Recipe Evolution

*By*

Gulshan Zainab, Rida Rafique & Munaza Nadeem



A thesis submitted to the Department of CS&IT.

University of Sargodha – Pakistan

In partial fulfillment of the requirement for the degree of

BS (Information Technology)

**Supervisor**

Mr. Fahad Maqbool Submission Date: 27-06-2022

Department of CS & IT.

The University of Sargodha.

## Declaration

I Gulshan Zainab (BSITF18M033), Rida Rafique (BSITF18M026), and Munaza Nadeem (BSITF18M027) hereby declare that we have produced the work presented in this thesis, during the scheduled period of study. I also declare that I have not taken any material from any source except referred to wherever due to that amount of plagiarism is within the acceptable range. If a violation of HEC rules on research has occurred in this thesis, I shall be liable to punishable action under the plagiarism rules of the HEC.

Date: 27-06-2022 Signature of the student:

Gulshan Zainab Rida Rafique Munaza Nadeem

BSITF18M033 BSITF18M026 BSITF18M027

## DEDICATION

This thesis is my work of pride which I would dedicate to the only people who were pillars of strength and motivation for me through my whole academic journey. I look up to making my teacher proud and satisfied with my work. Satisfaction is not easy to achieve in our competitive world, so I look forward to making all these precious people’s eyes shine and head held high.

# ACKNOWLEDGEMENTS

A teacher is the beacon of enlightenment and leadership, they can lead you from the ground to the great heights. I am privileged to work with some extraordinary faculty members. First, I would like to bestow my acknowledgment to my Supervisor Mr. Fahad Maqbool for the continuous navigation, support, and uttermost wisdom in my BS study. His guidance leads me to write not only a better thesis but learns a lot during the whole process. Along with my advisor, I would like to express my regards to Mr. Waleed Ahmed for his encouragement, courtesy, and affection. I would like to wholeheartedly thank my thesis fellows for bucking me and showing such politeness when we worked together to finish the work within the given deadline, and for all the enjoyment we have had in the last 4 years.

**Gulshan Zainab BITF18M033**

**Rida Rafique BITF18M026**

**Munaza Nadeem BITF18M027**

**ABSTRACT**

Recipe Evolution

A chef's success directly relates to the taste and nutritional values of recipes that he/she proposes. Formulating new recipes is more of an art than a science. Machine Learning and commercial AI have proved their significance in many fields from the last few decades, including bio-informatics, medical imaging, and data mining. Similarly, machine-generated recipes are not out of the question, and one can develop computer programs that can float novel recipes rich in taste and nutritional values. The major hurdle in such developments is the lack of resources in machine-readable ingredients, instructions, and nutritional values. In this project titled “Recipe Evolution," we will create a software application that will be using a large dataset of recipes from all over the world to intelligently generate new and improved recipes having not only great taste but will also contain required nutritional value.

**Table of Contents**

[Declaration 2](#_Toc17805)

[DEDICATION 3](#_Toc20073)

[ACKNOWLEDGEMENTS 4](#_Toc4323)

[Introduction 1](#_Toc13176)

[1.1 Evolutionary Algorithms 1](#_Toc18973)

[1.1.1 Genetic Algorithms 3](#_Toc5556)

[1.1.1.1 Population Initialization 4](#_Toc9289)

[1.1.1.1.1 Random Initialization 4](#_Toc1195)

[1.1.1.1.2 Source File Initialization 4](#_Toc24210)

[1.1.1.2 Fitness Evaluation 5](#_Toc16745)

[1.1.1.3 Genetic Operators 5](#_Toc25141)

[1.1.1.3.1 Selection 5](#_Toc4874)

[1.1.1.3.2 Crossover 8](#_Toc14670)

[1.1.1.3.3 Mutation 9](#_Toc1257)

[1.1.1.4 Termination 9](#_Toc27855)

[1.1.2 Genetic Programming 9](#_Toc813)

[1.1.2.1 Initial Population 11](#_Toc2903)

[1.1.2.2 Fitness Function 11](#_Toc12474)

[1.1.2.3 Crossover Operator 12](#_Toc26583)

[1.1.2.3.1 Generating one offspring 12](#_Toc27105)

[1.1.2.3.2 Generating two offspring 13](#_Toc22035)

[1.1.2.4 Mutation Operator 14](#_Toc16696)

[1.1.2.4.1 Function Node Mutation 14](#_Toc7307)

[1.1.2.4.2 Terminal Node Mutation 14](#_Toc28798)

[1.1.2.4.3 Swap Mutation 14](#_Toc20048)

[1.1.2.4.4 Grow Mutation 14](#_Toc17799)

[1.1.2.4.5 Gaussian Mutation 14](#_Toc5323)

[1.1.2.4.6 Asexual Mutation Operators 14](#_Toc4981)

[2 Related Work 16](#_Toc28807)

[3 Methodologies 19](#_Toc13004)

[3.1 Data Parsing and Data Processing 19](#_Toc29051)

[3.1.1 Data Extraction 19](#_Toc12073)

[3.1.2 Data Transformation 22](#_Toc10850)

[3.1.3 Data Consolidation 22](#_Toc30621)

[3.2 Knowledge Graph 23](#_Toc41)

[3.3 Adjacency Matrix 25](#_Toc20224)

[4 Results 28](#_Toc15739)

[4.1 Data Extraction Results 28](#_Toc25526)

[4.2 Data Transformation Results 29](#_Toc21487)

[4.3 Data Consolidation Results 31](#_Toc29369)

[4.4 Knowledge Graph Results 34](#_Toc16231)

[4.5 Adjacency Matrix Results 35](#_Toc14317)

[Conclusion 37](#_Toc26011)

[References 38](#_Toc15872)

**List of Figures**

[Figure 1.1 Roulette wheel 6](#_Toc23398)

[Figure 1.2 Rank Selection pie 7](#_Toc18324)

[Figure 1.3 Tree Illustration of GP 11](#_Toc16328)

[Figure 1.4 crossover generating one offspring 12](#_Toc23349)

[Figure 1.5 crossover generating two offspring 13](#_Toc14102)

[Figure 3.1 Knowledge Graph 24](#_Toc10758)

[Figure 3.2 Triples 25](#_Toc11026)

[Figure 3.3 Adjacency Matrix Graph 25](#_Toc21107)

[Figure 3.4 Adjacency Matrix 26](#_Toc18711)

[Figure 4.1 Data Extraction 28](#_Toc25947)

[Figure 4.2 Data Extraction Results 29](#_Toc26274)

[Figure 4.3 Extracted Data 29](#_Toc3767)

[Figure 4.4 Data Transformation 30](#_Toc3858)

[Figure 4.5 Data Transformation File 30](#_Toc12669)

[Figure 4.6 Database Created 31](#_Toc7852)

[Figure 4.7 Database Connection 31](#_Toc31428)

[Figure 4.8 Primary Key 32](#_Toc3810)

[Figure 4.9 Separation of duplicate recipes 32](#_Toc12328)

[Figure 4.10 SQL script 33](#_Toc22485)

[Figure 4.11 SQL queries 33](#_Toc15728)

[Figure 4.12 SQL query for Nutrition 34](#_Toc2840)

[Figure 4.13 Unified Database 34](#_Toc17774)

[Figure 4.14 Knowledge Graph of recipes 35](#_Toc2999)

[Figure 4.15 Knowledge Graph Output 35](#_Toc14887)

[Figure 4.16 Frequency-based Adjacency Matrix 36](#_Toc1912)

**List of Tables**

[Table 1.1 Pseudo-code of Evolutionary Algorithms 2](#_Toc31527)

[Table 1.2 Pseudo-code of Genetic Algorithms 4](#_Toc8608)

[Table 1.3 Fitness Values for roulette wheel 6](#_Toc230)

[Table 1.4 Fitness values for Rank Selection 7](#_Toc11851)

[Table 3.1 Data Extraction 22](#_Toc5286)

[Table 3.2 Creation of Adjacency Matrix 26](#_Toc6331)

[Table 3.3 Adjacency Matrix of ingredients 27](#_Toc8599)

[Table 3.4 Ingredient-Ingredient Frequency relationship 27](#_Toc11564)

**Chapter 1**

# Introduction

Culinary art is not just about providing food but is a whole complex task involving multiple ingredients, best-spotted spices, and delivering a mixture of appetizing flavors by processing food through various baking, frying, or grilling methods. Every culture has a unique recipe, and each individual has a distinct preference for taste and flavor. Recipes evolve through various cultures, presenting a wide range of cuisines, each involving its specific texture, outlook, aroma, and taste. Recipes available on the internet nowadays are so vast, introducing an unlimited number of feasible combinations of ingredients for a single recipe used in the world.

The role of intelligence in culinary art has been under discussion for the long run now. Computational Creativity, one of the emerging branches of Artificial Intelligence (AI), is essential in many creative arts like music production, arts, and graphics.

Our study will use Evolutionary Algorithms to generate recipes, considering the apparent advantage of artificial intelligence and evolution. Recipe generation involves computational methods taking into account speediness, ease, efficiency, and accuracy. We can successfully create recipes by intermixing, evolving, and optimizing existing recipes for the best-fitted recipe according to one’s taste.

## Evolutionary Algorithms

Evolution is a process that helps individuals to adapt themselves to the constantly changing environment. When talking about evolution, it is better to emphasize the area in which evolution is being defined, for example, stellar, cosmic, planetary, organic, or man-made systems of evolution. While talking about Evolutionary Computing, we consider evolution as biological evolution. Charles Darwin’s theory of natural selectionhas become the root of biological evolution. The Darwinian theory of evolution can be summarized as: Individuals in a species vary in their traits, this variation in traits is because of differences in their genes. In a world with limited resources, each individual competes with others for their survival. Individuals with characteristics best suited to their environments are more likely to survive, reproduce, and pass on their genes to their children. Over time, these better-adapted genes become dominant in a population.

Evolutionary Computing refers to computer-based problem-solving systems inspired by the natural selection theory of evolution. Several Evolutionary Algorithms have been developed. An Evolutionary Algorithm, a subset of Evolutionary Computing, is a stochastic search for an optimum solution to a given problem. Due to its efficient nature and robust behavior, it has become an efficient means for the problem-solving method used for the global optimization of problems. The evolutionary search process is influenced by the following main components of an Evolutionary Algorithm:

* an encoding of solutions to the problem as a chromosome;
* a function to evaluate the fitness or survival strength of individuals;
* initializationof the initial population;
* selection operators; and
* reproductionoperators.

**Algorithm 1.1** shows how these components combine to form a generic EA

|  |
| --- |
| **Algorithm 1.1: Generic Evolutionary Algorithm** |
| Let *t* = 0 be the generation counter;  Create and initialize an n-dimensional population, C(0), to consist of ns individuals;  **while** *stopping condition(s) is not true* **do**  Evaluate the fitness, f(xi(t)), of each individual, xi(t);  Perform reproduction to create offspring;  Select the new population, C(t + 1);  Advance to the new generation, i.e. t = t + 1;  **end** |

**Table 1.1** **Pseudo-code of Evolutionary Algorithms**

The different ways in which the EA components are implemented result in different EC paradigms:

**Genetic algorithms (GAs),** model genetic evolution.

**Genetic programming (GP),** is based on genetic algorithms, but individuals are programs (represented as trees).

### Genetic Algorithms

Genetic Algorithms are a part of evolutionary computing, playing an integral part in artificial intelligence. The idea of evolutionary computing came into being in the 1960s by I. Reichenberg. His idea was studied and developed by many researchers, one of those being John Holland, who invented Genetic Algorithms. Genetic Algorithms are randomized search algorithms built to emulate Darwin’s concepts of natural selection and evolution.

All living beings consist of cells. Each cell got a set of chromosomes that contain genes. Gene is a complete code of life that determines what trait the organism possesses. Genes have evolved through reproduction. During reproduction, **crossover** happens to recombine genes from both parents; hence the offspring produced would carry on their traits in a distinct way. If there's no crossover, the resulting offspring will be the same copy of its parents, thus bringing no diversity. Crossover helps to carry on both parents’ traits rather than being an exact duplicate of the previous population. After crossover, newly developed offspring is then mutated by introducing a new gene that differs from that of the parents. Through **mutation**, a part of the chromosome is changed, introducing uniqueness to the next generation. The mutation is usually generated randomly. Just as life evolved through evolution, passing on best-fitted genes to the next generation, in the same way, genetic algorithms are supposed to provide the best fitting solution to the problems after evolving them.

While solving the problems using a genetic algorithm “set of solutions” in **search space** is considered just like a “set of chromosomes” in a population. Search space consists of all feasible solutions to that problem. Each solution present has a fitness value. While looking for the most feasible solution, some extremes (minimum or maximum) are considered. The most suitable solutions bear the offspring that survive in the next generation. Offspring produced bear characteristics of both parents (because of crossover) and some randomly introduced traits as the result of mutation.

Since we are evolving the recipes, to evolve them, the GA will select suitable parent recipes that survive in one generation to pass on their best traits(taste) to the next population of recipes. The pseudo-code of GA is included in Algorithm 1.2:

|  |
| --- |
| **Algorithm 1.2: Genetic Algorithms** |
| **Result:** The best solution   1. Optimization problem 2. START 3. Generate the initial population 4. Compute fitness of the population 5. **while** termination conditions **do** 6. Selection 7. Crossover 8. Mutation 9. Compute Fitness 10. STOP |

**Table 1.2** **Pseudo-code of Genetic Algorithms**

The evolutionary process in GA includes the following phases:

#### Population Initialization

A population that has a set of chromosomes(solutions) is introduced. Population size determines the number of solutions in one generation. If the participants involved in one generation are fewer only a small number of solutions are being implored, but if the population size is too large, GA slows down. It is efficient to introduce an optimum number of solutions in one population.

While initializing our recipe population we introduce recipes from all around the world in preprocessed recipe-instruction trees. This population enters the cycle for it to evolve and discover more novel recipes.

There are two methods to initialize a population:

##### Random Initialization

In Random Initialization, the entire population is populated with random solutions. Random solutions are the ones that guide the population toward an optimum approach because they are more likely to introduce diversity to the population.

##### Source File Initialization

Source File Initialization, also known as Heuristic Initialization, is the one where the initial population is populated using a source code with known solutions. This method leads to a population having similar solutions and very little diversity, as the result, it will affect the initial fitness of the population.

That being the case, the best approach is to start with Heuristic Initialization, where the source file has all good solutions, and then fill the rest of the population with some randomly initialized solutions.

#### Fitness Evaluation

Fitness Evaluation determines how close a given solution is to the optimum solution of the desired problem. In Genetic Algorithms, each solution in the problem space is represented by some binary number. Each solution is tested and granted some fitness score, to indicate how close it is to meet the requirement of being the optimum solution to the problem at hand. For this purpose, the objective function is used to evaluate the fitness of each solution. This fitness criterion determines a certain individual's probability of getting selected for the reproduction phase.

To evaluate a recipe’s fitness, multiple aspects contribute such as ingredient composition, ingredient preferences, recipe validity, and authenticity check by comparing it to the valid recipes.

#### Genetic Operators

Genetic Operators in GA guide the algorithm to the solution of the problem. They use existing solutions and develop new solutions for the population. Various Genetic Operators are used to maintain diversity within the population and effectively explore the available search space for the solutions.

##### Selection

Selection is the process of selecting the parent solutions that will recombine and produce offspring for the next generation. This is a crucial stage as selecting parents with good genes will lead to better and fitter offspring.

There are the following selection mechanisms:

###### Roulette Wheel Selection: In Roulette Wheel Selection, the selection criterion is usually determined by the fitness value of each solution. Hence, the fitter a solution is, the higher the chance of it merging with other solutions and passing on its traits to the next generation. A circular wheel is divided as described below. A fixed point is chosen at the circumference of the wheel and the wheel is rotated. After rotation, the area which comes intact with the fixed point is selected as the parent. For the second parent, this process is repeated.

|  |  |
| --- | --- |
| solutions | Fitness value |
| A | 8.2 |
| B | 3.2 |
| C | 1.4 |
| D | 1.2 |
| E | 4.2 |
| F | 0.3 |

**Table 1.3** **Fitness Values for roulette wheel**

**Figure 1.1** **Roulette wheel**

###### Tournament Selection: In K-way tournament selection, K number of solutions are selected randomly from the population and the best out of these are chosen to be parents. Tournament selection is used as a parent selection mechanism where each parent goes through a compatibility check before performing the recombination operator. The same process is repeated to select the next parent. It is an extremely popular approach as it can even work with solutions that have negative fitness values.

###### **Rank Selection:** Rank Selection can also work with negative fitness values and is usually used when all the solutions in the population have very close fitness values to each other. As shown below, At the end of the cycle, all the solutions occupy nearly an equal share space in the pie which leads to all the individuals having the same probability of getting selected as parents. In this case, each individual is ranked according to their fitness. Higher-ranked solutions are preferred more than the lower-ranked ones.

|  |  |
| --- | --- |
| solutions | Fitness value |
| A | 8.1 |
| B | 8.0 |
| C | 7.95 |
| D | 8.02 |
| E | 7.99 |
| F | 8.05 |

**Table 1.4** **Fitness values for Rank Selection**

**Figure 1.2** **Rank Selection pie**

###### Random Selection: In this Selection mechanism, parents are randomly selected from the population regardless of their fitness. Therefore, this strategy is usually avoided as it does not result in the production of better-fitted children.

##### Crossover

In Crossover Genetic Operator, genes from both parents are combined and passed on to the next generation. A crossover point is chosen, and offspring are produced by combining each parent’s genes at that crossover point. The resulting child is the exact copy of both its parents.

The types of crossover operators are as follows:

###### Single Point Crossover: In Single Point Crossover, one crossover point in parent 1 is selected, and the rest of the information beyond that crossover point is swapped with that of parent 2. Two offsprings are produced by combining the parents at that crossover point. It is the most simple form of crossover.

Parent 1: 10|01

Parent 2: 11|10

Offspring 1: 10|10

Offspring 1: 11|01

###### **Two-Point Crossover:** In this type of crossover, to provide a great combination of the parents two random crossover points are chosen. After selecting the crossover points, information is exchanged at these points. As a result of recombination, two offsprings are produced.

Parent 1: 1001|0110 0011|1111 0001

Parent 2: 1100|1001 1010|0110 0101

Offspring 1: 1001|1001 1010|1111 0001

Offspring 1: 1100|0110 0011|0110 0101

###### Uniform Crossover: In Uniform Crossover, there is no fixed crossover point, we don’t need to divide the gene into segments rather each gene is treated separately. Consider flip-a-coin for each chromosome, each gene (bit) is randomly copied from the corresponding chromosome of one of the parents. The resulting offspring has a mixture of genes from both parents.

Parent 1: 11111111

Parent 2: 00000000

Offspring 1: 10001101

Offspring 1:01110010

###### Three Parent Crossover: In this type of crossover, three parents are selected for the recombination. Each gene from the first parent is compared to that of the second. If genes are similar, the gene is copied for the offspring, or else the equivalent gene is taken from the third parent for the offspring. It is mostly used in the case of binary encoded chromosomes.

##### Mutation

To maintain diversity in genes, a randomly initialized gene is introduced in the offspring so that it is not the same copy as its parents. This change might help to revolutionize the offspring. Diversity is introduced with the help of mutation.

Following are the types of mutation operators:

###### Swap Mutation: In Swap Mutation, two genes are randomly selected in the chromosome and swap each other’s values:

**1 2 3 4 5 6 7 8 9 => 1 6 3 4 5 2 7 8 9**

###### Scramble Mutation: In Scramble Mutation, a part of the chromosome is selected and its values are shuffled/scrambled randomly:

**1 2 3 4 5 6 7 8 9 => 1 2 6 3 7 3 4 8 9**

###### Inversion Mutation: In Inversion Mutation, just like Scramble Mutation, a part of the chromosome is selected, but instead of shuffling it the entire string is inverted:

**1 2 3 4 5 6 7 8 9 => 1 2 7 6 5 4 3 8 9**

#### Termination

After Mutation, a new population of solutions that is better than the previous generation is produced. If the objective of GA to find the best-fitted solution to the problem has been achieved, the evaluation process stops, else we loop back to the selection phase of the GA.

### Genetic Programming

Genetic Programming introduced the concept that computers could learn to solve a problem without being explicitly programmed to do so. Genetic Programming has been considered a specialization of Genetic algorithms. Just like GA, GP mainly focuses on evolving genes, but they are different in a way too. Where GA manages and works on string representation, GP uses tree structures. The size and length of chromosomes in GA are fixed, whereas, in GP, their size varies.

Crammer in 1985, developed the first tree-structured GA for basic symbolic regression. However, it was Koza who was greatly responsible for the popularization of GP in computer science. His GP algorithms have been widely applied to solve problems in many areas, like, symbolic regression, control, games, robotics, and classification.

Genetic Programming algorithms work on a population of individuals, each of them representing a potential solution to the problem. Unlike GA, individuals in GP trees do not have a fixed length or size. The size of the tree refers to the tree depth, and shape refers to the branching of each node. GP has adaptive individuals, as they vary their shape and size according to the application of the reproduction operators. Moreover, domain-specific grammar is defined that reflects how to present a solution to solve the problem. To understand how GP algorithms solve a problem, you need to be aware of the following terminology:

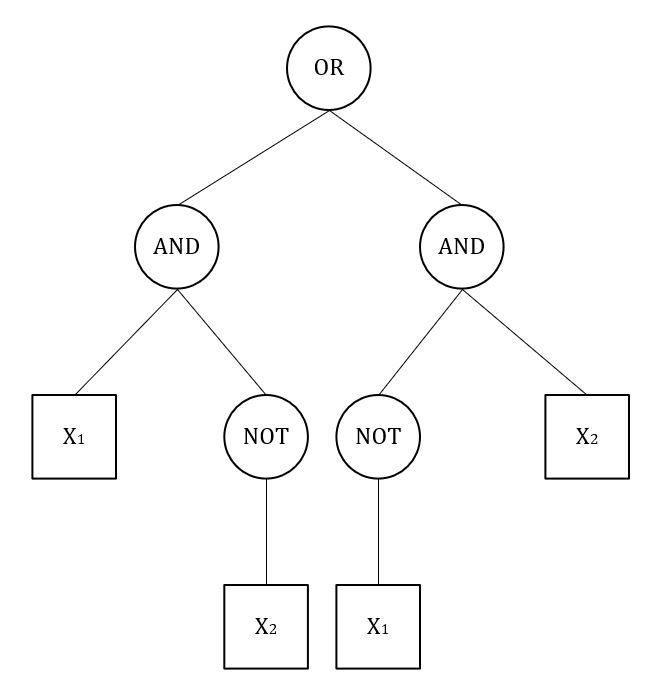
* **The Terminal Set**

The terminal set indicates all input variables and constants.

* **The Function Set**

It is a set of all domain-specific functions that can be applied to the elements of the terminal set to construct potential solutions to a given problem. These functions include arithmetic, mathematical or Boolean functions. Decision functions such as if-else-then and loops can also be included in the function set.

Trees defined in a search space are constructed using the defined grammar. In addition to the terminal and function sets, rules can be applied to ensure the construction of semantically correct trees.



**Figure 1.3** **Tree Illustration of GP**

Let’s take a Boolean expression as an example to illustrate GP representations.

(x1 AND NOT x2) OR (NOT x1 AND x2 )

In the above problem, the terminal set is defined as {x1, x2}, and the function set is {AND, NOT, OR}. The solution is represented in Figure 1.3.

#### Initial Population

The population is initialized randomly within the limitations of maximum depths and semantics as expressed by the given grammar. For each tree, a root is selected randomly from the Function set elements while the branches are determined by the operands of the selected function. Each non-leaf node is randomly selected by the initialization algorithm. The selected element can be either from the function set or terminal set. Just as the element is selected, the corresponding node becomes a leaf and is no longer considered for expansion. Trees in GP are usually initialized as simpler ones. During the evolutionary process, these trees can grow their complexity if necessary.

#### Fitness Function

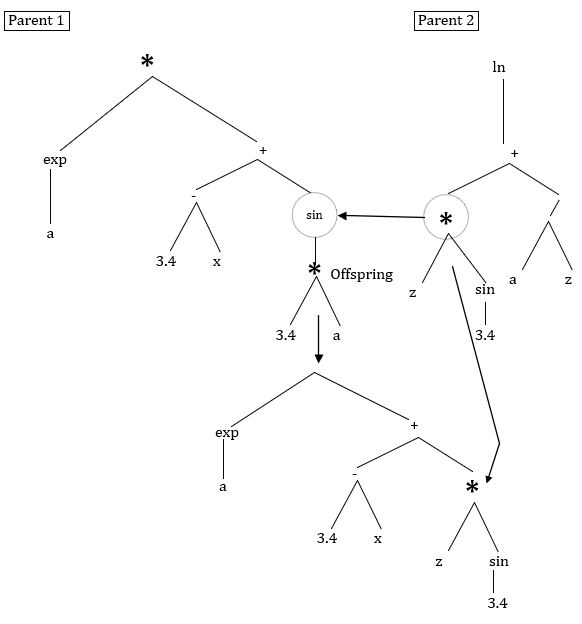
Fitness functions are used to direct simulations toward the best possible design solutions. They show how close a design solution will come to achieving the requirements.

#### Crossover Operator

Any of the two parents that are selected according to their fitness values recombine and produce offspring. There are two types of crossover operators, each producing a different number of offspring.

##### Generating one offspring

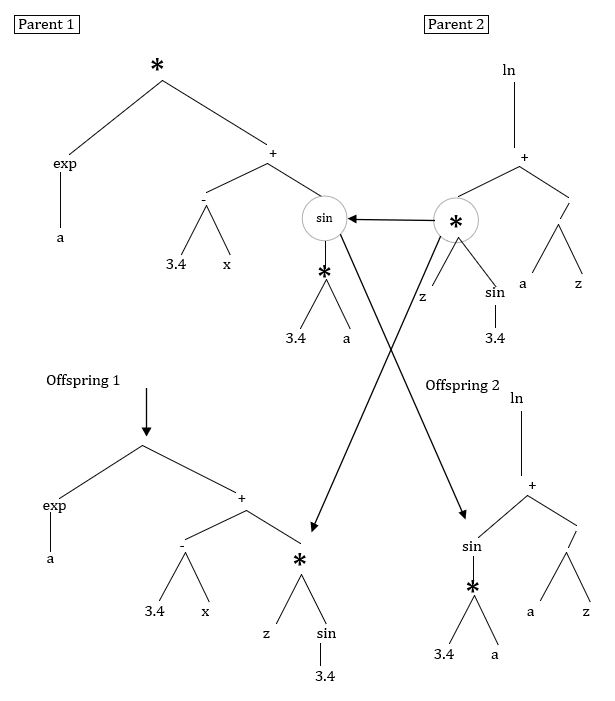
In this, a random node is selected in each parent. The crossover takes place by replacing the corresponding subtree in the first parent with that of the second parent. Figure 1.4 illustrates a crossover operator that produces one offspring.



**Figure 1.4** **crossover generating one offspring**

##### Generating two offspring

Just like in the first case, a random node is selected in both parents. Here, instead of replacing subtree in one parent, the corresponding a subtrees are swapped, resulting in two offsprings. Figure 1.5 shows how crossover in trees can produce two offspring.



**Figure 1.5** **crossover generating two offspring**

#### Mutation Operator

Mutation operators used to solve a GP problem are usually developed to suit the specific application. Nevertheless, many of the mutation operators developed for GP apply to general GP representation. Following general mutation operators can be used:

##### Function Node Mutation

A non-terminal node (function node) is randomly selected and is replaced with the same parity node that is arbitrarily chosen from the function set.

##### Terminal Node Mutation

A leaf node (terminal node) is randomly selected and superseded with a new terminal node. This new terminal node is arbitrarily chosen from the terminal set.

##### Swap Mutation

A non-terminal node is arbitrarily selected and its arguments are swapped as illustrated in Figure 1.4.

##### Grow Mutation

In Grow Mutation, a node is selected randomly and is replaced by a randomly generated subtree. The new subtree has some predetermined depth- restrictions. Figure 1.5 shows that node 3.4 is replaced by a new subtree.

##### Gaussian Mutation

A terminal node containing a constant is randomly selected and mutated by adding a Gaussian random value to that constant.

##### Asexual Mutation Operators

In addition to the mutation operators mentioned above, Koza also introduced some asexual operators. These are as follows:

**Permutation Operator:** This operator is just like the swap mutation. If a function has an n parameter, the permutation operator generates a random permutation from the possible n! permutations of parameters. The arguments of the function are then permutated according to this randomly generated permutation.

**Editing Operator:** This operator is employed to reorganize individuals following their predefined rules.

**Building Block Operator:** This operator aims to identify prospective building blocks automatically. It begins with defining a new function node for a building block that is identified and further is used to replace the subtree which is set forth by the building block. The utmost benefit of this operator is that reproduction operators will not resort to building blocks.

**Chapter 2**

# Related Work

This section presents previous research that specifically looked into the task of substitute generation as well as various methods for evaluating such solutions. Previous research had focused on personalized recipe selection utilizing a variety of characteristics and machine learning-based approaches.

Some approaches had detected ingredient substitution by applying domain knowledge to a Thai cuisine recipe and embedding Semantic Web Rule Language (SWRL) for knowledge inference using a Protege/OWL tool, they were able to obtain ingredient substitution knowledge based on each dish and ingredient characteristic. For ingredient substitution, they used the Semantic Searching approach. The Semantic Web Rule Language (SWRL) had been used to create a collection of rule bases that may be used to find existing ingredients that can be used to replace the rare Thai ingredients. The ingredients substitution model applied Semantic Query-Enhanced Web Rule Language (SQWRL) which is SQL query language for retrieving ontology knowledge. It can replace an ingredient based on sensory properties such as taste, flavor, texture, and appearance. The flaw is that it should include visualization for appearance characteristics, as appearance cannot be anticipated unless it can be seen with the eyes. (T. Angskun, 2014)

Other researchers had focused on the problem when the ingredients needed for the cooking process are listed in a recipe on the website and some people may not be able to obtain all of the items stated in the recipe. To overcome these issues, we need to find a suitable substitute ingredient that works well with the remaining elements in the recipe. The machine learning techniques were applied to find substitute ingredients and predicted which ones will be preferred. The selected ingredients are matched with similar category ingredients using machine learning algorithms, and the top 5 matched ingredients will be recommended to the user. The compatibility and resemblance of substances in the same category had been used to select alternative ingredients. The method is based on a combination of the smoothed correlation weight function and a graph-based approach. The drawback is that the quantity of each ingredient in the recipe is not taken into account. There are no cooking demonstrations or taste assessments. (Yuka Shidochi, 23 October 2009)

Other researchers looked at recipe texts and applied statistical approaches to identify food alternatives. Food-context matrices were utilized to identify substitutions using food log data. They applied the distributional hypothesis, which states that substances found in similar scenarios are more likely to be similar. But converting a large amount of data in the form of matrices is a difficult task to handle. (ACHANANUPARP, 2016) Some researchers had followed the concept of AutoChef which incorporates ideas from machine learning, natural language processing, evolutionary algorithms, and genetic programming to create new recipes. Using natural language processing, AutoChef gathers data from existing recipes, learns the combination of ingredients, preparation operations, and cooking procedures, and develops recipes autonomously. To represent and evolve the recipes, AutoChef uses Genetic Programming. Recipe fitness is a measure of how well a combination of ingredients, actions, and cooking methods can be learned from existing recipe data. Finally, human specialists analyzed the recipes after they had been translated back into text format. AutoChef shows that by combining an evolutionary algorithm and knowledge derived from user-generated recipes publicly available on the internet, it is possible to construct acceptable recipes, as well as clear and understandable instructions and a suitable balance of ingredients. But it is still struggling to come up with recipes that adapt to individual tastes. Even if the result is likely to be a great and unique meal, a user may not be interested in cooking. (Hajira Jabeen, 2020)

At the level of an individual ingredient, some researchers had followed the concept of recipe recommendation using an ingredient network. To capture the relationships between ingredients, two types of networks were built. The complement network, which is made up of two huge communities: one savory, the other sweet, captures which constituents tend to co-occur frequently. These networks stored information about which ingredients go well together and which may be substituted for better outcomes, as well as the ability to forecast which closely related recipes will be more highly rated by users. The substitution network, which was built from user-generated recipe modification suggestions, can be broken into multiple communities of functionally comparable items and represents users' preferences for healthier recipe versions. These experiments showed that characteristics derived from a combination of ingredient networks and nutrition data can accurately predict recipe ratings. For users interacting with recipes, whether the recipe is newly submitted and hence unrated, or they are browsing a cookbook, a whole of the new user-interface capabilities should be included. (ACM Digital Library, 2012)

Other researchers had followed the concept of EvoChef that recombined the instructions, spices, and ingredients from well-rated recipes from many cuisines to create new recipes. Ingredients, their status, spices, and cooking methods are all represented as property graphs in each recipe. The fitness function for the evolving recipes was based on expert opinion and user ratings. Machine learning techniques were used to automatically predict the ratings of the offspring recipes. It was discovered that as the number of generations increased, the general fitness of the recipes improved, and almost all of the resulting recipes were determined to be conceptually right. They also did a blind taste test to compare the original recipes to the EvoChef recipes, and the EvoChef was found to be more innovative. EvoChef is the first semi-automated, open-source, and genuine recipe generator that generates simple and novel recipes. Fact that their initial recipes are highly rated and are taken from popular web pages, they mostly used easy-to-find ingredients. The results produced by EvoChef are complete and precise. The drawback is that it overlooks several important details such as flavor information for component pairing, nutritional information, and the texture of the recipe. (Hajira Jabeen J. L., 2019)

Other researchers focused on the nature and evolution of online food preferences. For this, they used log data gathered from recipe websites to analyze consumers' online food preferences and explore their utility in understanding online food preferences. The following are some of the issues they look into as a result of this work: Ingredient preferences influence recipe preferences in part, recipe preference distributions were more regionally diverse than ingredient preference distributions, and weekday preferences differed significantly from weekend preferences. They concluded that recipe visits may represent a plausible signal for human population food preferences because (i) their observations can be linked to real-world events, such as studies showing that people eat more meat on weekends than other days of the week, and (ii) their observations were fairly consistent on a macro and micro level, implying that the observed online preference distributions can be reproduced at different scales. They expect that by evaluating the observable effect of such preferences on four different dimensions, their research will help others better understand the nature and evolution of online food preferences. The following are the key conclusions of this study: (i) The popularity distributions of recipes and ingredients are heavy-tailed and may be well represented by a severely truncated power-law function (recipes) and a truncated power-law function (ingredients). These effects can be detected at both the micro and macro levels (ii) There are larger regional disparities in recipe preference distributions than in ingredient preference distributions. (iii) Recipe choices are influenced in part by ingredient preferences, and (iv) daily and weekend preferences are diverse. (Claudia Wagner, 2014)

The endless possibilities of recipe combinations make it difficult to figure out which new component complements the others. Some researchers introduced RecipeBowl, a cooking recommendation system that takes a list of ingredients and cooking tags as input and generates a list of probable ingredient and recipe combinations. To train RecipeBowl on a generated dataset, they created a recipe completion challenge in which the model predicts a target ingredient that was previously removed from the original recipe. For prediction, the RecipeBowl had a set encoder and a 2-way decoder. They used the Set Transformer to create meaningful set representations for the set encoder. This model generates a set representation of a leave-one-out recipe and maps it to the ingredient and recipe embedding space. The success of this technique is demonstrated by the results of experiments. While RecipeBowl was able to propose both appropriate ingredients and recipe candidates for a given set of other ingredients, some of the suggested recipe candidates appeared to be incompatible with the suggested ingredients. They should make RecipeBowl better by encouraging it to offer recipe candidates based on some of the ingredients it suggests. (KEONWOO KIM, 2017)

Some researchers had followed food classification constraints by using nutritional information to construct a substitutability heuristic that ranks feasible substitution possibilities using both explicit semantic information about components, encoded in a food knowledge graph, and implicit semantics, recorded through word embeddings. But integrating a significant amount of data into a knowledge graph is a challenging task. (Sola S. Shirai, 2021) Additionally, no approach in our knowledge takes on a vast collection of recipes from all around the world to intelligently develop new and improved recipes that are not only delicious but also provide the essential nutritive value.

**Chapter 3**

# Methodologies

In this section, we will discuss the steps we took to investigate and research a problem, as well as our rationale for the specific processes and techniques we used to identify, collect, and analyze the data that will help you understand the problem. In our study, we try to switch ingredients by using a large dataset of recipes to intelligently generate new and valid recipes. For that purpose, we use different approaches, like:

* Data Parsing and Data Processing
* Data Extraction
* Data Transformation
* Data Consolidation
* Creation of Knowledge Graph
* Creation of Adjacency Matrix

The following gives more details for each step of our approach.

## Data Parsing and Data Processing

The process of converting a string of data from one format to another is known as data parsing. A data parser will assist you in converting raw HTML data into a more understandable format, such as plain text. When it comes to parsing information, not all of it gets converted during the process, and each software has its own set of rules. In a brief, a data parse application converts unstructured data into JSON, CSV, and other file formats while also adding structure to the data.

Parsing is defined as analyzing a string of symbols, special characters, and data structures using Natural Language Processing in the field of computer programming (NLP). Extracting information from data sets and giving it meaning by organizing it according to user-defined patterns is just what parsing is all about. Scholars and computer programmers have varied definitions of parsing, but the general understanding is that it is used to analyze sentences and map semantic relationships between them. In other terms, parsing is the process of extracting data from files and filtering it.

Data processing is the process of transforming raw data into useful information that is also machine-readable. As a result, data processing includes gathering, recording, analyzing, storing, and adapting or modifying raw data to turn it into valuable information.

We will use the following approaches for data parsing and processing:

### Data Extraction

The process or method of extracting data from (typically unstructured or poorly structured) data sources for further processing or migration is known as data extraction. Data transformation is frequently performed after the import into the intermediate extraction system. When data is first loaded into a computer from primary sources, the term data extraction is usually used. Web data extraction is a significant issue that has been tackled using a variety of scientific approaches and in a wide range of applications. Many methods for retrieving data from the Web were designed to address specific challenges in ad-hoc environments. Other approaches, on the other hand, mainly rely on information extraction techniques and algorithms.

For generating new and improved recipes, we have to extract data from multiple websites. Consider the data on various recipe websites being easily accessible in a structured manner that is ready to be analyzed. The majority of these sites do not allow users to save their data to a local or cloud storage location. Some websites offer APIs, however, they usually come with limitations and aren't trustworthy. Although copying and pasting data from a website to your local storage is technically possible, it is difficult and out of the question in real scenarios. For that purpose, we use the software Visual Web Ripper to extract data from targeted websites. It is a powerful web scraping tool that is used to scrape data from the web according to our requirements.

Here are the steps followed while extracting a site for recipes and its data:

**Steps of Data Extraction**

1. Copy and paste the URL of the targeted website’s recipe section into Visual Web Ripper.
2. Load the webpage in Ripper.
3. Select recipe category as page area.
4. Select the recipe category as a link (to open a recipe category).
5. Select recipes in a category as a page area.
6. Select recipes in a category as a link (to open a recipe as a whole).
7. Select and save the attributes as elements in the content tab. We use a recipe schema to follow the naming conventions for different attributes required by us in extraction. The schema along with all the naming conventions of attributes and their descriptions can be found at <https://schema.org/Recipe>
8. Select and save the collection of the attributes (ingredients, instructions, nutrition, and equipment/tool) as a page area in the templates tab.
9. Use page areas to select and store half the collection of the attributes (ingredients, instructions) as a whole unified content.
10. Use page areas to select and store half the collection of the attributes (nutrition, and equipment/tools) as a separate unified collection of content.
11. If Subheadings are found in ingredients and instructions, we use a separate template to extract these values (group and howToSection).
12. Once the whole page and all possible variations are selected for extraction, we save the project as a **.rip** file and run the project and it starts to extract recipe data from the website.

The extracted data is generated and exported in 3 formats XML, Excel Worksheet, and database file for transformation into SQL.

Following are some major recipe attributes and their description that are required to extract:

|  |  |
| --- | --- |
| ***RECIPE PROPERTY*** | **DESCRIPTION** |
| ***url*** | a recipe url that tells the address of the page where the recipe lies |
| ***name*** | a recipe’s name identifies it |
| ***image*** | a url that links us to the image showing what’s its final outlook like |
| ***author*** | a person or organization, who produces the content of the recipe |
| ***cookTime*** | Cook Time is the one that tells the duration it takes to cook a recipe |
| ***prepTime*** | Prep Time would be the time to do all the initial preparations as per directions |
| ***totalTime*** | Total Time is the time it took to perform all the instructions and complete the recipe |
| ***marinateTime*** | Marinate time, is used in the recipes where meat takes time to get marinated |
| ***recipeCuisine*** | Recipe Cuisine is the cuisine of the recipe, for example, Italian pasta |
| ***recipeYield*** | Recipe Yield denotes how many people does a recipe serve |
| ***yield*** | Yield to indicate the quantity it will produce |
| ***cookingMethod*** | The cooking Method tells the procedure we undertake to cook the recipe i.e frying, steaming, baking, grilling, etc. |
| ***commentCount*** | A Comment Count is indicating the number of comments the recipe has received so far |
| ***dateCreated*** | Date Created is the date on which the recipe was created |
| ***datePublished*** | Date Published is the date when the recipe was published and introduced to the public |
| ***dateModified*** | Date Modified is the recent date when the recipe was edited |
| ***keywords*** | Keywords are the tags that are used to search |
| ***aggregateRating*** | Aggregate Rating represents the overall ratings of a recipe, it is calculated in the backend by taking an average of the ratings collected so far |
| ***contentRating*** | Content rating, the official rating the recipe has gotten |
| ***nutrition*** | Nutrition indicates the nutrient content of a recipe consists of like calories, fat content, protein content, fiber content, and vitamins |
| ***estimatedCost*** | Estimated cost shows the cost of raw materials or supplies |
| ***recipeIngredients*** | Recipe Ingredients is the one representing a list of all the ingredients used to make the recipe |
| ***recipeIngredient*** | Recipe Ingredient is the content showing a single ingredient from the ingredient list |
| ***recipeInstructions*** | Recipe Instructions are all the steps that make up the “how-to” part of the recipe, mainly the directions to follow |
| ***step*** | A step in a single direction from the recipe instructions list |

**Table 3.1** **Data Extraction**

By following all of these steps, we successfully extract data from our targeted websites.

### Data Transformation

The process of modifying the format, structure, or values of data is known as data transformation. Data transformation can be either constructive (adding, copying, and replicating data) or destructive (deleting fields and records), aesthetic (standardizing salutations or street names), or structural (adding, copying, and replicating data) (renaming, moving, and combining columns in a database).

In our project, once the data has been successfully extracted, and exported, we convert the data into a SQL database file using SQLite Studio. The steps are as follows:

**Steps of Data Transformation**

1. Select the file named “internal\_websitename”, which is a database file from the output folder of the website’s Visual Web Ripper Project.
2. Open SQLite Studio.
3. Load the internal file into SQLite Studio using the add database option.
4. Once loaded, use the export database option to the same output folder as an SQL file of the database with the extension “.**sql**”.
5. Once the .sql file of the project is successfully exported, we apply 5 changes in that SQL file to make it compatible and according to our data requirements of the finalized database.
6. The five changes are done by using the “Find and Replace” option of Notepad. The five changes are as follows:
   1. Replace **"** with **`**
   2. Replace **[** with **`**
   3. Replace **]** with **`**
   4. Replace **nvarchar(4000)** with text
   5. Replace **guid** with **char(36)**
   6. Also, Comment on PRAGMA BEGIN at start and end.

Once these changes are done, the data is successfully transformed into our desired SQL database, we upload it onto our SQL Server at the address: <http://121.52.153.204/phpmyadmin/index.php>

### Data Consolidation

Data consolidation is the process of combining data from several sources, cleaning it up, and storing it in a single location. It's much easier to gain a 360-degree view of your work when all of your data is in one location. The data is in a standardized manner on the central data source since it is transformed before it is consolidated.

In our project, the recipe data of different websites have been extracted and transformed, and uploaded to the server. We consolidate all the data by using MySQL scripts and MySQL Workbench to move and cleanse all the data from separate databases to one unified database (Group10). The steps involved are:

**Steps for Data Consolidation:**

1. Install and open MySQL Workbench.
2. Setup a database connection using the login credentials of the database server.
3. Select the website database to use.
4. If there are multiple parts of a website’s database, then they are merged before consolidation.
5. Alter the recipe table by adding a column named “idid” after the “url” column.
6. Fill the idid with recipe IDs using its ID from Extraction Sheet and fill the column in sequence using the format of ID+000000 and respective values onwards from 1,2,3….
7. Create two tables named “temp” and “temp2” to separate duplicated and unique recipe records from the recipe table.
8. The table “temp” will store the row\_id and url of the recipes that are duplicated in the database (once or even more than one) and the table “temp2” will store the row\_id and url of the recipes that are unique and found only once in the recipe table of website’s database.
9. After that, we take up all the attributes from the recipe table of the website’s database and consolidate into the recipe table of the finalized database (Group10.Recipes) using its “idid” to uniquely identify each recipe and website it belongs to.
10. After the recipe table is successfully consolidated, we move on to consolidating the other three tables (ingredients, instructions, nutrition).
11. Using separate MySQL queries for subheadings and without subheadings, we consolidate all the values (idid, source index, groupName, and recipeIngredient) of the ingredients table of the website’s database into the ingredients table of the finalized database (Group1.Ingredients) using its “idid” as unique identifier.
12. Just like ingredients, using separate MySQL queries for subheadings and without subheadings, we consolidate all the values (idid, source index, howToStep, and recipeInstruction/step) of the instructions table of the website’s database into the instructions table of the finalized database (Group10.Instructions) using its “idid” as a unique identifier.
13. Nutrition data is consolidated as same as the recipe table if they are extracted inside the recipe table and it is done because of different values of different nutrition present in it. We use a separate query if the nutrition is extracted as a separate table.

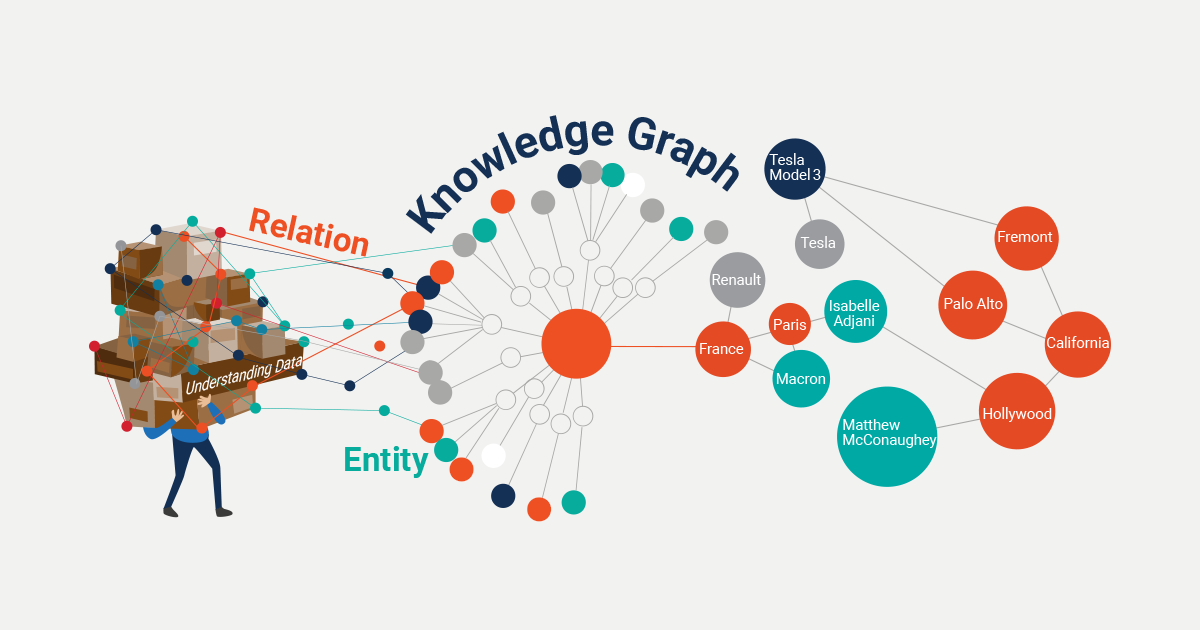
We consolidate all the values (idid, nutrition name) of the nutrition table of the website’s database into the instructions table of the finalized database (Group10.Nutrition) using its “idid” as a unique identifier.

Once all the tables are filled, a website’s consolidation is completed.

## Knowledge Graph

A knowledge graph, also known as a semantic network, depicts the interaction between a network of real-world elements, such as objects, events, situations, or concepts. The term "knowledge graph" comes from the fact that this information is frequently kept in a graph database and represented as a graph structure.

Knowledge graphs have been associated with linked open data projects since the emergence of the Semantic Web, concentrating on the links between concepts and entities. They're also linked to and used by search engines like Google, Bing, and Yahoo, as well as knowledge-engines and question-answering services like WolframAlpha, Apple's Siri, and Amazon Alexa, and social media sites like LinkedIn and Facebook.



**Figure 3.1** **Knowledge Graph**

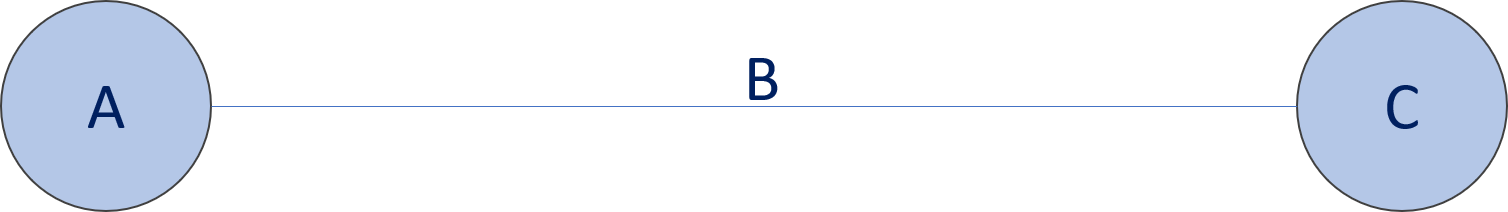
Knowledge graphs contain features from a set of data management paradigms:

* **Database**, because structured queries can be used to explore the data;
* **Graph**, They can be studied in the same way as any other network data structure;
* **Knowledge base**, because they have formal semantics that can be utilized to understand data and derive new truths.

Because knowledge graphs, which are represented in RDF, include the following features, they give the ideal framework for data integration, unification, linking, and reuse:

* **Expressivity:** RDF(S) and OWL, two Semantic Web standards, provide for the fluent representation of numerous forms of data and content, including data structure, taxonomies, vocabularies, and various types of metadata, reference, and master data. Provenance and other structured metadata can be easily modeled with the RDF\* extension.
* **Performance:** All of the criteria have been carefully considered and tested to enable the effective handling of graphs containing billions of facts and characteristics.
* **Interoperability:** Data serialization, access (SPARQL Protocol for end-points), management (SPARQL Graph Store), and federation are all covered by several protocols. Data integration and publication are made easier with the usage of globally unique IDs.
* **Standardization:** All of the above is standardized through the W3C community process to ensure that the needs of many actors – from logicians to enterprise data management specialists and system operations teams – are met.

Nodes, edges, and labels are the three fundamental components of a knowledge graph. A node can be anything, location, or person. The relationship between the nodes is defined by a straight line called an edge as shown in Figure 3.1. They express knowledge as subject-predicate-object triples, with the predicate indicating the subject-object relationship.



**Figure 3.2** **Triples**

In Figure 3.2, A represents the subject, B for the predicate, and C for the object.

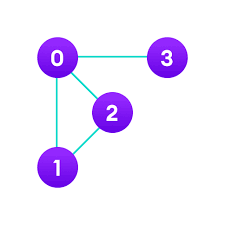
For forecasting the efficacy of recipes, many knowledge-graph approaches have already been developed. The majority of these methods do not use predicates, instead relying on the similarity of recipes to predict efficacy. The number of common elements in a graph between two recipes, or between an ingredient and a recipe, is counted using these approaches. The underlying idea behind these methods is that a large number of common entities suggest that recipes are similar, and hence are likely to be effective for the same recipe. They usually work well with pre-existing, well-defined recipes.

By using this idea in which nodes are connected to based on common objects, we tried to connect common ingredients in different recipes and by doing so all the existing recipes which contain common ingredients are connected.

## Adjacency Matrix

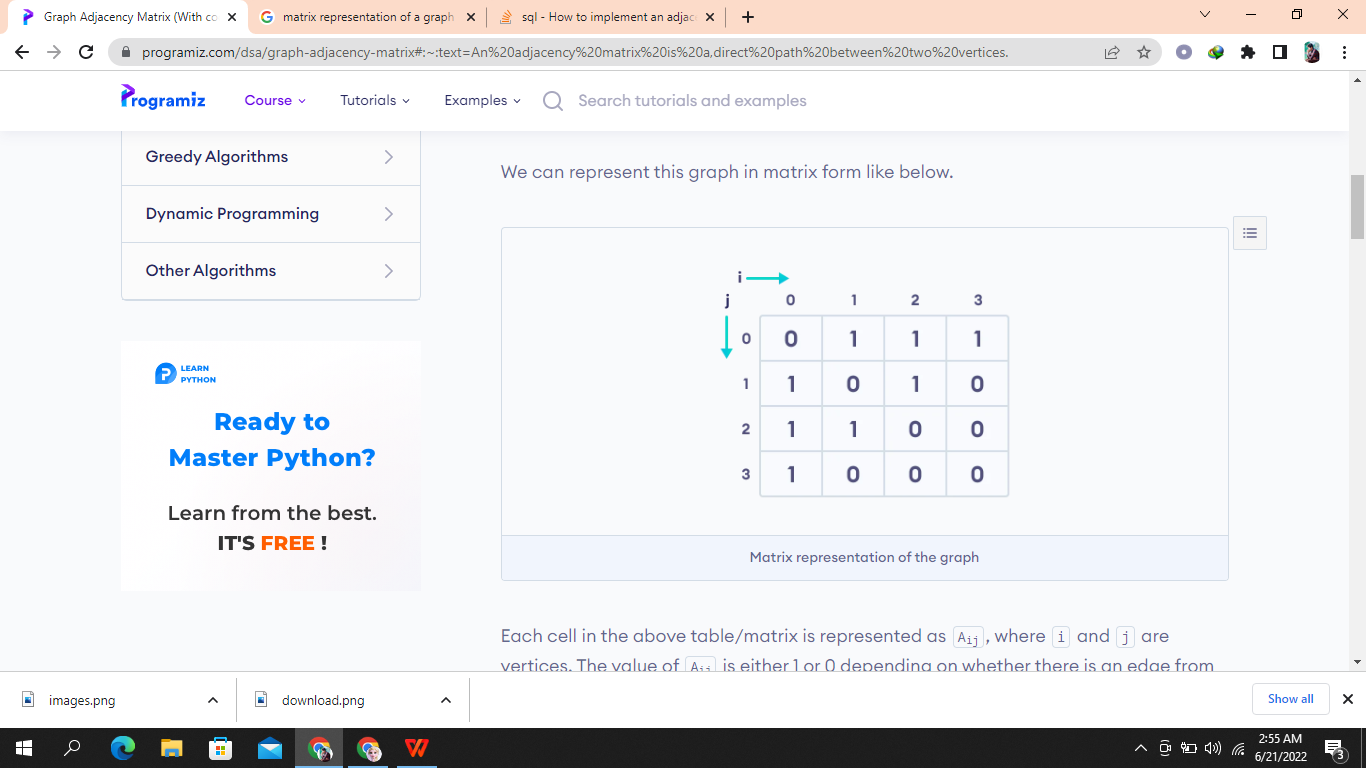
A graph can be represented as a matrix of boolean (0s and 1s) using an adjacency matrix. On a computer, a finite graph can be represented as a square matrix, with the boolean value of the matrix indicating whether there is a direct path between two vertices.

For example, have a look at the graph below in Figure 3.3.



**Figure 3.3** **Adjacency Matrix Graph**

This graph can be represented as a matrix as seen below in Figure 3.4.



**Figure 3.4** **Adjacency Matrix**

Aij represents each cell in the preceding Figure 3.4, where I and j are vertices. Depending on whether there is an edge from vertex I to vertex j, the value of Aij is either 1 or 0.

If there is a path from I to j, Aij equals 1, otherwise, it equals 0. There is a path from vertex 1 to vertex 2, for example, so A12 is 1, and there is no way from vertex 1 to 3, so A13 is 0.

Because every edge (i,j) has an edge (i,j), the matrix is symmetric around the diagonal in undirected networks (j, i).

Basic operations such as adding an edge, removing an edge, and verifying whether there is an edge from vertex I to vertex j take very little time and are performed in constant time. An adjacency matrix should be used first if the graph is rich and the number of edges is large. The use of matrices, on the other hand, provides the greatest benefit. We can gain crucial insights into the nature of the network and the relationship between its vertices by executing operations on the neighboring matrix.

We apply this idea to recipe data. The creation of a knowledge graph gives us the path from which we can recognize which ingredient is commonly used in the recipes. By doing so we recognize different groups of commonly used ingredients and we give them different frequencies. By using these frequencies, we create an adjacency matrix of common ingredients in each recipe.

We have looked for the most efficient approach to do this task in MySQL. We work on phpMyAdmin. In our table, for example, we have two columns: "ID" and "Ingredients." Assume the table is Table 3.2:

|  |  |
| --- | --- |
| ID | Ingredients |
| 301 | Garlic |
| 302 | Ginger |
| 303 | Tomato |
| 304 | Garlic |
| 305 | Potato |
| 306 | Ginger |

**Table 3.2** **Creation of Adjacency Matrix**

We create an adjacency matrix with ingredients as column and row headers. For example, ginger and garlic have a relationship in id 301 and id 303, whereas garlic and potato have a relationship in id 303, therefore we make an adjacency matrix based on that as shown in Table 3.3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Garlic | | Ginger | Tomato | Potato |
| Garlic | 0 | 2 | 0 | 1 |
| Ginger | 2 | 0 | 0 | 1 |
| Tomato | 0 | 0 | 0 | 0 |
| Potato | 1 | 1 | 0 | 0 |

**Table 3.3** **Adjacency Matrix of ingredients**

We also keep track of this in a database as an ingredient-ingredient relationship as shown in Table 3.4.

|  |  |  |
| --- | --- | --- |
| Ingredient | Ingredient | Frequency |
| Garlic | Garlic | 0 |
| Garlic | Ginger | 2 |
| Garlic | Tomato | 0 |
| Garlic | Potato | 1 |
| Ginger | Garlic | 2 |

**Table 3.4** **Ingredient-Ingredient Frequency relationship**

In the following, we will outline the key findings of our work and their implications. In future work we plan to identify common neighbors of ingredients in the existing recipes and then based on this, we may produce valid and innovative recipes having not only wonderful taste but a similar texture.

**Chapter 4**

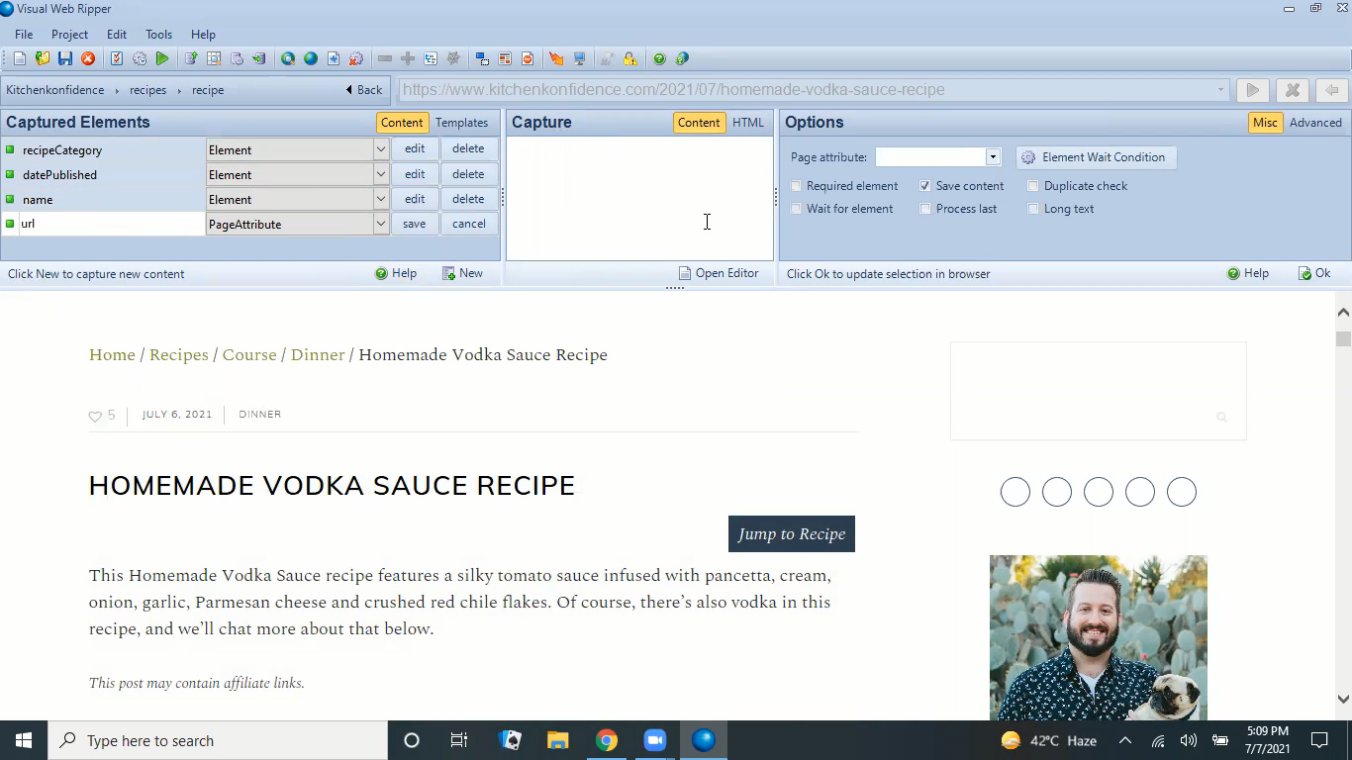
# Results

This section demonstrates that valid and novel recipes can be produced using the methodologies outlined earlier and the information collected from publically available user-generated recipes on the internet. Finding a nice combination of materials and creating clear, straightforward instructions, in particular, seem to perform effectively. Here, we present the outcomes of our approaches.

## Data Extraction Results

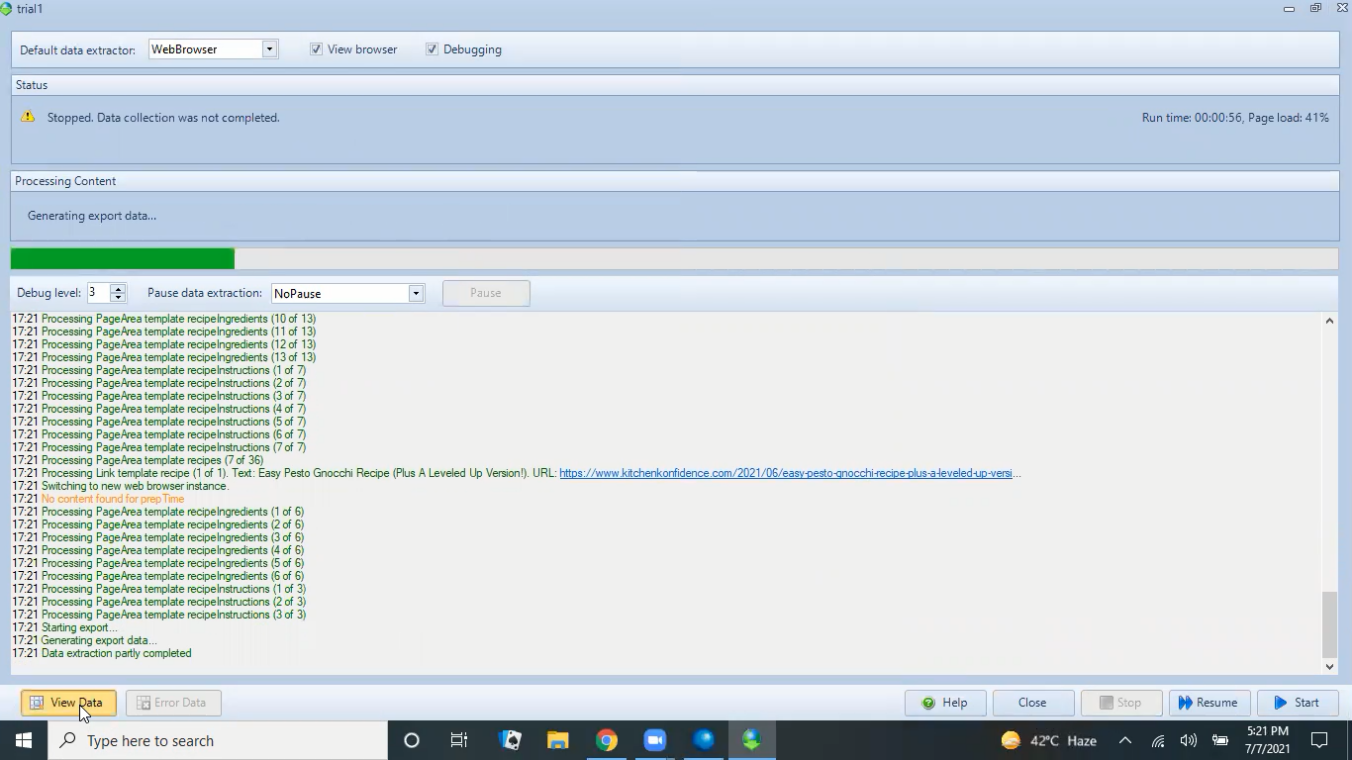
We use the software Visual Web Ripper to extract data from targeted websites.

After pasting the URL of the website from which we want to extract in visual web ripper, we took instructions, nutrition, prep time, cook time, and all other things which are present in the Content Tab in the following Figure 4.1.

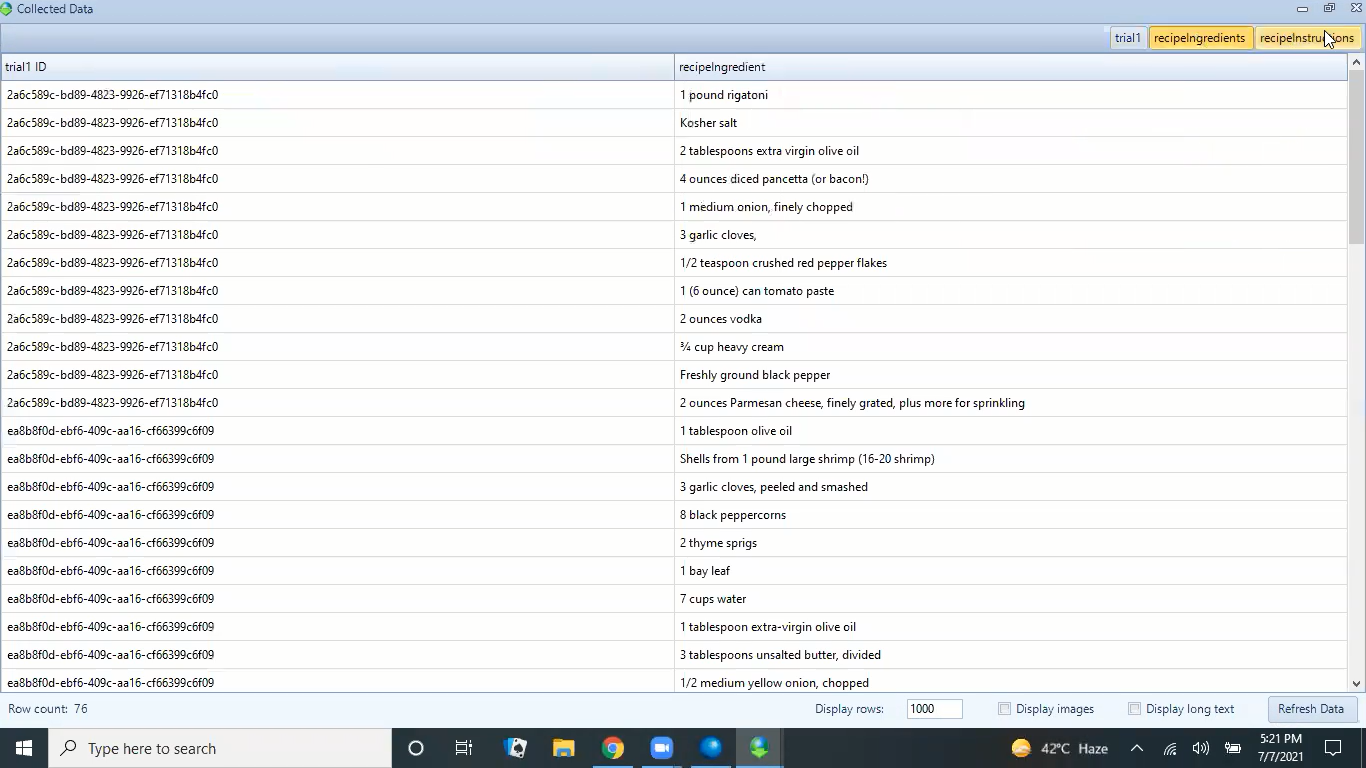


**Figure 4.1** **Data Extraction**

After choosing all the data which is present on the website, we run the project and extract all the data successfully as shown in Figures 4.2 and 4.3.



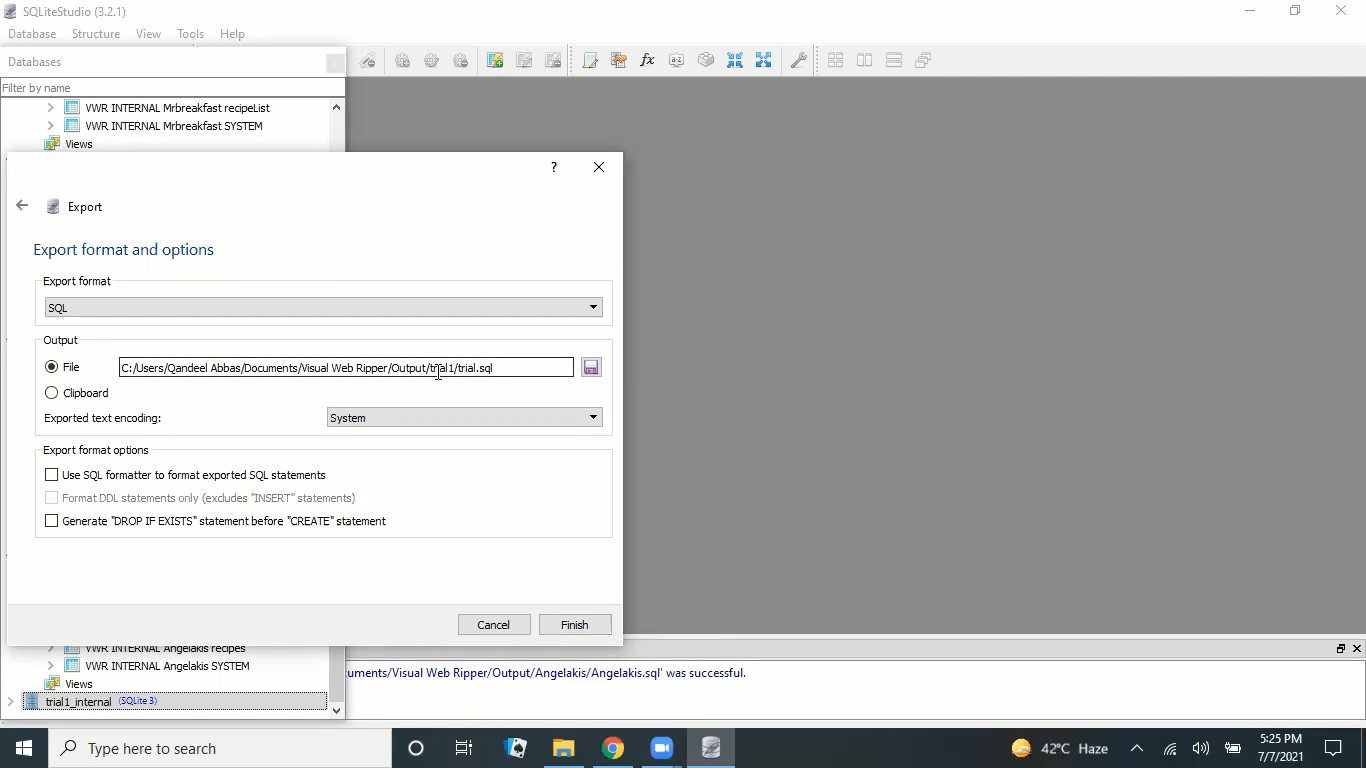
**Figure 4.2** **Data Extraction Results**



**Figure 4.3** **Extracted Data**

## Data Transformation Results

We use the software SQLite for data transformation. First, we convert the .rip file to a .sql file as shown in Figure 4.4.



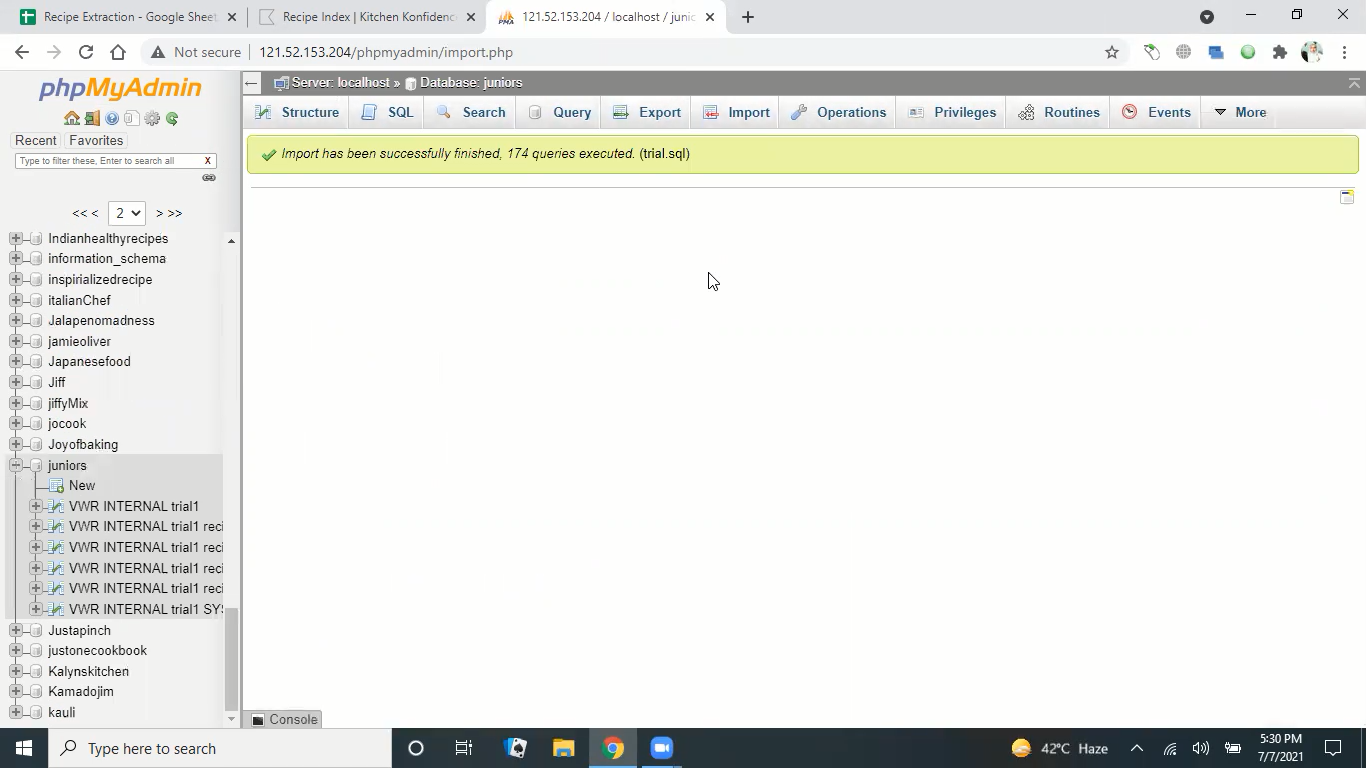
**Figure 4.4** **Data Transformation**

After converting the file, we perform some changes as discussed earlier to convert the file according to our requirements as shown in figure 4.5.



**Figure 4.5** **Data Transformation File**

When all changes were done successfully, we create a database with the name of our targeted website using phpMyAdmin and upload our file data into the newly created database as shown in Figure 4.6.

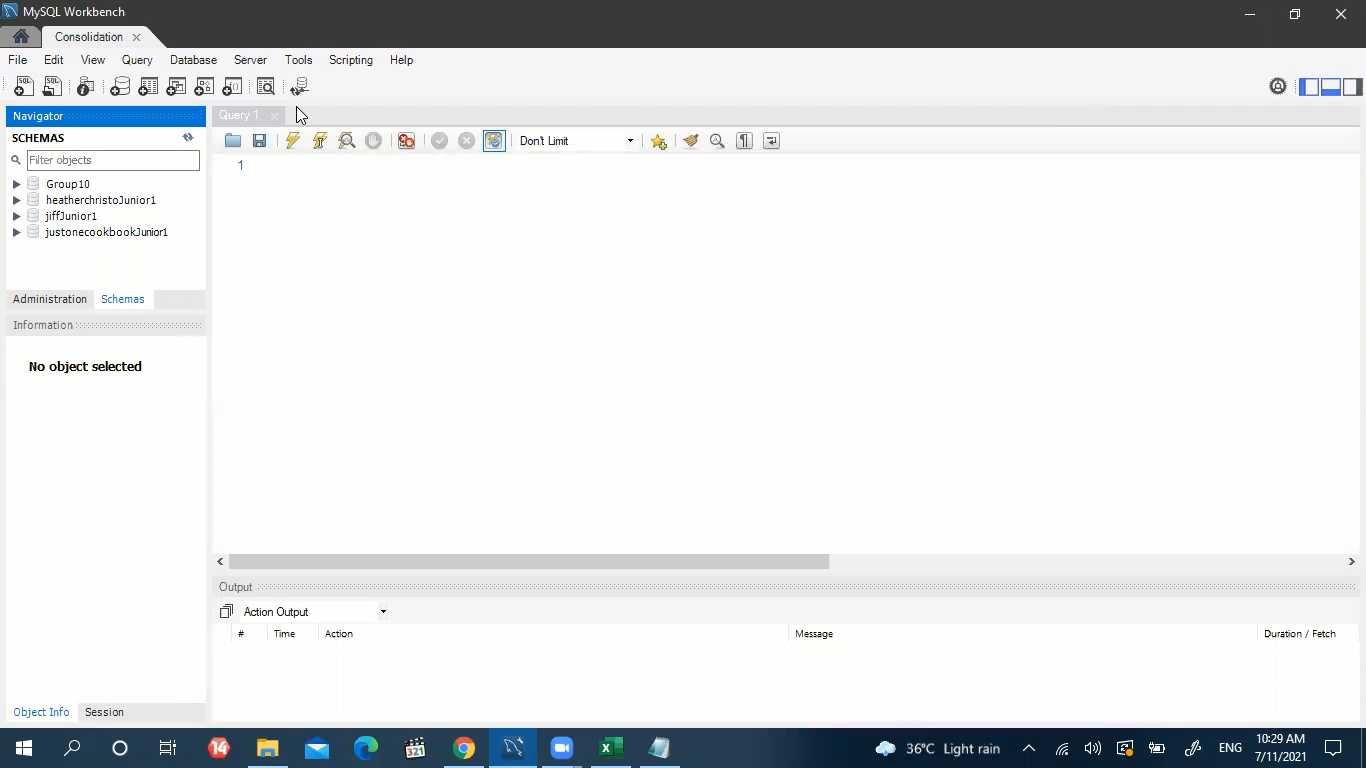


**Figure 4.6** **Database Created**

## Data Consolidation Results

We use MySQL scripts and MySQL Workbench for data consolidation.

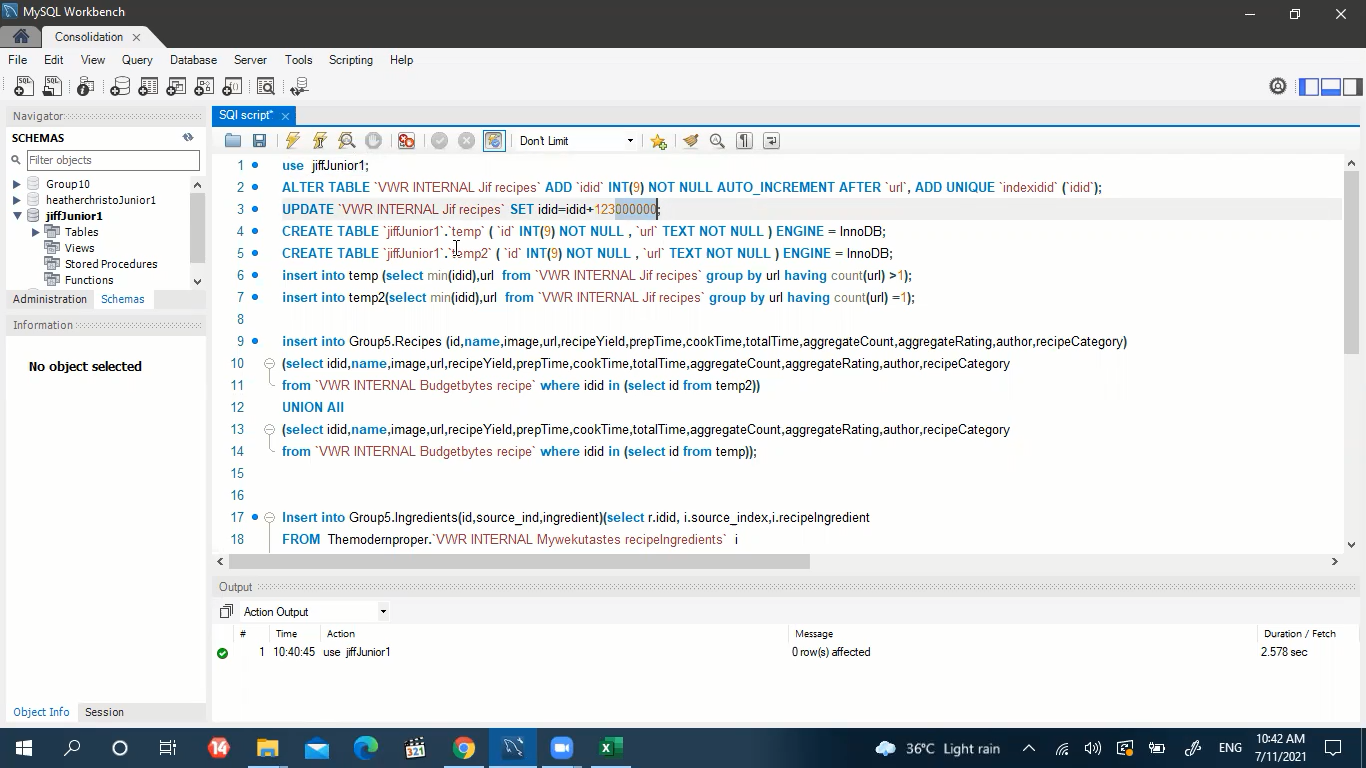
First, we make a connection from the database in the SQL workbench using our login credentials as shown in Figure 4.7.



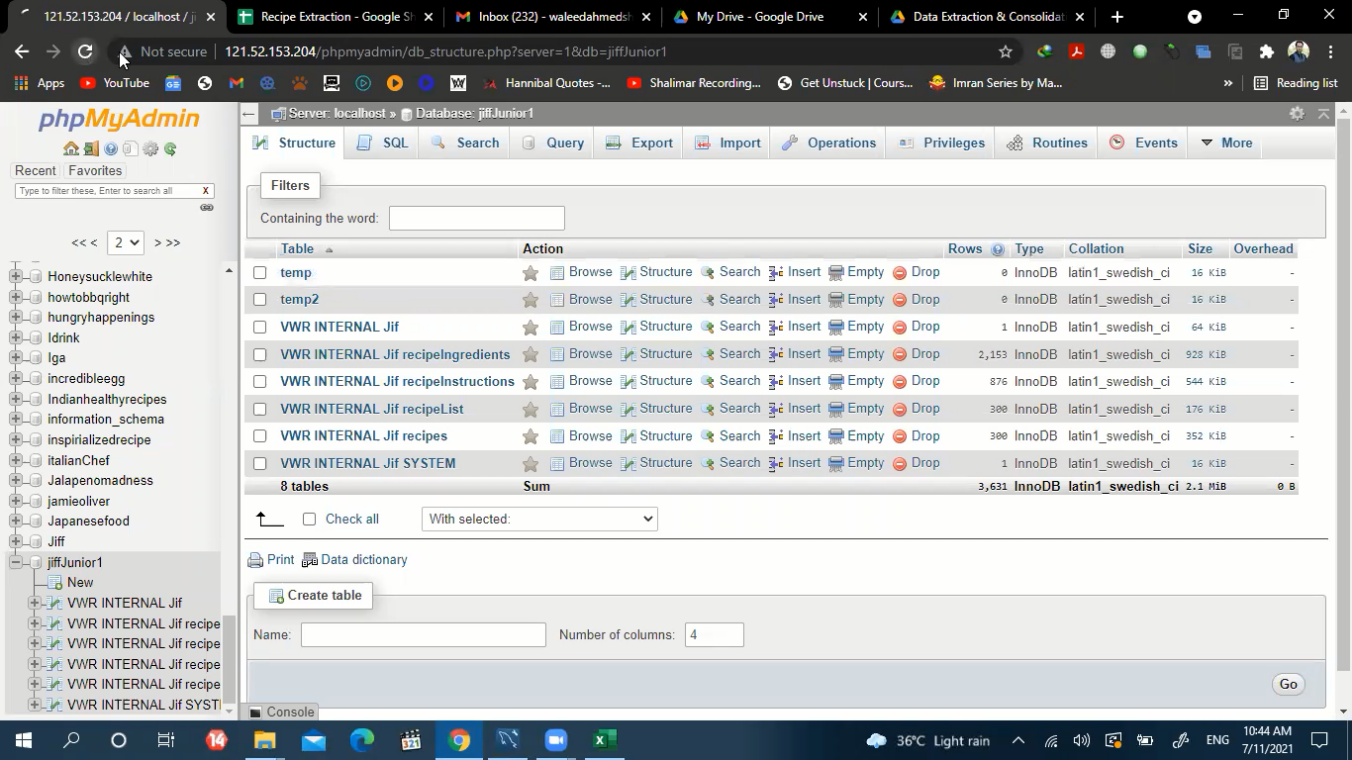
**Figure 4.7** **Database Connection**

After connecting successfully with our database, we open the SQL Script file and run SQL Query to gather all the recipe data from separate databases into one unified database.

We took “idid” as the primary key as highlighted in Figure 4.8 and by using the “idid” key we extract all the duplicate recipes in the “temp” table and extract all the unique recipes in the “temp2” table as shown in Figure 4.9.

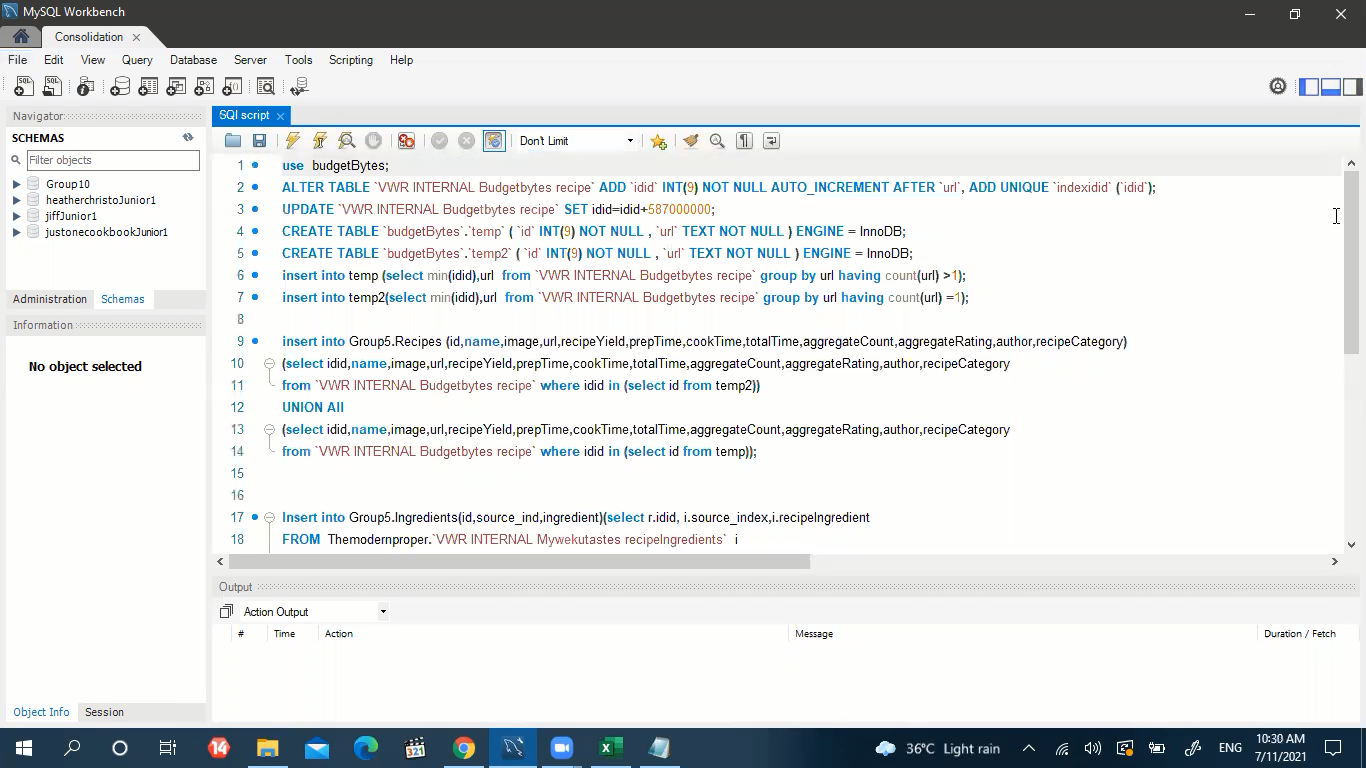


**Figure 4.8** **Primary Key**



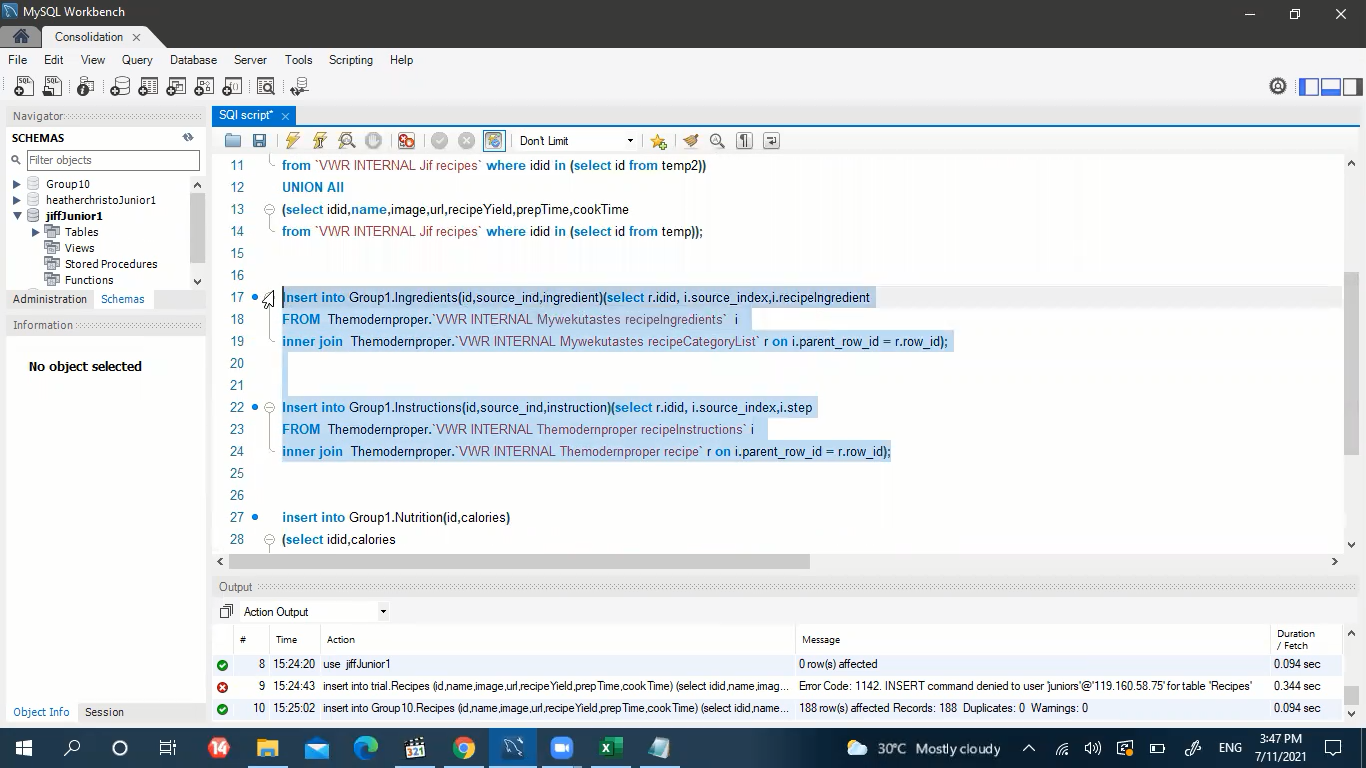
**Figure 4.9** **Separation of duplicate recipes**

After extracting unique recipes, we upload all these recipes in the database named “Group10” using SQL script as shown in Figure 4.10.



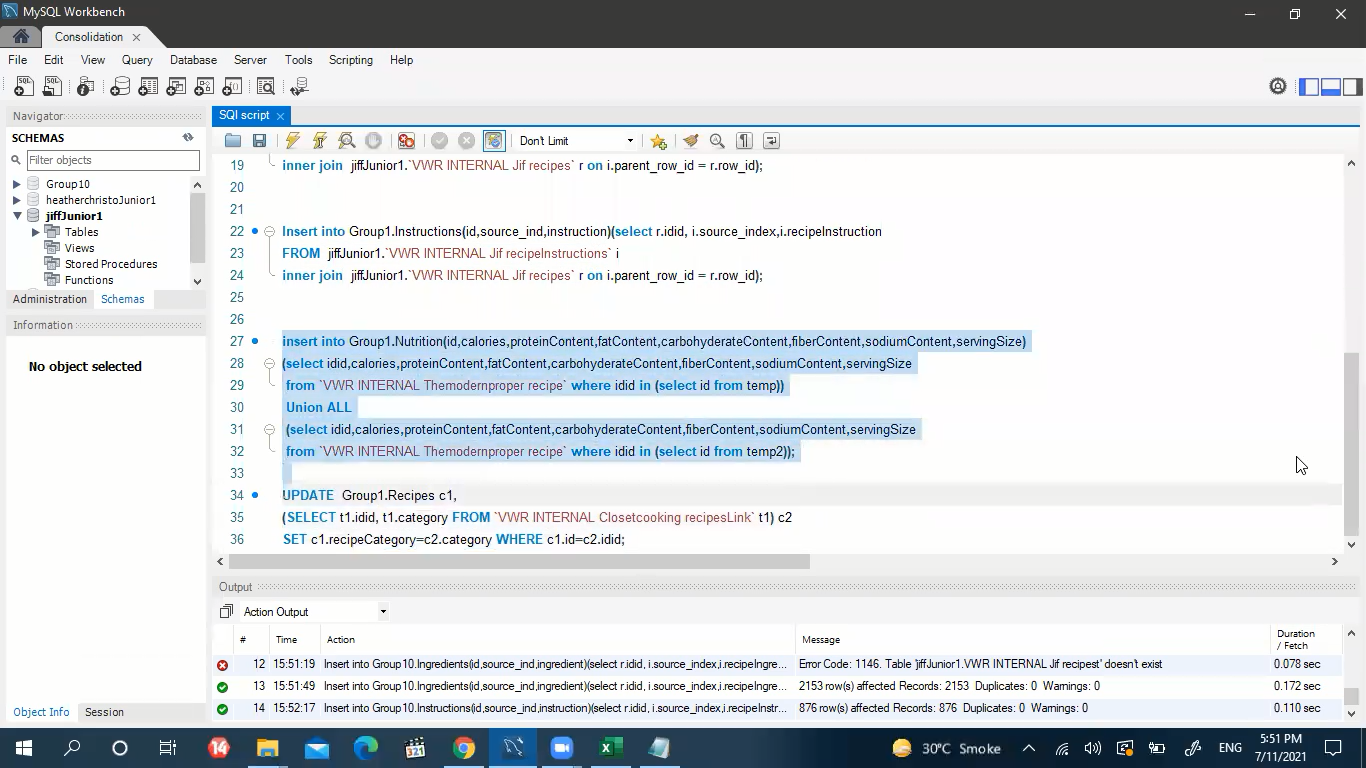
**Figure 4.10** **SQL script**

We upload all the instructions and ingredients in “Group10” by using SQL queries as shown in Figure 4.11.



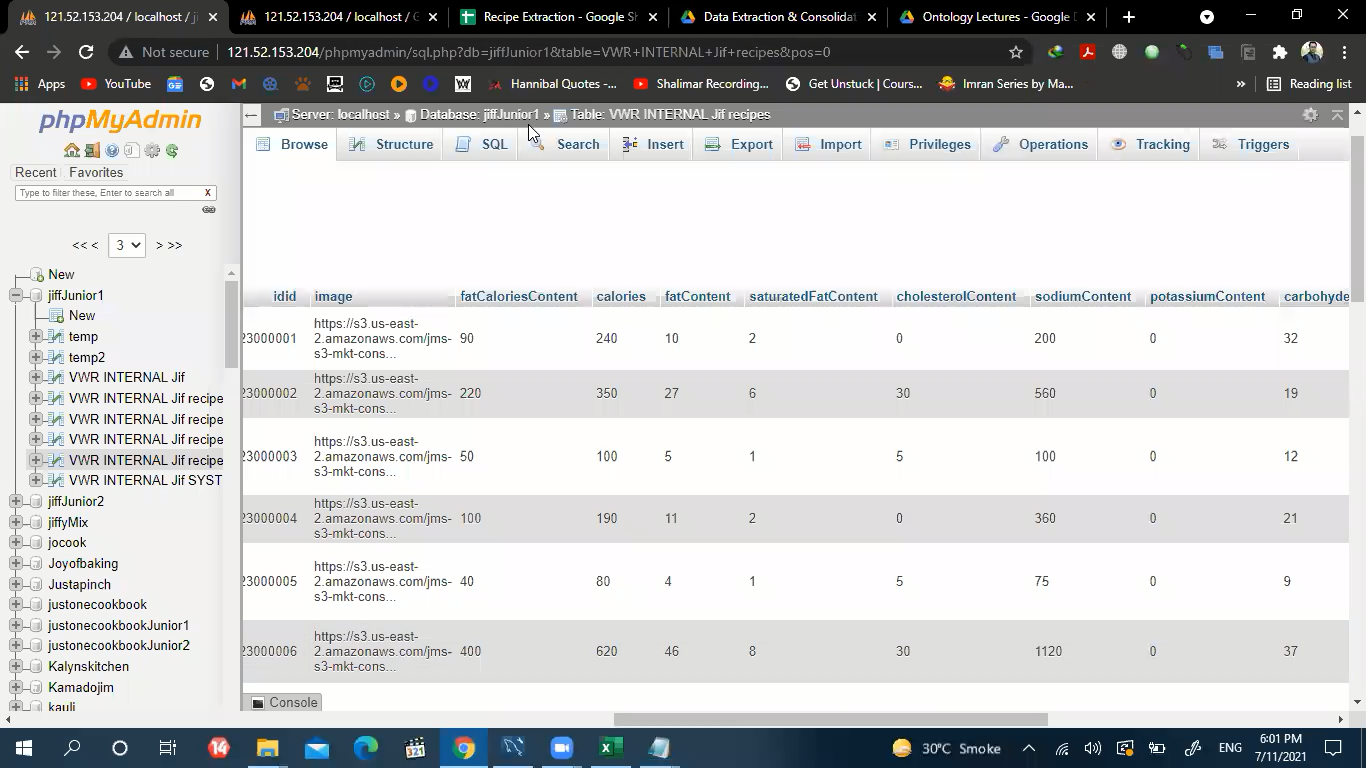
**Figure 4.11** **SQL queries**

As well as we upload all the Nutrition in “Group10” by using SQL queries as shown in Figure 4.12.



**Figure 4.12** **SQL query for Nutrition**

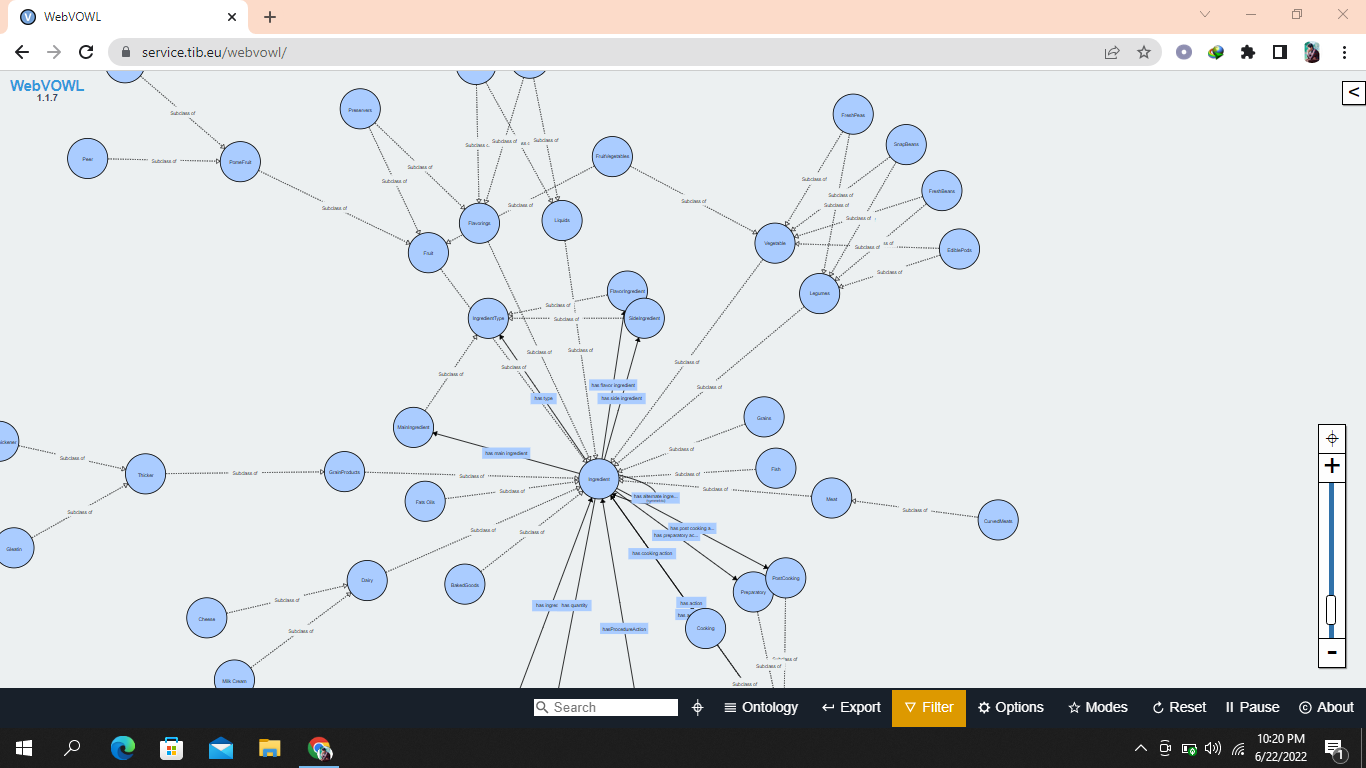
By following all the steps that are discussed in the previous chapter, we successfully exported all the recipe data into one unified database as shown in Figure 4.13.



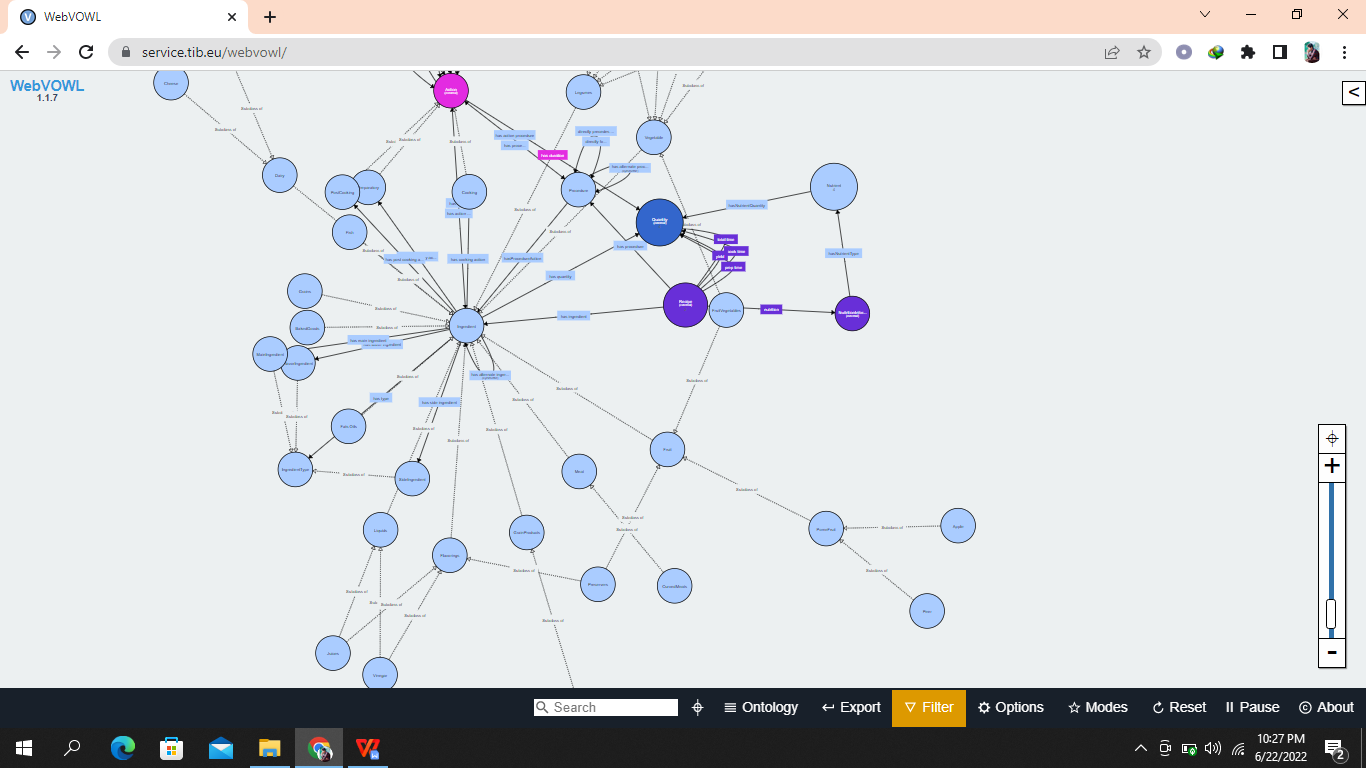
**Figure 4.13** **Unified Database**

## Knowledge Graph Results

We create a knowledge graph for connecting different types of recipes. The knowledge graph in the following Figures 4.14 and 4.15 is based on common ingredients and actions in the existing recipes which are connected showing that this specific ingredient can also be used in the other specific recipe. This provides the base for using common ingredients in different recipes.



**Figure 4.14** **Knowledge Graph of recipes**

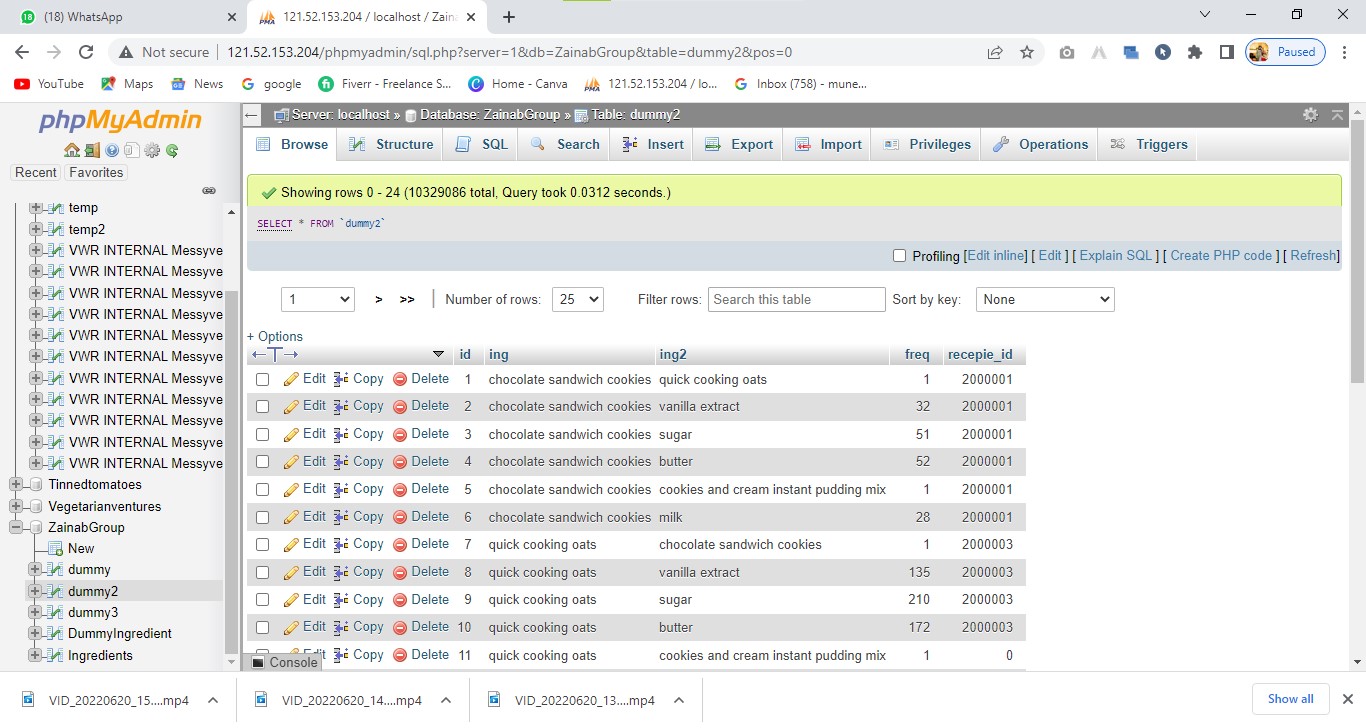


**Figure 4.15** **Knowledge Graph Output**

## Adjacency Matrix Results

By using the knowledge graph as previously discussed in this chapter, which contains all ingredients. We gave some value to each ingredient by applying some SQL queries. Those ingredients which have approximately close value become a common neighbor with the same frequency and can be substituted with each other.

For that purpose, we also use different SQL queries to separate the ingredients with the same frequencies. In the following Figure 4.16, table “dummy2” in the database “ZainabGroup” shows the common ingredients “ing” and “ing2” with their frequencies in the “freq” row. This frequency of the common ingredients creates an adjacency matrix of existing recipes which can be used for generating novel recipes.



**Figure 4.16** **Frequency-based Adjacency Matrix**

The main results of this research are a)The food item’s quality and amount stay consistent. b)The portion size is standard, and the price is similarly consistent. c)The nutritional composition is guaranteed to satisfy dietary concerns for specific groups of people. d)The food items relate to the premise of “truth on the menu”.

Conclusion

A significant strategy to improve and maintain one's health is through controlling and manipulating nutrient intake from diet. One option to help people improve their diets is to substitute ingredients, however, it can be challenging for people to find acceptable substitutes for ingredients in a recipe and to evaluate which substitutions are "healthier" for their specific dietary requirements. By combining explicit and implicit semantic data on ingredients from different sources, we develop a strategy for ingredient substitutability. Utilizing nutritional data and food classification on common ingredients, our strategy may be used to automatically replace valid ingredient substitution. To only rank substitutions that are "healthier" for a specific dietary goal, substitution possibilities might be filtered based on common ingredient content and classification. Our findings indicate that automatically recommending alternatives that meet a person's nutritional needs could enable people to better control their diets. By examining the common ingredients in existing recipes, we expect that this research will help to understand the emergence and evolution of ingredient substitution. In conclusion, our preliminary research has revealed interesting outcomes in this underrepresented field, and there are a lot of different ways to continue this work.

# References

(2012, June). Retrieved from ACM Digital Library: https://scholar.google.com/scholar?hl=en&as\_sdt=0%2C5&q=ingredients+substitution+in+recipe+evolution&oq=#d=gs\_qabs&u=%23p%3DI\_GmkWnVnIQJ

ACHANANUPARP, P. (2016). Retrieved from CORE: https://core.ac.uk/download/pdf/111753794.pdf

Claudia Wagner, P. S. (2014). Retrieved from SpringerLink: https://link.springer.com/content/pdf/10.1140/epjds/s13688-014-0036-7.pdf

Hajira Jabeen, J. L. (2019, March). Retrieved from ResearchGate: https://www.researchgate.net/publication/332328076\_EvoChef\_Show\_Me\_What\_to\_Cook\_Artificial\_Evolution\_of\_Culinary\_Arts

Hajira Jabeen, J. W. (2020, July). Retrieved from ResearchGate: https://www.researchgate.net/publication/346700556\_AutoChef\_Automated\_Generation\_of\_Cooking\_Recipes

KEONWOO KIM, D. P. (2017). Retrieved from IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/9570315

Sola S. Shirai, O. S. (2021, January). Retrieved from https://www.meta.org/papers/identifying-ingredient-substitutions-using-a/33733228

T. Angskun, J. A. (2014). Retrieved from Semantic Scholar: https://www.semanticscholar.org/paper/ONTOLOGY-BASED-KNOWLEDGE-ACQUISITION-FOR-THAI-Angskun-Angskun/1677278b35537ea7192bf201e076739ae2a7b494

Yuka Shidochi, T. T. (23 October 2009). Retrieved from Semantic Scholar: https://www.semanticscholar.org/paper/Finding-replaceable-materials-in-cooking-recipe-Shidochi-Takahashi/44d6f8f378c07aeae21b446c07358b1226cbf0ec