_trust (pos_pct_code, pos_pct_name)
        SELECT DISTINCT pctc, pctn FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_EURO_ELECTORAL_REGION TABLE MIGRATION
-- ROW COUNTS:  15
----------------------------------------
INSERT INTO postcode_euro_electoral_region (pos_eer_code, pos_eer_name)
        SELECT DISTINCT eerc, eern FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_PARLIAMENT_CONSTITUENCY TABLE MIGRATION
-- ROW COUNTS:  653
----------------------------------------
INSERT INTO postcode_parliament_constituency (pos_par_cons_code, pos_par_cons_name)
        SELECT DISTINCT par_cons_code, par_cons_name FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_REGION TABLE MIGRATION
-- ROW COUNTS:  15
----------------------------------------
INSERT INTO postcode_region (pos_region_code, pos_region_name)
        SELECT DISTINCT region_code, region_name FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_COUNTRY TABLE MIGRATION
-- ROW COUNTS:  7
----------------------------------------
```

```
INSERT INTO postcode_country (pos_country_code, pos_country_name)
        SELECT DISTINCT countrycode, countryname FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_WARD TABLE MIGRATION
-- ROW COUNTS:  9115
----------------------------------------
INSERT INTO postcode_ward (pos_ward_code, pos_ward_name)
        SELECT DISTINCT wardcode, wardname FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_LOCAL_AUTHORITY_COUNTY TABLE MIGRATION
-- ROW COUNTS:  394
----------------------------------------
INSERT INTO postcode_local_authority_county (pos_lac_code, pos_lac_name)
        SELECT DISTINCT county_lac, county_lan FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_COUNTY TABLE MIGRATION
-- ROW COUNTS:  34
----------------------------------------
INSERT INTO postcode_county (pos_county_code, pos_county_name)
        SELECT DISTINCT countycode, countyname FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_CARTESIAN_COORDINATE TABLE MIGRATION
-- ROW COUNTS:  1662088
----------------------------------------
INSERT INTO postcode_cartesian_coordinate (pos_easting, pos_northing)
        SELECT DISTINCT easting, northing FROM nspl_rawdata;

----------------------------------------
-- POSTCODE_DETAIL TABLE MIGRATION
-- ROW COUNTS:  1754882                 <--- SAME ROW WITH RAW DATA.
----------------------------------------
INSERT INTO postcode_detail (pos1, pos2, pos3, pos_date_introduce, pos_usertype, pos_cart_coordinate_id, position_quality, pos_spatial_accuracy, pos_location, pos_socrataid, pos_last_upload)
        SELECT postcode1, postcode2, postcode3, date_introduce, usertype, pos_cart_coordinate_id, position_quality, spatial_accuracy, location, socrataid, last_upload
        FROM nspl_rawdata AS rawdata
        JOIN postcode_cartesian_coordinate AS pos_car_coor
                ON rawdata.easting = pos_car_coor.pos_easting
                AND rawdata.northing = pos_car_coor.pos_northing;
```

Listing P.2: PL/pgSQL's DML Script for Postcode Normalized Database Migration.

### P.1.3 Script for Company Normalized Database Migration.

```
-- FILE: 03_yinghua_insert_company_table_DML.sql
-- DATE: Mon Jan 9 17:00 MYT 2018
-- AUTHOR: Chai Ying Hua
-- VERSION: 1.0
-- DATABASE: psql (PostgreSQL) 9.5.10
-- DESCRIPTION:
-- ====================================================
--
--      1. Delete all data in reverse order.
--      2. Migrate all data from raw table into normalized lookup table.
-- =====================================================

-- SELECT UNIQUE DATA FROM RAW TABLE AND INSERT INTO NORMALIZED TABLE.

----------------------------------------
-- COMPANY_RETURNS TABLE MIGRATION
-- ROW COUNTS:  28697
----------------------------------------
INSERT INTO company_returns (com_return_nextduedate, com_return_lastmadeupdate)
        SELECT DISTINCT return_nextduedate, return_lastmadeupdate FROM company_rawdata;

----------------------------------------
-- COMPANY_MORTGAGES TABLE MIGRATION
-- ROW COUNTS: 3710
----------------------------------------
INSERT INTO company_mortgages (com_num_mortchanges, com_num_mortoutstanding, com_num_mortpartsatisfied, com_num_mortsatisfied)
        SELECT DISTINCT nummortcharges, nummortoutstanding, nummortpartsatisfied, nummortsatisfied FROM company_rawdata;
```

```

----------------------------------------
-- COMPANY_SICCODE TABLE MIGRATION
-- ROW COUNTS: 51693
----------------------------------------
INSERT INTO company_siccodes (com_siccode1, com_siccode2, com_siccode3, com_siccode4)
        SELECT DISTINCT siccode1, siccode2, siccode3, siccode4 FROM company_rawdata;

----------------------------------------
-- COMPANY_PARTNERSHIP TABLE MIGRATION
-- ROW COUNTS: 279
----------------------------------------
INSERT INTO company_partnership (com_num_genpartners, com_num_limpartners)
        SELECT DISTINCT numgenpartners, numlimpartners FROM company_rawdata;

----------------------------------------
-- COMPANY_URI TABLE MIGRATION
-- ROW COUNTS: 2033290
----------------------------------------
-INSERT INTO company_uri (com_uri)
        SELECT DISTINCT uri FROM company_rawdata;

----------------------------------------
-- COMPANY_CONF_STMT TABLE MIGRATION
-- ROW COUNTS: 14900
----------------------------------------
INSERT INTO company_conf_stmt (com_conf_stmt_nextduedate, com_conf_stmt_lastmadeupdate)
        SELECT DISTINCT confstmtnextduedate, confstmtlastmadeupdate FROM company_rawdata;

----------------------------------------
-- COMPANY_PREVIOUSNAME TABLE MIGRATION
-- ROW COUNTS: 190185
----------------------------------------
INSERT INTO company_previousname (com_pn1_condate, com_pn1_companyname, com_pn2_condate, com_pn2_companyname
      , com_pn3_condate, com_pn3_companyname, com_pn4_condate, com_pn4_companyname, com_pn5_condate,
      com_pn5_companyname, com_pn6_condate, com_pn6_companyname, com_pn7_condate, com_pn7_companyname,
      com_pn8_condate, com_pn8_companyname, com_pn9_condate, com_pn9_companyname, com_pn10_condate,
      com_pn10_companyname)
        SELECT DISTINCT pn1_condate, pn1_companyname, pn2_condate, pn2_companyname, pn3_condate,
      pn3_companyname, pn4_condate, pn4_companyname, pn5_condate, pn5_companyname, pn6_condate,
      pn6_companyname,pn7_condate, pn7_companyname, pn8_condate, pn8_companyname, pn9_condate,
      pn9_companyname, pn10_condate, pn10_companyname FROM company_rawdata;
```

Listing P.3: PL/pgSQL's DML Script for Company Normalized Database Migration.

Listing P.1, P.2 and P.3 shows PL/pgSQL's DML scripts for Company, Postcode and Education normalized database migration. These scripts retrieve the UNIQUE value from raw data from each columns and stored into the **resources table** created in Appendix M. The SQL scripts use INSERT with SELECT concepts to migrate countless rows of data from legacy table into new storage.

The row counts of each table are displayed and updated into the each Listing P.1, P.2 and P.3.

# P.2 Go programming language based data migration program.

## P.2.1 Postcode data migration program.

### P.2.1.1 Extract Normalized Table Key Field.

```

package main

import (
        "fmt"
        _ "github.com/jinzhu/gorm/dialects/postgres"
)

var (
        detail_id [] int64
        county_id [] int64
        lac_id [] int64
        ward_id [] int64
        country_id [] int64
        region_id [] int64
        par_cons_id [] int64
        eer_id [] int64
        pct_id [] int64
        lsoa_id [] int64
        msoa_id [] int64
        oac_id [] int64
        greek_coordinate_id [] int64
)

func retrieve_detail() {

        rows, err := db.Query("SELECT pos_detail_id FROM nspl_rawdata AS rawdata JOIN postcode_detail AS detail ON detail.pos1 = rawdata.postcode1 AND detail.pos2 = rawdata.postcode2 AND detail.pos3 = rawdata.postcode3 AND detail.pos_date_introduce = rawdata.date_introduce AND detail.pos_usertype = rawdata.usertype AND detail.position_quality = rawdata.position_quality AND detail.pos_spatial_accuracy = rawdata.spatial_accuracy AND      detail.pos_location = rawdata.location AND detail.pos_socrataid = rawdata.socrataid AND detail.pos_last_upload = rawdata.last_upload;" )

        checkErr(err, "Error on query DB")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_detail_id)
                checkErr(err, "Retrieve pos_detail_id key")

                detail_id = append(detail_id, n.pos_detail_id);
        }

        fmt.Printf("Postcode detail: %d \n", len(detail_id))
        defer rows.Close()
}

func retrieve_county() {

        rows, err := db.Query("SELECT pos_county_id FROM nspl_rawdata AS rawdata JOIN postcode_county AS county ON county.pos_county_code = rawdata.countycode AND county.pos_county_name = rawdata.countyname JOIN postcode_local_authority_county AS lac ON lac.pos_lac_code = rawdata.county_lac AND lac.pos_lac_name = rawdata.county_lan;" )

        checkErr(err, "Error on query DB")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_county_id)
                checkErr(err, "Retrieve pos_county_id key")

                county_id = append(county_id, n.pos_county_id);
        }

        fmt.Printf("Postcode county: %d \n", len(county_id))
        defer rows.Close()
}

```

```
func retrieve_local_authority_council() {

        rows, err := db.Query("SELECT pos_lac_id FROM nspl_rawdata AS rawdata JOIN postcode_local_authority_county AS lac ON lac.pos_lac_code = rawdata.county_lac AND lac.pos_lac_name = rawdata.county_lan;" )

        checkErr(err, "Error on query DB")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_lac_id)
                checkErr(err, "Retrieve pos_lac_id key")

                lac_id = append(lac_id, n.pos_lac_id);
        }

        fmt.Printf("Postcode Local Authority Council: %d \n", len(lac_id))
        defer rows.Close()
}

func retrieve_ward() {

        rows, err := db.Query("SELECT pos_ward_id FROM nspl_rawdata AS rawdata JOIN postcode_ward AS ward ON ward.pos_ward_code = rawdata.wardcode AND ward.pos_ward_name = rawdata.wardname;" )
        checkErr(err, "Error on query pos_ward_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_ward_id)
                checkErr(err, "Retrieve pos_ward_id key")

                ward_id = append(ward_id, n.pos_ward_id);
        }

        fmt.Printf("Postcode Ward: %d \n", len(ward_id))
        defer rows.Close()
}

func retrieve_country() {

        rows, err := db.Query("SELECT pos_country_id FROM nspl_rawdata AS rawdata JOIN postcode_country AS country ON country.pos_country_code = rawdata.countrycode AND country.pos_country_name = rawdata.countryname;" )
        checkErr(err, "Error on query pos_country_id")

        for rows.Next() {
                var n postcode_id

                err = rows.Scan(&n.pos_country_id)
                checkErr(err, "Retrieve pos_country_id key")

                country_id = append(country_id, n.pos_country_id);
        }

        fmt.Printf("Postcode Country: %d \n", len(country_id))
        defer rows.Close()
}

func retrieve_region() {

        rows, err := db.Query("SELECT pos_region_id FROM nspl_rawdata AS rawdata JOIN postcode_region AS region ON region.pos_region_code = rawdata.region_code AND region.pos_region_name = rawdata.region_name;" )
        checkErr(err, "Error on query pos_region_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_region_id)
                checkErr(err, "Retrieve pos_region_id key")

                region_id = append(region_id, n.pos_region_id);
        }

        fmt.Printf("Postcode Region: %d \n", len(region_id))
        defer rows.Close()
}

func retrieve_parliament_constituency() {

        rows, err := db.Query("SELECT pos_par_cons_id FROM nspl_rawdata AS rawdata JOIN postcode_parliament_constituency AS ppc ON ppc.pos_par_cons_code = rawdata.par_cons_code AND ppc.pos_par_cons_name = rawdata.par_cons_name;" )
        checkErr(err, "Error on query pos_par_cons_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_par_cons_id)
                checkErr(err, "Retrieve pos_par_cons_id key")

                par_cons_id = append(par_cons_id, n.pos_par_cons_id);
        }

        fmt.Printf("Postcode Parliament Constituency: %d \n", len(par_cons_id))
        defer rows.Close()
}

func retrieve_euro_electoral_region() {

        rows, err := db.Query("SELECT pos_eer_id FROM nspl_rawdata AS rawdata JOIN postcode_euro_electoral_region AS eer ON eer.pos_eer_code = rawdata.eerc AND eer.pos_eer_name = rawdata.eern;" )
        checkErr(err, "Error on query pos_eer_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_eer_id)
                checkErr(err, "Retrieve pos_eer_id key")

                eer_id = append(eer_id, n.pos_eer_id);
        }

        fmt.Printf("Postcode Euro Electoral Region: %d \n", len(eer_id))
        defer rows.Close()
}

func retrieve_primary_care_trust() {

        rows, err := db.Query("SELECT pos_pct_id FROM nspl_rawdata AS rawdata JOIN postcode_primary_care_trust AS pct ON pct.pos_pct_code = rawdata.pctc AND pct.pos_pct_name = rawdata.pctn;" )
        checkErr(err, "Error on query pos_pct_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_pct_id)
                checkErr(err, "Retrieve pos_pct_id key")

                pct_id = append(pct_id, n.pos_pct_id);
        }

        fmt.Printf("Postcode Primary Care Trust: %d \n", len(pct_id))
        defer rows.Close()
}

func retrieve_lower_super_output_area () {

        rows, err := db.Query("SELECT pos_lsoa_id FROM nspl_rawdata AS rawdata JOIN postcode_lower_super_output_area AS lsoa ON lsoa.pos_lsoa_code = rawdata.isoac AND lsoa.pos_lsoa_name = rawdata.isoan;" )
        checkErr(err, "Error on query pos_lsoa_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_lsoa_id)
                checkErr(err, "Retrieve pos_lsoa_id key")

                lsoa_id = append(lsoa_id, n.pos_lsoa_id);
        }

        fmt.Printf("Postcode Lower Super Output Area: %d \n", len(lsoa_id))
        defer rows.Close()
}

func retrieve_middle_super_output_area() {

        rows, err := db.Query("SELECT pos_msoa_id FROM nspl_rawdata AS rawdata JOIN postcode_middle_super_output_area AS msoa ON msoa.pos_msoa_code = rawdata.msoac AND msoa.pos_msoa_name = rawdata.msoan;" )
        checkErr(err, "Error on query pos_msoa_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_msoa_id)
                checkErr(err, "Retrieve pos_msoa_id key")

                msoa_id = append(msoa_id, n.pos_msoa_id);
        }

        fmt.Printf("Postcode Middle Super Output Area: %d \n", len(msoa_id))
        defer rows.Close()
}

func retrieve_output_area_classification() {

        rows, err := db.Query("SELECT pos_oac_id FROM nspl_rawdata AS rawdata JOIN postcode_output_area_classification AS oac ON oac.pos_oac_code = rawdata.oacc AND oac.pos_oac_name = rawdata.oacn;" )
        checkErr(err, "Error on query pos_oac_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_oac_id)
                checkErr(err, "Retrieve pos_oac_id key")

                oac_id = append(oac_id, n.pos_oac_id);
        }

        fmt.Printf("Postcode Output Area Classification: %d \n", len(oac_id))
        defer rows.Close()
}

func retrieve_greek_coordinate() {

        rows, err := db.Query("SELECT pos_greek_coordinate_id FROM nspl_rawdata AS rawdata JOIN postcode_greek_coordinate AS pgc ON pgc.pos_longitude = rawdata.longitude AND pgc.pos_latitude = rawdata.latitude;" )
        checkErr(err, "Error on query pos_greek_coordinate_id")

        for rows.Next() {

                var n postcode_id

                err = rows.Scan(&n.pos_greek_coordinate_id)
                checkErr(err, "Retrieve pos_greek_coordinate_id key")

                greek_coordinate_id = append(greek_coordinate_id, n.pos_greek_coordinate_id);
        }

        fmt.Printf("Postcode Greek Coordinate: %d \n", len(greek_coordinate_id))
        defer rows.Close()
}
```

Listing P.4: Resource Table Key Retrieval Function.

Listing P.4 shows the source code for resource table key retrieval. Each function is used specifically to retrieve primary key of specific resource table and stored in dedicated array with **append**. Once each function had finish executed, these arrays contain extracted PRIMARY KEY (PK) and await to be insert into the another table as FOREIGN KEY (FK). This process is called *referential integrity.*

#### P.2.1.2 Migrate data with Referencial Integrity.

```

package main

import (
        "log"
        "fmt"
        _ "github.com/jinzhu/gorm/dialects/postgres"
        "database/sql"
        "sync"
)

const (
        DB_USER     = "yinghua"
        DB_PASSWORD = "123"
        DB_NAME     = "postcode"
        ENTRIES     = 1754882
)

var (
        db *sql.DB
        sqlStatement = "INSERT INTO postcode (pos_detail_id, pos_county_id, pos_lac_id, pos_ward_id, pos_country_id, pos_region_id, pos_par_cons_id, pos_eer_id, pos_pct_id, pos_lsoa_id, pos_msoa_id, pos_oac_id, pos_greek_coordinate_id) values ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13);";

)

//================================================
//function to check error and print error messages
//================================================
func checkErr(err error, message string) {
        if err != nil {
                panic(message + " err: " + err.Error())
        }
}

//================================================
// initialize connection with database
//================================================
func initDB() {

        dbInfo := fmt.Sprintf("user=%s password=%s dbname=%s sslmode=disable",
        DB_USER, DB_PASSWORD, DB_NAME)
        sqldb, err := sql.Open("postgres", dbInfo)
        checkErr(err, "Initialize database")

        sqldb.SetMaxOpenConns(90)
        db = sqldb
}

func main() {
        initDB()
        retrieve_key()
        insert_key()

        fmt.Println("The postcode data had migrated complete")
}

=========================================================================
Function that migrate all the keys into Postcode table with Reference Integrity
=========================================================================
func insert_key() {

        fmt.Println("Begin to migrate postcode data")

        stmt, err := db.Prepare(sqlStatement)
        checkErr(err, "Prepare insert statement")

        wg := sync.WaitGroup{}

        // ensure all routines finish before returning
        defer wg.Wait()

        for i := ENTRIES; i > 0 ; i-- {
                wg.Add(1)
                go func () {
                        defer wg.Done()
                        res, err := stmt.Exec(detail_id[i],  county_id[i], lac_id[i], ward_id[i], country_id[i], region_id[i], par_cons_id[i], eer_id[i], pct_id[i], lsoa_id[i], msoa_id[i], oac_id[i], greek_coordinate_id[i])
                        checkErr(err, "Insert statement execution error")

                        if res == nil {
                                log.Fatal(err)
                        }
                }()
```

```
 82          }
 83  }
 84
 85  ==================================================================================
 86  Function that retrieve all the key from Normalized Resource table and stored into array
 87  ==================================================================================
 88  func retrieve_key() {
 89          retrieve_detail()
 90          retrieve_county()
 91          retrieve_local_authority_council()
 92          retrieve_ward()
 93          retrieve_country()
 94          retrieve_region()
 95          retrieve_parliament_constituency()
 96          retrieve_euro_electoral_region()
 97          retrieve_primary_care_trust()
 98          retrieve_lower_super_output_area()
 99          retrieve_middle_super_output_area()
100          retrieve_output_area_classification()
101          retrieve_greek_coordinate()
102  }
```

Listing P.5: Postcode Data Migration main program.

Listing P.5 shows the source code for postcode data migration main program. The main function is where **a program start its execution**.

When the main program is compiled and executed, main() will called **retrieve_key()** function to retrieve all the PRIMARY KEY (PK) from each resource table and stored into dedicated array (refer row 50). Once the process had finished executed, the **insert_key*()** function will be execute to retrieved these key values in array and migrated into the postcode table (refer row 71-82). The PK in array is insert into another table as FOREIGN KEY (FK) to establish relationship between entity.

The **insert_key()** function use channels to synchronize migration execution across goroutines to form an concurrent execution. The synchronization primitives of Go programming language is used to perform communication in mutual exclusion locks. The entire execution of this function will be display and print on terminal (refer row 53).

The result obtained will be tabulated, compared and discussed.

## P.2.2 Company data migration program

### P.2.2.1 Extract Normalized Table Key Field.

```
package main

import (
        "fmt"
        _ "github.com/jinzhu/gorm/dialects/postgres"
)

func retrieve_detail_id() {
        fmt.Println("Begin to retrieve company_detail_id from company_detail")
        rows, err := db.Query("SELECT com_detail_id FROM company_detail;")
        checkErr(err, "Error on query com_detaiL_id statement")

        var (
                com_detail_id int
        )

        for rows.Next() {
                err = rows.Scan(&com_detail_id)
                checkErr(err, "Retrieve com_detail_id")

                com_detail_idArray = append(com_detail_idArray, com_detail_id)
        }

        fmt.Printf("Company detail id: %d \n", len(com_detail_idArray))
        defer rows.Close()
}

func retrieve_normal_detail() {
        fmt.Println("Begin to retrieve normal detail from company_rawdata")
        rows, err := db.Query("SELECT dissolutiondate, incorporationdate, countryoforigin, careof, pobox, addressline1, addressline2, posttown, county, country, postcode FROM company_rawdata;")
        checkErr(err, "Error on query normal detail statement")

        var (
                dissolutiondate string
                incorporationdate string
                countryoforigin string
                careof string
                pobox string
                addressline1 string
                addressline2 string
                posttown string
                county string
                country string
                postcode string
        )

        for rows.Next() {
                err = rows.Scan(&dissolutiondate, &incorporationdate, &countryoforigin, &careof, &pobox, &addressline1, &addressline2, &posttown, &county, &country, &postcode)
                checkErr(err, "Retrieve company normal detail")

                dissolutiondateArray = append(dissolutiondateArray, dissolutiondate)
                incorporatedateArray = append(incorporatedateArray, incorporationdate)
                countryoforiginArray = append(countryoforiginArray, countryoforigin)
                careofArray = append(careofArray, careof)
                poboxArray = append(poboxArray, pobox)
                addressline1Array = append(addressline1Array, addressline1)
                addressline2Array = append(addressline2Array, addressline2)
                posttownArray = append(posttownArray, posttown)
                countyArray = append(countyArray, county)
                countryArray = append(countryArray, country)
                postcodeArray = append(postcodeArray, postcode)
        }

        fmt.Printf("Dissolution date: %d \n", len(dissolutiondateArray))
        fmt.Printf("Incorporationdate: %d \n", len(incorporatedateArray))
        fmt.Printf("Country of origin: %d \n", len(countryoforiginArray))
        fmt.Printf("Careof: %d \n", len(careofArray))
        fmt.Printf("Pobox: %d \n", len(poboxArray))
        fmt.Printf("Address line 1: %d \n", len(addressline1Array))
        fmt.Printf("Address line 2: %d \n", len(addressline2Array))
        fmt.Printf("Post town: %d \n", len(posttownArray))
        fmt.Printf("County: %d \n", len(countyArray))
        fmt.Printf("Country: %d \n", len(countryArray))
        fmt.Printf("Postcode: %d \n", len(postcodeArray))

        defer rows.Close()
}

func retrieve_account_id() {
        fmt.Println("Begin to retrieve com_acc_id from company_account")
        rows, err := db.Query("SELECT com_acc_id FROM company_account;")
        checkErr(err, "Error on query com_acc_id statement")

        var (
                com_acc_id int
        )

        for rows.Next() {
                err = rows.Scan(&com_acc_id)
                checkErr(err, "Retrieve com_detail_id")

                com_acc_idArray = append(com_acc_idArray, com_acc_id)
        }

        fmt.Printf("Company account id: %d \n", len(com_acc_idArray))
        defer rows.Close()
}

func retrieve_returns_id() {
        fmt.Println("Begin to retrieve com_return_id from company_returns")
        rows, err := db.Query("SELECT com_return_id FROM company_returns AS return JOIN company_rawdata AS raw ON raw.return_nextduedate = return.com_return_nextduedate AND raw.return_lastmadeupdate = return.com_return_lastmadeupdate;")
        checkErr(err, "Error on query com_return_id statement")

        var com_return_id int

        for rows.Next() {
                err = rows.Scan(&com_return_id)
                checkErr(err, "Retrieve com_return_id")

                com_return_idArray = append(com_return_idArray, com_return_id)
        }

        fmt.Printf("Company return id: %d \n", len(com_return_idArray))
        defer rows.Close()
}

func retrieve_mort_id() {
        fmt.Println("Begin to retrieve com_mort_id from company_mortgages")
        rows, err := db.Query("SELECT com_mort_id FROM company_mortgages AS mort JOIN company_rawdata AS raw ON mort.com_num_mortchanges = raw.nummortcharges AND mort.com_num_mortoutstanding = raw.nummortoutstanding AND mort.com_num_mortpartsatisfied = raw.nummortpartsatisfied AND mort.com_num_mortsatisfied = raw.nummortsatisfied;")
        checkErr(err, "Error on query com_mort_id statement")

        var com_mort_id int

        for rows.Next() {
                err = rows.Scan(&com_mort_id)
                checkErr(err, "Retrieve com_mort_id")

                com_mort_idArray = append(com_mort_idArray, com_mort_id)
        }

        fmt.Printf("Company mort id: %d \n", len(com_mort_idArray))
        defer rows.Close()
}

func retrieve_sic_id() {
        fmt.Println("Begin to retrieve com_sic_id from company_siccodes")
        rows, err := db.Query("SELECT com_sic_id FROM company_siccodes AS sic JOIN company_rawdata AS raw ON sic.com_siccode1 = raw.siccode1 AND sic.com_siccode2 = raw.siccode2 AND raw.siccode3 = sic.com_siccode3 AND raw.siccode4 = sic.com_siccode4;")
        checkErr(err, "Error on query com_sic_id statement")

        var com_sic_id int

        for rows.Next() {
                err = rows.Scan(&com_sic_id)
                checkErr(err, "Retrieve com_sic_id")

                com_sic_idArray = append(com_sic_idArray, com_sic_id)
        }

        fmt.Printf("Company mort id: %d \n", len(com_sic_idArray))
        defer rows.Close()
}

func retrieve_partnership_id() {
        fmt.Println("Begin to retrieve com_partnership_id from company_partnership")
        rows, err := db.Query("SELECT com_partnership_id FROM company_partnership AS part JOIN company_rawdata AS raw ON raw.numgenpartners = part.com_num_genpartners AND raw.numlimpartners = part.com_num_limpartners;")
        checkErr(err, "Error on query com_partnership_id statement")

        var com_partnership_id int

        for rows.Next() {
                err = rows.Scan(&com_partnership_id)
                checkErr(err, "Retrieve com_sic_id")

                com_partnership_idArray = append(com_partnership_idArray, com_partnership_id)
        }

        fmt.Printf("Company partnership: %d \n", len(com_partnership_idArray))
        defer rows.Close()
}

func retrieve_uri_id() {
        fmt.Println("Begin to retrieve com_uri_id from company_uri")
        rows, err := db.Query("SELECT com_uri_id FROM company_uri AS uri JOIN company_rawdata AS raw ON uri.com_uri = raw.uri;")
        checkErr(err, "Error on query com_uri_id statement")

        var com_uri_id int

        for rows.Next() {
                err = rows.Scan(&com_uri_id)
                checkErr(err, "Retrieve com_uri_id")

                com_uri_idArray = append(com_uri_idArray, com_uri_id)
        }

        fmt.Printf("Company uri: %d \n", len(com_uri_idArray))
        defer rows.Close()
}


func retrieve_previousname_id() {
        fmt.Println("Begin to retrieve com_pn_id from company_previousname")
        rows, err := db.Query("SELECT com_pn_id FROM company_rawdata AS raw JOIN company_previousname AS pn
     ON raw.pn1_condate = pn.com_pn1_condate AND raw.pn1_companyname = pn.com_pn1_companyname AND raw.
     pn2_condate = pn.com_pn2_condate AND raw.pn2_companyname = pn.com_pn2_companyname AND raw.pn3_condate
     = pn.com_pn3_condate AND raw.pn3_companyname = pn.com_pn3_companyname AND raw.pn4_condate = pn.
     com_pn4_condate AND raw.pn4_companyname = pn.com_pn4_companyname AND raw.pn5_condate = pn.
     com_pn5_condate AND raw.pn5_companyname = pn.com_pn5_companyname AND raw.pn6_condate = pn.
     com_pn6_condate AND raw.pn6_companyname = pn.com_pn6_companyname AND raw.pn7_condate = pn.
     com_pn7_condate AND raw.pn7_companyname = pn.com_pn7_companyname AND raw.pn8_condate = pn.
     com_pn8_condate AND raw.pn8_companyname = pn.com_pn8_companyname AND raw.pn9_condate = pn.
     com_pn9_condate AND raw.pn9_companyname = pn.com_pn9_companyname AND raw.pn10_condate = pn.
     com_pn10_condate;")
        checkErr(err, "Error on query com_pn_id statement")

        var com_pn_id int

        for rows.Next() {
                err = rows.Scan(&com_pn_id)
                checkErr(err, "Retrieve com_pn_id")

                com_previousname_idArray = append(com_previousname_idArray, com_pn_id)
        }

        fmt.Printf("Company previousname: %d \n", len(com_previousname_idArray))
        defer rows.Close()
}

func retrieve_conf_stmt_id() {
        fmt.Println("Begin to retrieve com_conf_stmt_id from company_conf_stmt")
        rows, err := db.Query("SELECT com_conf_stmt_id FROM company_conf_stmt AS stmt JOIN company_rawdata
     AS raw ON stmt.com_conf_stmt_nextduedate = raw.confstmtnextduedate AND stmt.
     com_conf_stmt_lastmadeupdate = raw.confstmtlastmadeupdate;")
        checkErr(err, "Error on query com_pn_id statement")

        var com_conf_stmt_id int

        for rows.Next() {
                err = rows.Scan(&com_conf_stmt_id)
                checkErr(err, "Retrieve com_conf_stmt_id")

                com_conf_stmt_idArray = append(com_conf_stmt_idArray, com_conf_stmt_id)
        }

        fmt.Printf("Company conference statement: %d \n", len(com_conf_stmt_idArray))
        defer rows.Close()
}
```

Listing P.6: Resource Table Key Retrieval Function.

Listing P.6 shows the source code for company resource table key retrieval. Each function is used specifically to retrieve primary key of specific resource table and stored in dedicated array with **append**. Once each function had finish executed, these arrays contain extracted PRIMARY KEY (PK) and await to be insert into the another table as FOREIGN KEY (FK). This process is called *referential integrity*.

### P.2.2.2 Migrate data with Referencial Integrity.

```
package main

import (
        "fmt"
        _ "github.com/jinzhu/gorm/dialects/postgres"
        "time"
```

```
 7  )
 8
 9  func retrieve_key_from_normalized_table(){
10          initDB()
11          retrieve_detail_id()
12          retrieve_normal_detail()
13          retrieve_account_id()
14          retrieve_returns_id()
15          retrieve_mort_id()
16          retrieve_sic_id()
17          retrieve_partnership_id()
18          retrieve_uri_id()
19          retrieve_conf_stmt_id()
20          retrieve_previousname_id()
21  }
22
23  func import_company_table() {
24
25          start := time.Now()
26          retrieve_key_from_normalized_table()
27          insert_company_table()
28          fmt.Printf("%.5fs seconds on import company. \n", time.Since(start).Seconds())
29  }
30
31  func insert_company_table() {
32          sem := make (chan bool, CONCURRENCY)
33
34          fmt.Println("Begin to insert company data")
35          var sqlStatement = "INSERT INTO company (com_detail_id, com_dissolutiondate, com_incorporationdate,
        com_countryoforigin, com_careof, com_pobox, com_addressline1, com_addressline2, com_posttown,
        com_county, com_country, com_postcode, com_acc_id, com_return_id, com_mort_id, com_sic_id,
        com_partnership_id, com_uri_id, com_pn_id, com_conf_stmt_id) VALUES ($1, $2, $3, $4, $5, $6, $7, $8,
        $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20);"
36
37          stmt, err := db.Prepare(sqlStatement)
38          checkErr(err, "Prepare insert company")
39
40          for i := len(dissolutiondateArray); i > 0; i-- {
41                  sem <- true
42                  go func () {
43                  defer func() {<-sem}()
44                          _, err := stmt.Exec(com_detail_idArray[i], dissolutiondateArray[i],
        incorporatedateArray[i], countryoforiginArray[i], careofArray[i], poboxArray[i], addressline1Array[i],
         addressline2Array[i], posttownArray[i], countyArray[i],
45                          countryArray[i], postcodeArray[i], com_acc_idArray[i], com_return_idArray[i],
        com_mort_idArray[i],
46                          com_sic_idArray[i], com_partnership_idArray[i], com_uri_idArray[i],
        com_previousname_idArray[i], com_conf_stmt_idArray[i])
47                          checkErr(err, "Insert statement execution error")
48                  }()
49          }
50
51          for i := 0 ; i < cap(sem); i++ {
52                  sem <- true
53          }
54  }
```

Listing P.7: Resource Table Key Retrieval Function.

Listing P.7 shows the source code for company data migration main program. The main function is where **a program start its execution**.

When the main program is compiled and executed, main() will called **retrieve_key_from_normalized_table()** function to retrieve all the PRIMARY KEY (PK) from each resource table and stored into dedicated array (refer row 9 to 20). Once the process had finished executed, the **insert_company_table()** function will be execute to retrieved these key values in array and migrated into the postcode table (refer row 31-54). The PK in array is insert into another table as FOREIGN KEY (FK) to establish relationship between entity.

The **insert_key()** function use *Semaphore* to control the access of 400,000 *Goroutines* on common resource provided by PostgreSQL database and operating system environment. The concurrency of data migration execution in this program are controlled and limited to prevent race condition. These Goroutines communicate with each other with flag to utilized 299 open connection with PostgreSQL database on migrating 3.5 millions of data with specific resource provided.

The result obtained will be tabulated, compared and discussed.

## P.3 List of database relation

### P.3.1 List Company Database Table Size

```
 Schema |           Name           |   Type   |  Owner  |    Size    | Line counts
--------+--------------------------+----------+---------+------------+-------------
 public | company                  | table    | yinghua | 725 MB     | 3595702  <-- same counts
 public | company_account          | table    | yinghua | 262 MB     | 3595702
 public | company_account_category | table    | yinghua | 8192 bytes | 16
 public | company_category         | table    | yinghua | 8192 bytes | 21
 public | company_conf_stmt        | table    | yinghua | 904 kB     | 14900
 public | company_detail           | table    | yinghua | 300 MB     | 3595702
 public | company_mortgages        | table    | yinghua | 216 kB     | 3710
 public | company_partnership      | table    | yinghua | 40 kB      | 279
 public | company_previousname     | table    | yinghua | 48 MB      | 190185
 public | company_rawdata          | table    | yinghua | 2476 MB    | 3595702  <-- same counts
 public | company_returns          | table    | yinghua | 1720 kB    | 28697
 public | company_siccodes         | table    | yinghua | 9872 kB    | 51693
 public | company_status           | table    | yinghua | 32 kB      | 14
 public | company_uri              | table    | yinghua | 164 MB     | 2033290
```

Listing P.8: List size of company normalized table.

Listing P.8 shows all the database relation found in Company database. The result shows the normalized entity are migrated successfully based on Entity Relationship Diagram database design with PL/pgSQL's DDL scripts and Go migration program. The normalized table (company) has smaller size compare to original datasets (company_rawdata). Moreover, the data does not loss and missing after the data migration execution is completed.

### P.3.2 List Postcode Database Table Size

```
 Schema |                Name                |   Type   |  Owner  |    Size    | Line Counts
--------+------------------------------------+----------+---------+------------+-------------
 public | nspl_rawdata                       | table    | yinghua | 1403 MB    | 1754882    <- same counts
 public | postcode                           | table    | yinghua | 152 MB     | 1754882    <- same counts
 public | postcode_cartesian_coordinate      | table    | yinghua | 70 MB      | 1662088
 public | postcode_country                   | table    | yinghua | 8192 bytes | 7
 public | postcode_county                    | table    | yinghua | 8192 bytes | 34
 public | postcode_detail                    | table    | yinghua | 225 MB     | 1754882
 public | postcode_euro_electoral_region     | table    | yinghua | 8192 bytes | 15
 public | postcode_greek_coordinate          | table    | yinghua | 70 MB      | 1664728
 public | postcode_local_authority_county    | table    | yinghua | 48 kB      | 394
 public | postcode_lower_super_output_area   | table    | yinghua | 2560 kB    | 42460
 public | postcode_middle_super_output_area  | table    | yinghua | 528 kB     | 8484
 public | postcode_output_area_classification | table    | yinghua | 8192 bytes | 77
 public | postcode_parliament_constituency   | table    | yinghua | 64 kB      | 653
 public | postcode_primary_care_trust        | table    | yinghua | 40 kB      | 200
 public | postcode_region                    | table    | yinghua | 8192 bytes | 15
 public | postcode_ward                      | table    | yinghua | 544 kB     | 9115
```

Listing P.9: List size of Postcode normalized table.

Listing P.9 shows all the database relation found in Postcode database. The result shows the normalized entity are migrated successfully with PL/pgSQL's DML scripts and Go migration program. The normalized table (postcode) has smaller size compare to original datasets (nspl_rawdata). Moreover, the data does not loss and missing after the data migration execution is completed.

### P.3.3 List Education Database Table Size

```
 Schema |          Name          |   Type   |  Owner  |    Size    | Line Counts
--------+------------------------+----------+---------+------------+-------------
 public | leo                    | table    | yinghua | 4400 kB    | 32706      <- Same counts
 public | leo_detail             | table    | yinghua | 3680 kB    | 32706
 public | leo_earning            | table    | yinghua | 872 kB     | 14372
 public | leo_graduation         | table    | yinghua | 8192 bytes | 195
 public | leo_match              | table    | yinghua | 200 kB     | 3992
 public | leo_polar              | table    | yinghua | 320 kB     | 6793
 public | leo_prior_attainment   | table    | yinghua | 120 kB     | 2139
 public | leo_rawdata            | table    | yinghua | 5064 kB    | 32706      <- Same counts
 public | leo_sustain_employment | table    | yinghua | 336 kB     | 6192
 public | leo_uncaptured         | table    | yinghua | 296 kB     | 6283
```

Listing P.10: List size of Education normalized table.

Listing P.10 shows all the database relation found in Education database. The result shows the normalized entity are migrated successfully based on Entity Relationship Diagram database design with PL/pgSQL's DML scripts. The normalized table (leo) has smaller size compare to original datasets (leo_rawdata). Moreover, the data does not loss and missing after the data migration execution is completed.

## P.4 Execution of Company Migration Program

```
=========================
Step 1 - Change Directory
=========================
yinghua@yinghua:~$ cd gitRepo/go-import-company/src/main
yinghua@yinghua:~/gitRepo/go-import-company/src/main$

=========================
Step 2 - Compile and Run
=========================
yinghua@yinghua:~/gitRepo/go-import-company/src/main$ go build *.go
yinghua@yinghua:~/gitRepo/go-import-company/src/main$ time go run *.go

------------------------
Import company_uri data
------------------------
Begin to retrieve uri from company_rawdata
Company URI: 2033290
Begin to insert company_uri data
258.93969s seconds on import uri.

------------------------
Import company_partnership data
------------------------
Begin to retrieve partnership from company_rawdata
General partner: 279
Limited partner: 279
Begin to insert company_partnership data
2.71493s seconds on import partnership.

------------------------
Import company_mortgages data
------------------------
Begin to retrieve mortgages from company_rawdata
Mort charges: 3710
Mort outstanding: 3710
Mort partsatisfied: 3710
mort satisfied: 3710
Begin to insert company_mortgages data
5.16182s seconds on import mortgages.

------------------------
Import company_returns data
------------------------
Begin to retrieve returns from company_rawdata
Return next due date: 28697
Return last made update: 28697
Begin to insert company_returns data
14.24606s seconds on import returns.

------------------------
Import company_account_category data
------------------------
Begin to retrieve account category from company_rawdata
Category: 16
Begin to insert company_account_category data
1.56320s seconds on import account category.

------------------------
Import company_account data
------------------------
Begin to retrieve account from company_rawdata
Ref day : 3595702
Ref month: 3595702
Account nextduedate: 3595702
Account lastmadeupdate: 3595702
Category ID: 3595702
Begin to insert company_account data
2867.11349s seconds on import account.

------------------------
Import company_conf_stmt data
------------------------
Begin to retrieve conference statement from company_rawdata
Conference Statement next due date : 14900
Conference Statement last made update: 14900
Begin to insert company_conf_stmt data
14.31405s seconds on import conference statement.

------------------------
Import company_address data
------------------------
Begin to retrieve address from company_rawdata
Care of: 1419715
PO Box: 1419715
Address Line 1: 1419715
Address Line 2: 1419715
Post town: 1419715
County: 1419715
Country: 1419715
Postcode: 1419715
Begin to insert company_address data
181.64420s seconds on import address statement.

------------------------
Import company_countryoforigin data
------------------------
Begin to retrieve countryoforigin from company_rawdata
Country of origin: 196
Begin to insert company_countryoforigin data
2.43293s seconds on import countryoforigin statement.

------------------------
Import company_status data
------------------------
Begin to retrieve companystatus from company_rawdata
Company status: 14
Begin to insert com_status data
22.42986s seconds on import companystatus statement.

------------------------
Import company_category data
------------------------
Begin to retrieve companycategory from company_rawdata
Company category: 21
Begin to insert com_status data
1.39370s seconds on import company category statement.

------------------------
Import company_siccodes data
------------------------
Begin to retrieve siccode from company_rawdata
SIC code 1: 51693
SIC code 2: 51693
SIC code 3: 51693
SIC code 4: 51693
Begin to insert com_status data
16.41218s seconds on import companysiccode statement.

------------------------
Import company_previousname data
------------------------
Begin to retrieve previousdate from company_rawdata
Company change of date 1: 190185
Company change name 1: 190185
Company change of date 2: 190185
Company change name 2: 190185
Company change of date 3: 190185
Company change name 3: 190185
Company change of date 4: 190185
Company change name 4: 190185
Company change of date 5: 190185
Company change name 5: 190185
Company change of date 6: 190185
Company change name 6: 190185
Company change of date 7: 190185
Company change name 7: 190185
Company change of date 8: 190185
Company change name 8: 190185
Company change of date 9: 190185
Company change name 9: 190185
Company change of date 10: 190185
Company change name 10: 190185
Begin to insert company_previousname data
87.43327s seconds on import company previousdate statement.

------------------------
Import company_detail data
------------------------
Begin to retrieve companydetail from company_rawdata
Company name: 3595702
Company number: 3595702
Company category id: 3595702
Company status id: 3595702
Begin to insert company_detail data
7500.89631s seconds on import companydetail statement.

------------------------
Import company detail data
------------------------
Begin to retrieve company_detail_id from company_detail
Company detail id: 3595702

------------------------
Migrate company data
```

```
------------------------
Begin to retrieve normal detail from company_rawdata
Dissolution date: 3595702
Incorporationdate: 3595702
Country of origin: 3595702
Careof: 3595702
Pobox: 3595702
Address line 1: 3595702
Address line 2: 3595702
Post town: 3595702
County: 3595702
Country: 3595702
Postcode: 3595702

Begin to retrieve com_acc_id from company_account
Company account id: 3595702
Begin to retrieve com_return_id from company_returns
Company return id: 3595702
Begin to retrieve com_mort_id from company_mortgages
Company mort id: 3595702
Begin to retrieve com_sic_id from company_siccodes
Company mort id: 3595702
Begin to retrieve com_partnership_id from company_partnership
Company partnership: 3595702
Begin to retrieve com_uri_id from company_uri
Company uri: 3595702
Begin to retrieve com_conf_stmt_id from company_conf_stmt
Company conference statement: 3595702
Begin to retrieve com_pn_id from company_previousname
Company previousname: 3595702

Begin to insert company data
14821.83897s seconds on import company.
```

Listing P.11: Execution of Company Migration Program.

# Appendix Q

# Data Retrieval Results.

## Q.1 Result for Go program for CSV file data retrieval.

### Q.1.1 Go Sequential program vs Go Concurrent program.

| **CSV Datasets** | **Sequential Duration (s)** | **Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.09179 | 0.12362 |
| Company | 32.64937 | 36.22334 |
| Postcode (NSPL) | 13.07156 | 15.21926 |
| **Total** | **45.81286** | **36.22355** |

Table Q.1: Phase 2 Go Sequential program vs Go Concurrent program.

Table Q.1 show the table of results comparison between Go sequential program and concurrent program in CSV file data retrieval. The total elapsed time of concurrent program is faster than sequential program in retrieving 4 millions of data from three CSV files.

## Q.2 Result for Rust program for CSV file data retrieval.

### Q.2.1 Rust Sequential program vs Rust Concurrent program.

| **CSV Datasets** | **Sequential Duration (s)** | **Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.90461 | 1.038585794 |
| Company | 292.70488175 | 314.530471492 |
| Postcode (NSPL) | 109.972792579 | 116.362977683 |
| **Total** | **403.582455002** | **314.530967308** |

Table Q.2: Phase 2 Rust Sequential program vs Rust Concurrent program.

Table Q.2 show the table of results comparison between Rust sequential program and Rust concurrent program in CSV file data retrieval. The total elapsed time of concurrent program is faster than sequential program in retrieving 4 millions of data from three CSV files.

## Q.3 Result for Go program for PostgreSQL data retrieval.

### Q.3.1 Go Sequential program vs Go Concurrent program.

| **PostgreSQL table** | **Sequential Duration (s)** | **Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.22304 | 0.5291 |
| Company | 39.8771 | 43.36509 |
| Postcode (NSPL) | 39.8771 | 14.52721 |
| **Total** | **52.06485** | **43.36518** |

Table Q.3: Phase 2 Go Sequential program vs Go Concurrent program.

Table Q.3 show the table of results comparison between Go sequential program and Go concurrent program in PostgreSQL database retrieval. The total elapsed time of concurrent program is faster than sequential program in retrieving 4 millions of data from three tables in PostgreSQL database.

## Q.4 Result for Rust program for PostgreSQL data retrieval.

### Q.4.1 Rust Sequential program vs Rust Concurrent program.

| **PostgreSQL table** | **Sequential Duration (s)** | **Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.720544494 | 0.789323246 |
| Company | 172.584919465 | 181.387234079 |
| Postcode (NSPL) | 60.442268738 | 65.702471599 |
| **Total** | **233.752923612** | **181.389403179** |

Table Q.4: Phase 2 Rust Sequential program vs Rust Concurrent program.

Table Q.4 show the table of results comparison between Rust sequential program and Rust concurrent program in PostgreSQL database retrieval. The total elapsed time of concurrent program is faster than sequential program in retrieving 4 millions of data from three tables in PostgreSQL database.

## Q.5 Comparison of concurrent programming languages performance.

### Q.5.1 Go CSV Concurrent program vs Rust CSV Concurrent program.

| **CSV Datasets** | **Go Concurrent Duration (s)** | **Rust Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.12362 | 1.038586 |
| Company | 36.22334 | 314.53047 |
| Postcode (NSPL) | 15.21926 | 116.36298 |
| **Total** | **36.22355** | **314.53097** |

Table Q.5: Phase 2 Go CSV Concurrent program vs Rust CSV Concurrent program.

Table Q.5 show the table of results comparison between Go concurrent program and Rust concurrent program in CSV file data retrieval. The total elapsed time of Go concurrent program is faster than Rust concurrent program in retrieving 4 millions of data from three datasets in CSV file.

### Q.5.2 Go PostgreSQL Concurrent program vs Rust PostgreSQL Concurrent program.

| **PostgreSQL table** | **Go Concurrent Duration (s)** | **Rust Concurrent Duration (s)** |
|---|---|---|
| Education (LEO) | 0.5291 | 0.78932 |
| Company | 43.36509 | 181.38723 |
| Postcode (NSPL) | 14.52721 | 65.70247 |
| **Total** | **<u>43.36518</u>** | **181.38940** |

Table Q.6: Phase 2 Go PostgreSQL Concurrent program vs Rust PostgreSQL Concurrent program.

Table Q.6 show the table of results comparison between Go concurrent program and Rust concurrent program in PostgreSQL database retrieval. The total elapsed time of Go concurrent program is faster than Rust concurrent program in retrieving 4 millions of data from three tables in PostgreSQL database.