**《数据库实训》 课程报告（分组）**

**(2019-2019 学年第2 学期)**

**3NF范式分解小工具**

**提交日期： 2023 年 07 月 09日**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学 院** | **软件** | | **专业班级** | **软件工程** | |
| **课程名称** | **数据库实训** | | **任课教师** | **曾兵** | |
| **人数** | **4** | |  |  | |
| **学 号** | | **学生姓名** | | | **分 数** |
| **202130190273** | | **邓靖丰** | | |  |
| **202164700240** | | **古旭鹏** | | |  |
| **202130134765** | | **晏璐涛** | | |  |
|  | |  | | |  |
|  | |  | | |  |
|  | |  | | |  |
|  | |  | | |  |
| **教师评语：**  **教师签名：** | | | | | |

**Training report**

[1. Main body 3](#_Toc139935463)

[1.1 Demand Analysis 3](#_Toc139935464)

[1.2 General Design 4](#_Toc139935465)

[1.3 Detailed Design 4](#_Toc139935466)

[1.3.1 Normalize Input (copy 你的函数以依赖添加的说明) 5](#_Toc139935467)

[1.3.2 Check FD Triviality 5](#_Toc139935468)

[1.3.3 Test & Decompose of BCNF 5](#_Toc139935469)

[1.3.4 Test & Decompose of 3NF 5](#_Toc139935470)

[1.3.5 UI Design 7](#_Toc139935471)

[2. System Tutorial 9](#_Toc139935472)

[2.1 Application Installation 9](#_Toc139935473)

[2.2 Click to Run & Input Your Schemas 9](#_Toc139935474)

[2.3 Click Submit Button to Get Result 10](#_Toc139935475)

[3 Group Discussion 11](#_Toc139935476)

[4 Personal summary 12](#_Toc139935477)

[4.1 邓靖丰 12](#_Toc139935478)

[4.2 张治展 12](#_Toc139935479)

[4.3 晏璐涛 12](#_Toc139935480)

[4.4 古旭鹏 13](#_Toc139935481)

[5. Work Allocation & Schedule 13](#_Toc139935482)

# 1. Main body

## 1.1 Demand Analysis

Our Normal Form Decomposition Widget may have more prominent needs now or in the future in the following areas:

* Teachers' teaching aids
* Self-testing gadgets for student use
* For analyzing large databases and providing reasonable storage recommendations.

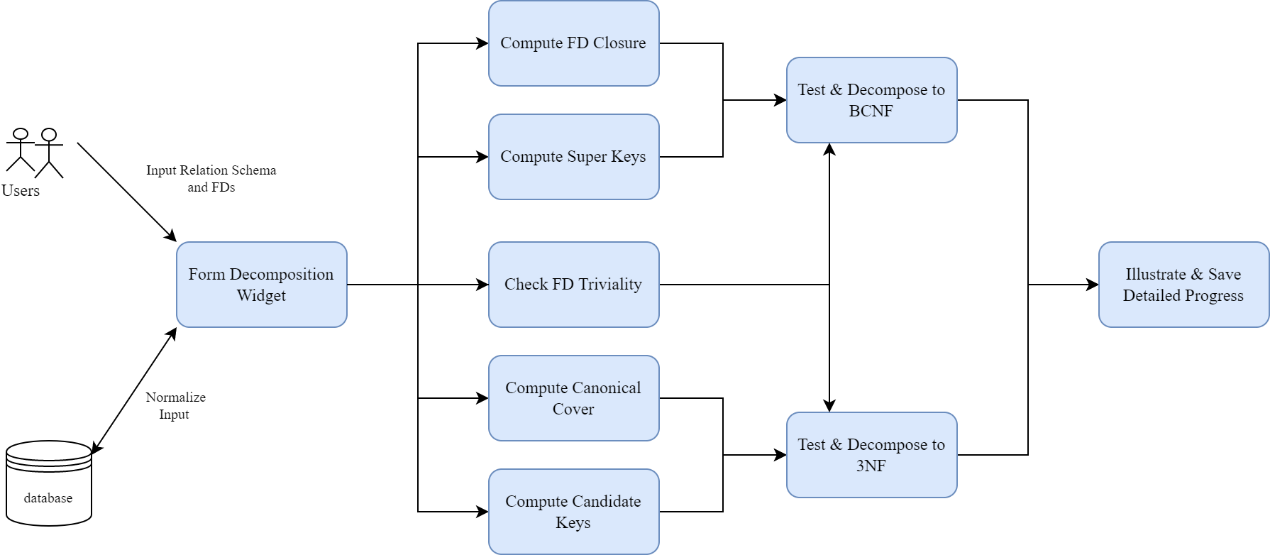
We believe that our project requirements analysis is reasonable for several reasons:

* The gadget is able to accept input relational schemas and functional dependencies and provide corresponding paradigm decomposition results to help teachers clearly demonstrate and explain paradigm concepts and processes.
* Students can use the tool to compare and contrast the results of paradigm decomposition given by the tool, thus checking their understanding of paradigm theory and deepening their knowledge of database design
* For large database systems, paradigm decomposition is important to improve data consistency and reduce data redundancy. This tool can accept complex relational schemas and function dependencies and perform efficient paradigm decomposition. It helps database administrators and analysts to understand the structure and dependencies of data and provide reasonable storage recommendations to optimize database performance and maintainability.

## 1.2 General Design

Users could input relation schemas and functional dependencies in the front-end, system then passes them to the back-end as strings and stores them in sqlite as string type (for easy management), the back-end calculates canonical cover and the closure of functional dependencies, then calculates attribute closures to obtain super keys and candidate keys further. Upon acquired all support materials, system will then test all function dependencies users have input, decomposing the schema once function dependencies destroying the 3NF/BCNF. Detailed outputs of information about testing and decomposition will be output through user interface and will be saved to txt files automatically.

## 1.3 Detailed Design



### 1.3.1 Normalize Input (copy 你的函数以依赖添加的说明)

### 1.3.2 Check FD Triviality

By defining the function is\_trivial\_fd(fd), we can determine whether a function dependency is a mundane function dependency with the following logic:

The receiver function depends on fd, where fd contains the left-hand attribute set lhs and the right-hand attribute set rhs.

Check if the right attribute set rhs is a subset of the left attribute set lhs. If it is a subset, return True. If it is not, return False.

The logic of the check\_trivial\_fds(fd) function is as follows:

The receiver function depends on fd. Call is\_trivial\_fd(fd) to check if it is a trivial function dependency. Returns False if it is not a trivial function dependency. If it is a trivial function dependency, return True.

These functions are used to check whether a function dependency is a trivial function dependency. is\_trivial\_fd function verifies that the right attribute set is a subset of the left attribute set to determine whether it is a trivial function dependency. check\_trivial\_fds function calls is\_trivial\_fd and returns the appropriate boolean value based on the result.

### 1.3.3 Test & Decompose of BCNF

#### 1.3.3.1 Compute Super Keys

#### 1.3.3.2 Compute FD Closure

### 1.3.4 Test & Decompose of 3NF

#### 1.3.4.1 Compute Canonical Cover

Given a functional dependency set F, the steps for computing canonical cover can be concluded as follow:

1. Break up every functional dependency in F whose right hand side has more than one attributes to get . All functional dependencies in have only one attribute on their right hand side.
2. For each in , compute under and under . If two are the same, delete .
3. After finishing step 2, we get . For each in whose has more than one attribute, let where X is any attribute in . Compute under . If includes X, then X is extraneous in so we can delete it.
4. After finishing step 3, we get . Combine all functional dependencies in that has the same left hand side, so we get the canonical cover .

#### 1.3.4.2 Compute Candidate Keys

Input: Attributes & FDs

Output: list of candidate keys (candidate\_keys).

Algorithm:

1. first, generate the power set `sorted\_power\_set` of the attribute by calling the `powerset` function, where the `powerset` function returns all possible subsets of the attribute, ordered by size. 2. for each subset of the attribute, the power set `sorted\_power\_set` is used to generate the power set `sorted\_power\_set` of the attribute.

2. for each subset `set\_`, convert it to a list of attributes `attrs`.

3. use the `is\_super\_key` function to check if the current attribute list `attrs` is a super key. 4. if the current attribute list is a super key, use the `is\_super\_key` function to check if the current attribute list is a super key.

4. if the current attribute list is a super key:

- If the candidate code list is empty. add the current attribute list to the candidate code list. Within the first loop: If the candidate list is not empty, iterates through each candidate in the candidate list. In the second nested loop: Converts the current candidate code and the candidate codes in the candidate code list to a character list; Check if the current candidate code is a subset of one of the candidate codes in the candidate code list. If it is a subset, the current candidate code is not the smallest super key and the candidate code is skipped.If the current candidate code is the smallest super key, add it to the candidate code list.

5. Return the list of candidate codes.

This code determines the candidate codes by generating a power set of attributes and using the concept of super key. It checks if each subset satisfies the condition of a super key, and if it is a super key and is not a subset of an already existing candidate code, it adds it as a candidate code to the candidate code list. The final list of all candidate codes is returned.

### 1.3.5 UI Design

The user interface for this widget mainly uses TKinter (that is, Python's standard Tk GUI toolkit interface) to complete, through the ‘TK’ function to set the two UI interface window size, color and whether to support resizing. In addition, the UI's labels and text boxes and buttons are also set through TKinter. After running the program, the UI window that outputs the results is set to a hidden state, and the input window pops up in the foreground for the user to enter the relational mode and function dependencies.

There are two buttons in the input window to let the user decide to perform 3NF decomposition or BCNF decomposition. After the user clicks either of the "Submit" buttons, the program will check the user's input format. If the user enters any characters that do not meet the requirements, the program will pop up a new prompt box to prompt the user to "the input format is wrong, please modify and resubmit". After the user enters the content conforming to the specification, the program will separate the function dependencies by "|", separate the left and right parts of the function dependencies by "->", and then separate the elements by ", "and then load them into the list respectively for back-end calculation. If the user clicks the Calculate 3NF button, the list will be submitted to the back-end program calculating 3NF for calculation. If the user clicks the Calculate BCNF button, the list is submitted to the back-end program that calculates the BCNF. After the calculation is completed, the output window will pop up to the foreground, the user can click "Save results to txt" on the window to save the entire calculation result as txt file, two additional buttons on the interface can be used by the user to quickly browse the output result from the beginning or from the end, in addition, if the user chooses to calculate the BCNF decomposition, In addition, a tree diagram is displayed to help users understand the decomposition process and the final result.

In the BC decomposition step-by-step instructions, we added extra tree diagram drawing, the next will briefly explain the principle of tree diagram drawing:

In the UI function for tree plotting, there are several key functions as follows:

1. `list\_to\_tree(midRes)`: This function converts a given list `midRes` into a tree structure. It does this by iterating over the elements in the list and constructing connections between tree nodes based on parent-child relationships. It eventually returns the root node.

Input: `midRes`, a list representing the structure of the tree.

Output: the root node of the tree.

2. `draw\_node(canvas\_here, tree\_here, x, y, dx, dy, font\_size\_here)`: This function is used to draw the tree nodes and connectivity lines. It does this by iterating over the nodes of the tree and drawing the nodes and connecting lines on the canvas.

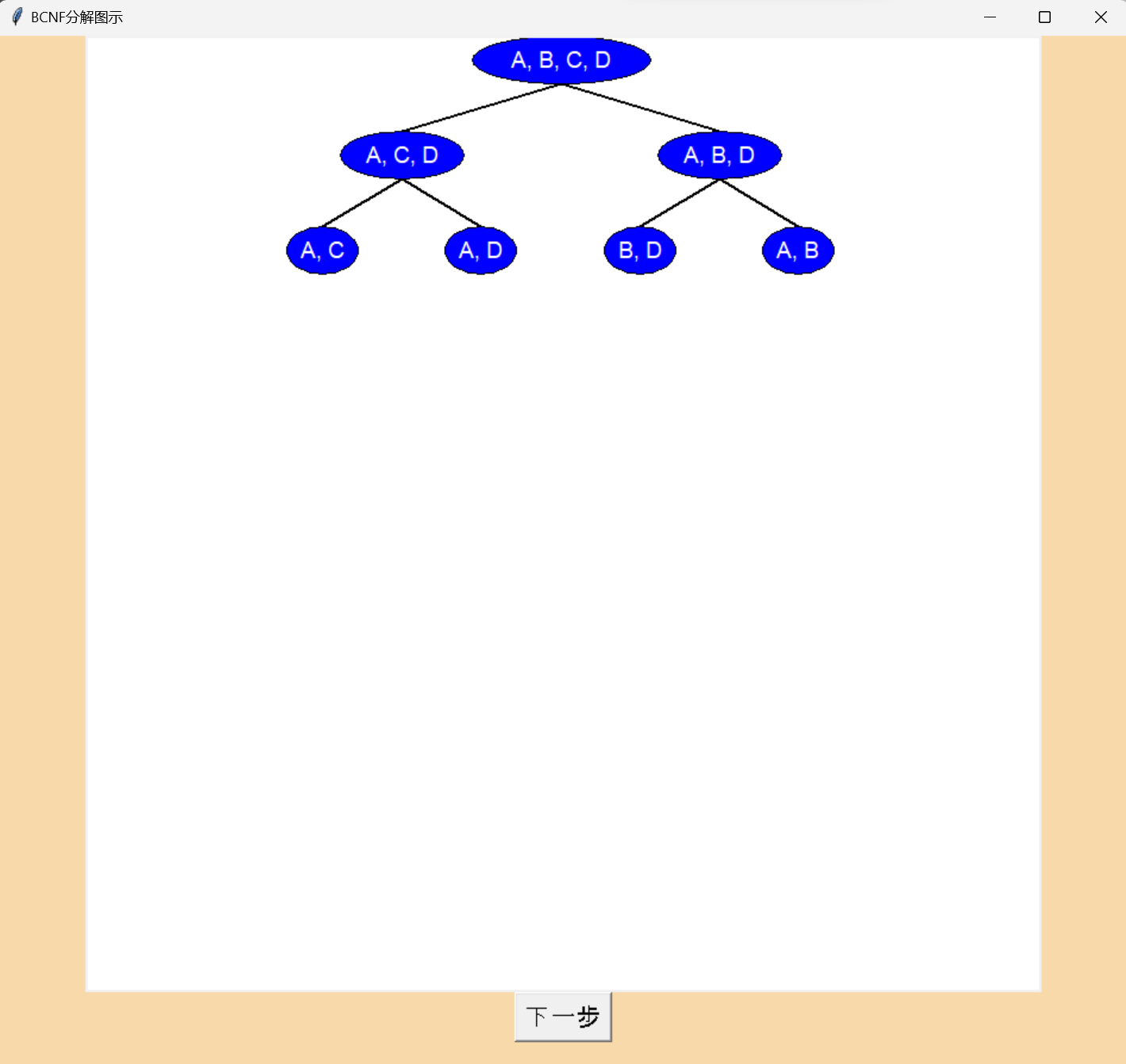
Input: `canvas\_here`, the canvas object; `tree\_here`, the root node of the tree; `x`, the horizontal coordinates of the nodes; `y`, the vertical coordinates of the nodes; `dx`, the horizontal spacing between the nodes; `dy`, the vertical spacing between the nodes; and `font\_size\_here`, the font size for the node text.

Output: nodes and connecting lines drawn on the canvas.

**Key algorithm:**

- The `list\_to\_tree(midRes)` function uses the stack `parent\_stack` to keep track of the parent nodes, and maintains proper parent-child relationships by iterating over the list elements and building the tree nodes. Parent node out of stack operations are handled by determining if the node is None.

- The `draw\_node(canvas\_here, tree\_here, x, y, dx, dy, font\_size\_here)` function implements depth-first tree node drawing by calling itself recursively. It draws the ellipse of the node, inserts the node text, and draws connecting lines based on parent-child relationships.

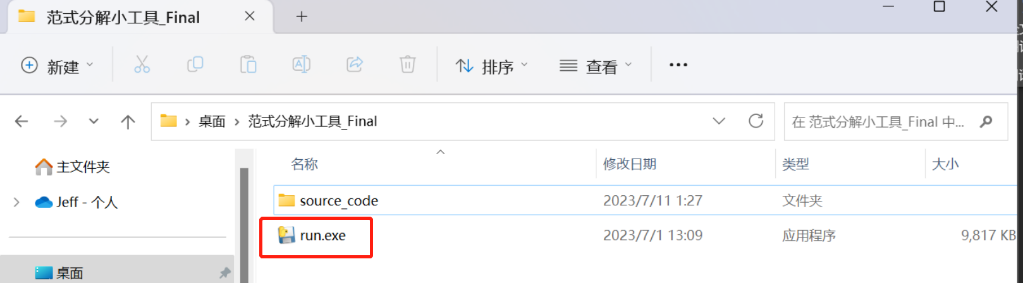


The inputs to these functions are primarily parameters related to the tree structure and drawing, and the output is the representation of the drawn nodes and connecting lines on the canvas. Together, these functions accomplish the visualization of converting a given list into a tree structure and drawing that tree structure on the canvas. Note that our illustration focuses on the decomposition process, as the final sub-table is not necessarily equivalent to the set of all tables at the leaf level of the table, and we should remove redundancy and delete tables with fewer attributes.

|  |
| --- |
|  |

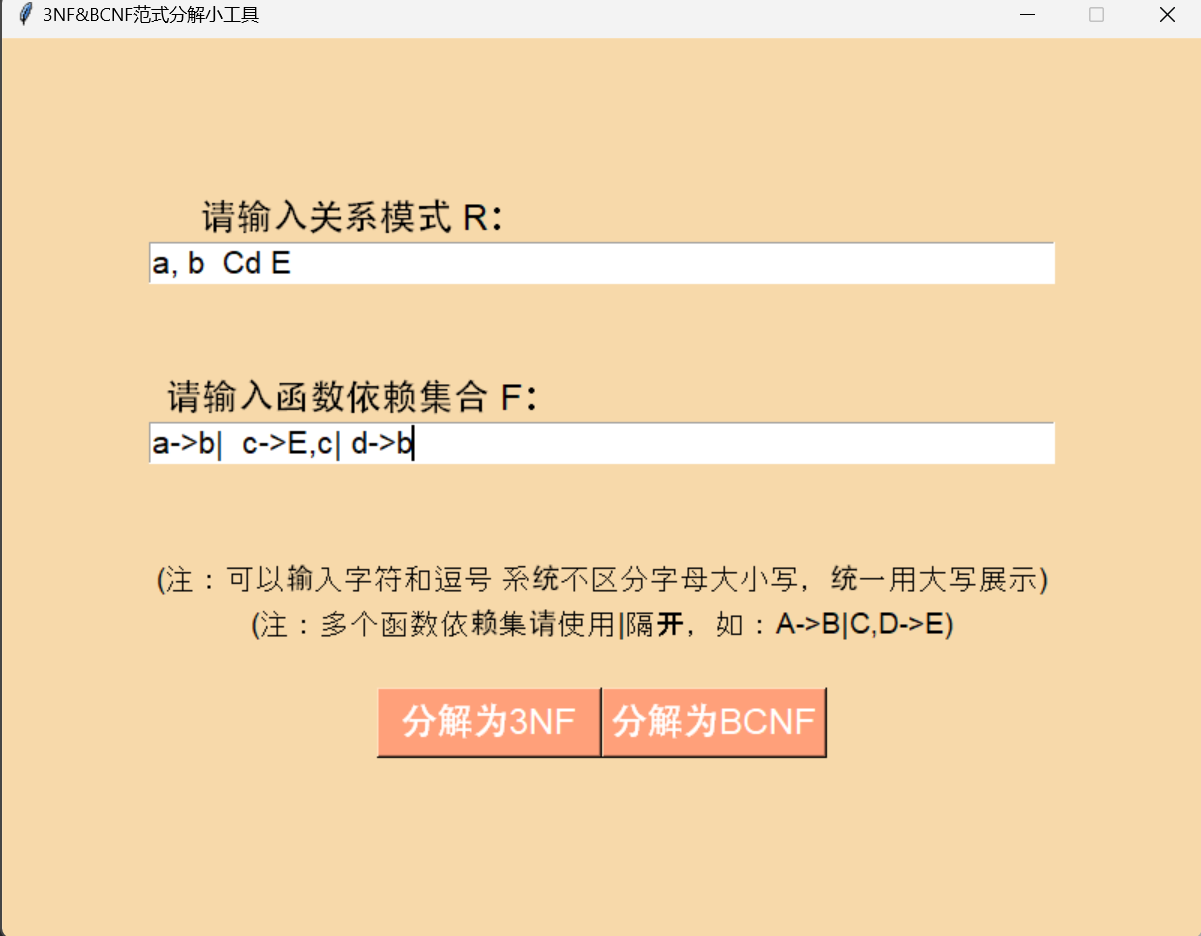
# 2. System Tutorial

## 2.1 Application Installation

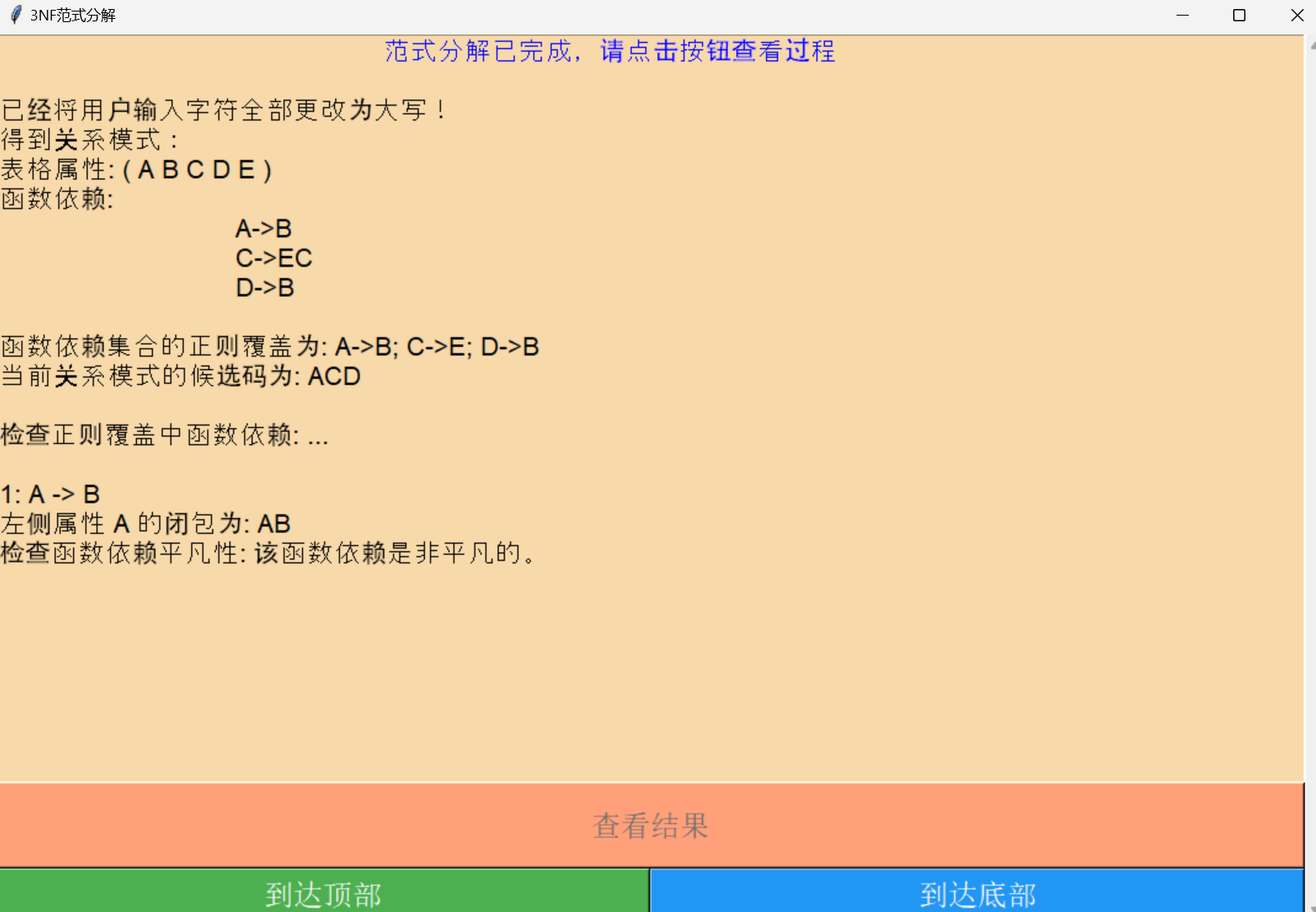
To use our decomposition widget, simply have one of the exe files in the image below. All results saved by running will be consistent with your exe file directory.

## 2.2 Click to Run & Input Your Schemas

Note that users are only allowed to input characters, commas and blanks in attributes and additionally an arrow string ‘->’ for entering FDs. Our input is set to be case insensitive.

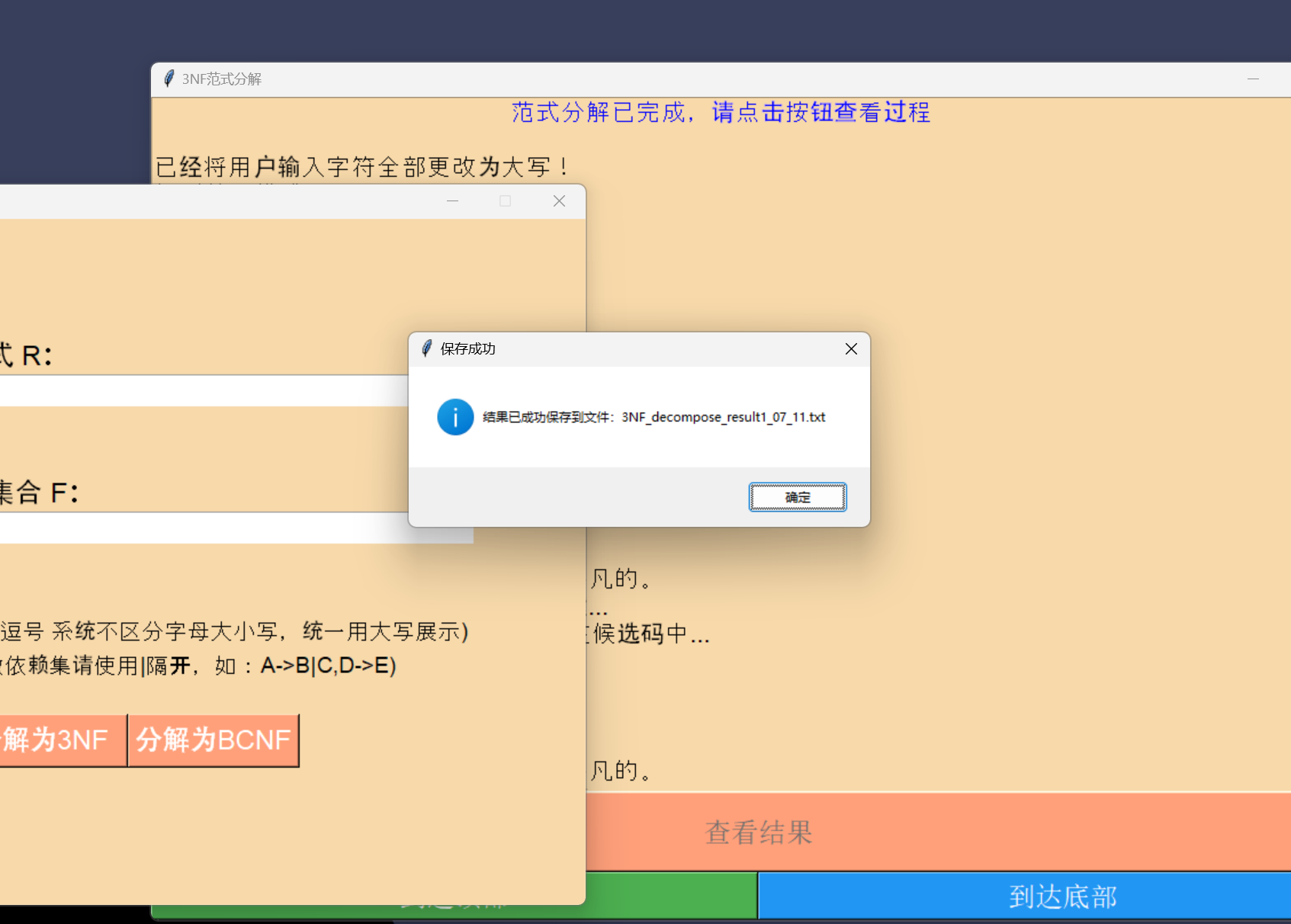


## 2.3 Click Submit Button to Get Result

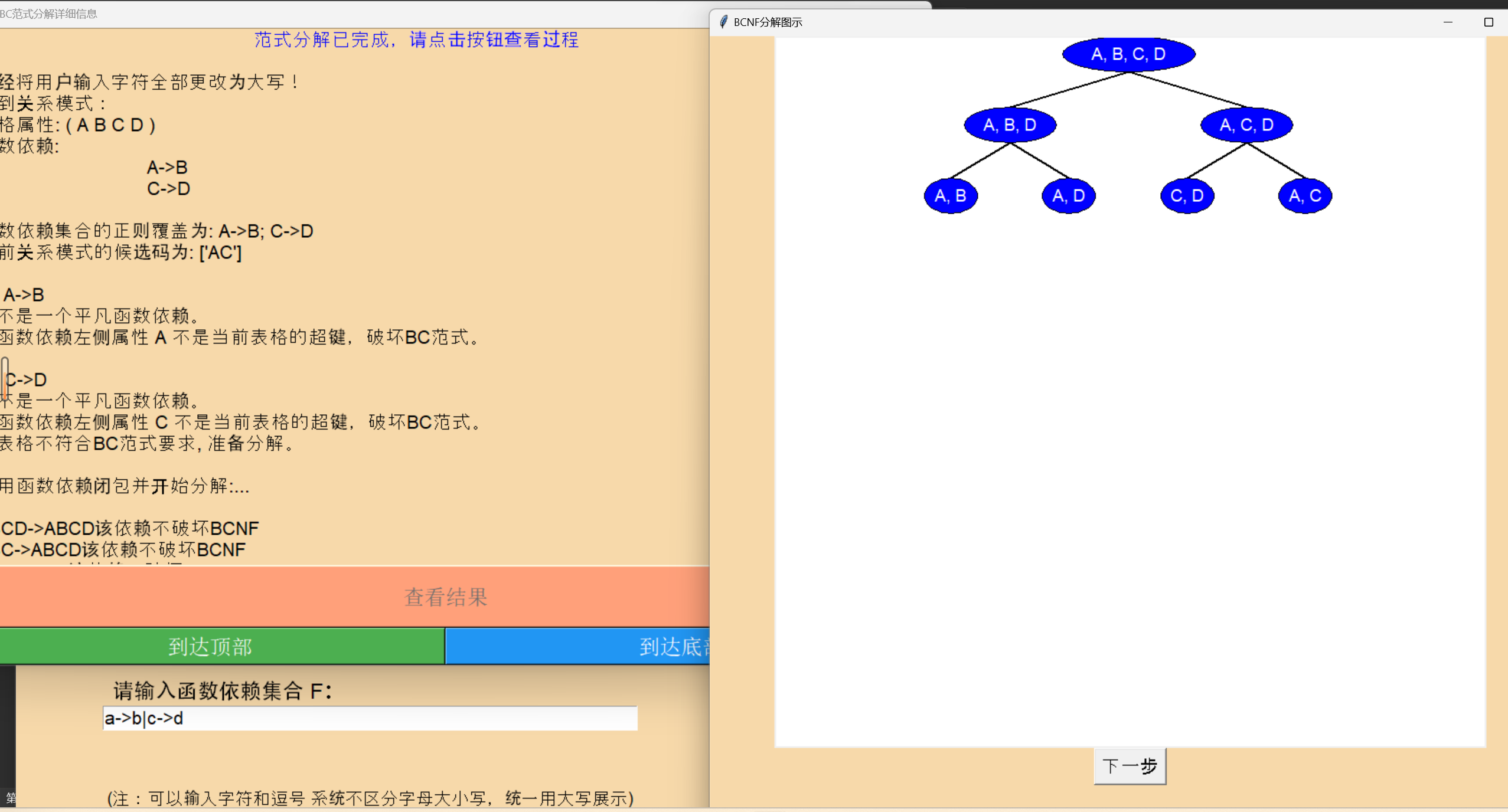


Note that the results will be shown steps by steps and there’s a time delay of 1 second for each info to make the presentation dynamic and user-friendly. There are 2 buttons on the bottom of the application window, one is for the shortcut navigating to the top of whole results and the other one is for the opposite side.

After the whole results are output on screen, the system will automatically save the result to text files under the same directory of this program.



Specifically, for BCNF’s result, there will be 2 windows prompted once you click the button for ‘Decompose to BCNF’. One of 2 windows remains quite the same as 3NF, showing messages with a slight delay steps by steps. The other interface shows a tree view of the decomposition process, where the next decomposed sub-table is dynamically created by clicking on the button 'Next Node'.



# 3 Group Discussion

|  |  |
| --- | --- |
| Date | Important Discussion |
| 2023.4.15 | Project selection and initial brainstorming |
|  |
|  |
| 2023.5.12 | Finalized the software functions and user interface design |
| 2023.5.21 | Implemented the attribute closure calculation algorithm |
| 2023.6.10 | Conducted thorough testing and identified bugs |
| 2023.6.18 | Resolved critical issues and achieved stable software performance |
| 2023.6.23 | Documented software features and prepared for software demonstration |
|  |
|  |
| 2023.7.5 | Finalized the project documentation and report |

# 4 Personal summary

## 4.1 邓靖丰

In this group training, it is the first time that I have ever been a team leader, and I was mainly responsible for the construction of the underlying framework, the calculation of the table candidate code, the implementation of the 3NF testing and decomposition algorithm, the front-end and back-end architectures, as well as the illustration of part of the UI and BC decomposition in the group. First of all, being the leader of the group is a big challenge for me by definition, because I have never been in this kind of responsibility before, so I feel guilty for the lack of group cooperation and team communication, but I believe I will do better in the future.

Through this practical training, I have a deeper understanding of two very important database BCNF and 3NF, and a deeper impression of the candidate code and decomposition algorithm; through the role of architect, I also have a clearer understanding of the front and back-end division of labor and cooperation, and the relationship between each other, as well as the importance of the architect for the project. Through this practical training, I not only learned a lot of front-end implementations, consolidated the combination of textbook knowledge and programming application ability, but also accumulated valuable experience for my future work.

## 4.2 张治展

The training not only improved my coding skills, but also deepened my understanding of normal form decomposition. For example, I found out that the result of BCNF decomposition could contain extra relations that should be deleted (see step 4 in BCNF decomposition in Detailed Design), which I did not know before. Overall, I think this is a meaningful experience.

## 4.3 晏璐涛

Throughout the tool development process, I was responsible for the implementation and optimization of the underlying code, which was a very challenging but also rewarding learning experience. Being responsible for the underlying code implementation gave me a deep understanding of the concepts of function dependencies and general overrides, and learned the associated algorithms and data structures. This program gave me the opportunity to learn and apply new technologies and tools. At first, I didn't know what 3NF was, but the gradual learning feeling was not so complicated, and I constantly improved my technical ability and problem-solving ability by solving practical problems and overcoming technical problems. I am very grateful for the opportunity to participate in this project and contribute to the realization of a useful tool.

## 4.4 古旭鹏

After completing a UI interface that allows users to enter functional dependencies and relational patterns, and automatically separates different functional dependencies, I learned some valuable lessons. First of all, writing such a UI requires a good user experience design. Using the Tkinter library, I learned how to create user-friendly interface elements such as labels, text boxes, and buttons, as well as how to set layouts and styles. This makes it easy for the user to enter function dependencies and relational patterns, and the program can determine if the user input meets the requirements. Second, processing user input involves validation and parsing of the data. I learned how to write logic to check that the user is typing in the correct format and give the appropriate error message. And by writing the corresponding parsing algorithm, I was able to automatically separate the functional dependencies entered by the user and pack the left and right elements into the list. This greatly improves processing efficiency and accuracy.

I also encountered some challenges during the development process. Because the format of function dependencies can vary, I need to take into account the various boundary cases and adopt a strategy for handling them accordingly. At the same time, ensuring the robustness and reliability of the program also needs to be focused on avoiding potential errors and exceptions.

After completing this project, my ability to process user input data and develop user interfaces has improved significantly. I also learned more deeply that a good user experience is critical to the success of software applications. At the same time, I realize that code maintainability and extensibility are essential for long-term development projects.

All in all, by writing UI interfaces that support function dependency input and relational schema definition, I have mastered UI design, user input validation and data parsing skills, and developed good problem solving and teamwork skills. This will have a positive impact on my future software development work and prepare me for bigger challenges.

# 5. Work Allocation & Schedule

邓靖丰:

Underlying code implementation, front-end and back-end architecture (computing candidate codes, 3NF testing and decomposition algorithms, BC tree mapping)

张治展:

Underlying code implementation (computation of function-dependent closures, computation of arbitrary property closures, computation of function-dependent regular coverings, BCNF decomposition algorithm)

晏璐涛:

Underlying code implementation (computing arbitrary property closures, BCNF testing and decomposition algorithms)

古旭鹏:

Front-end code implementation (all front-end interface framework builds, output back-end calculations)

|  |  |
| --- | --- |
| Main Work | Duration |
| The underlying algorithm implementation | 2周 |
| Modify the bottom layer, and the front-end architecture to build an overall program with an interactive interface | 2周 |
| Try to input some schemas with poor function dependencies, observe the results, and modify them with special judgments | 3-4周 |
| Comprehensive modification to optimize the performance of the program | 1周 |