DIRECTORY

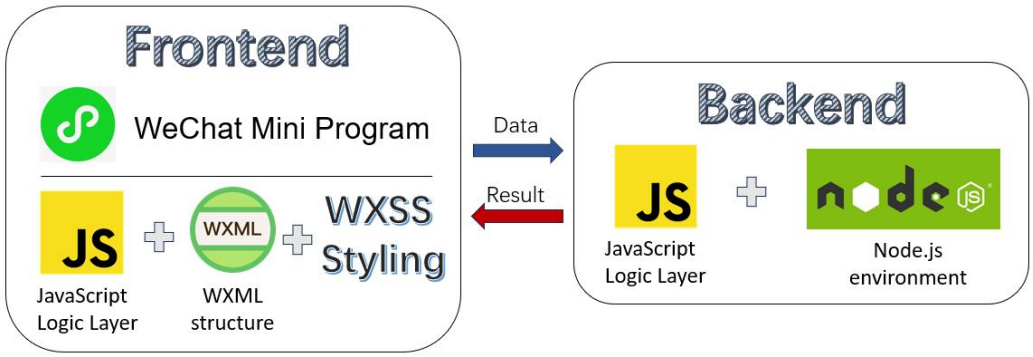
1. Requirement Analysis

Through research, we found that database is an important compulsory course from management to computer science and other engineering disciplines, and "relational database design" is an important part of it. Whether in practical application or understanding of the course, it is very meaningful to show the decomposition process clearly. Therefore, there is a great demand for a "normalization" teaching aid.

Therefore, based on the classroom learning needs of students and the design needs of developers, we have developed a "normalization" WeChat Mini Program that will show the BCNF decomposition and 3NF decomposition in detail, and provide other computational functions of related properties.

2. Outline Design

2.1 General framework diagram



2.2 System function diagram

3. Detailed Front-End Design

3.1 Functional overview

Introduction: Introduction to the use of the WeChat Mini Program

Configuration: User-defined input data, including relational schema R and function-dependent set F.

Select function: User selects different calculation modules according to the requirements, such as calculating F+, calculating BCNF decomposition results, etc.

3.2 User interface logic structure diagram

3.3 Key algorithm description

3.3.1 swiper component optimization

The swiper component is often used by merchants to display rotating images, in which the subcomponent swiper-item is often put into images for rotation. This application puts the button group into it and controls the number of swiper-item by the number of buttons, which makes use of the smooth animation and easy layout of the

swiper component to achieve a better visual effect and user experience.

3.3.2 Animating buttons using the animation property list:

This program provides fade-out and fade-in animations for the selected relationship, and also provides a withdrawal operation as well as a determination operation for the relationship selection. The fade-in and fade-out effects need to be triggered frequently during these operations. This program records the animation parameters needed for the recovery of the operation when the animation assignment operation occurs. Due to the mutually exclusive nature of the fade in and fade out operations, this process can easily realize the repeated playback of the fade in and fade out animation.

3.3.3 Using getCurrentPages to achieve more reasonable page jumps:

Since navigateTo and navigateBack do not really exit the page during the page jump, they cannot release memory in time. In the user, repeatedly entering the page and then exiting again can lead to serious memory leaks and page jams. This program uses getCurrentPages to navigate to the page that needs to be freed and manually calls the onUnload method of the page object to fix the memory leak.

4. Detailed Back-End Design

4.1 Basic function module

4.1.1 Function Overview

a. Compute arbitrary attribute closures.

b. Compute the function dependency closure F+.

c. Compute candidate codes.

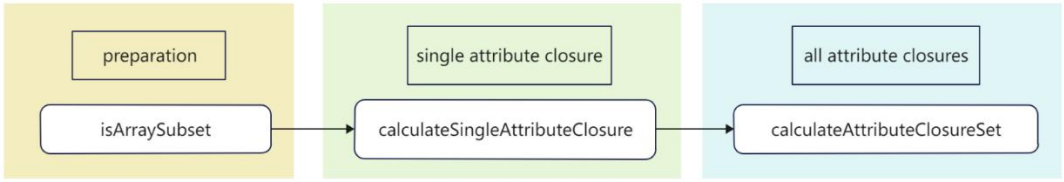
d. Compute the canonical cover set

4.1.2 Key algorithm logic

a. Compute arbitrary attribute closures.

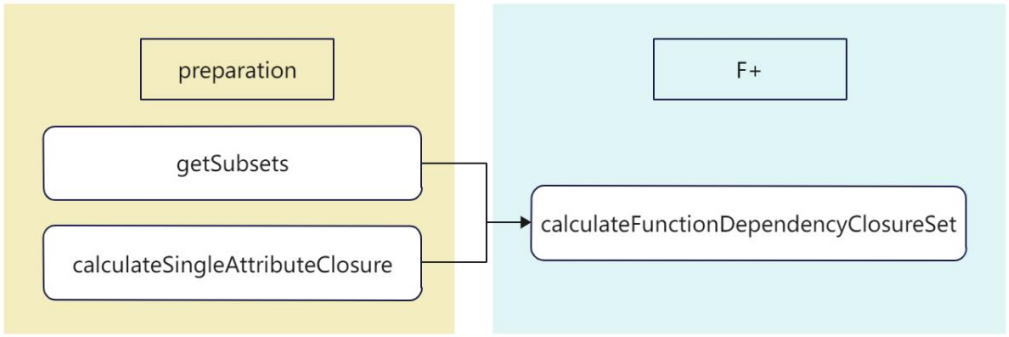
```
result := α;
repeat
        for each functional dependency β → γ in F do
            begin
                if β ⊆ result then result := result ∪ γ;
            end
until (result does not change)
```

Figure 8.8 An algorithm to compute α^+, the closure of α under F.



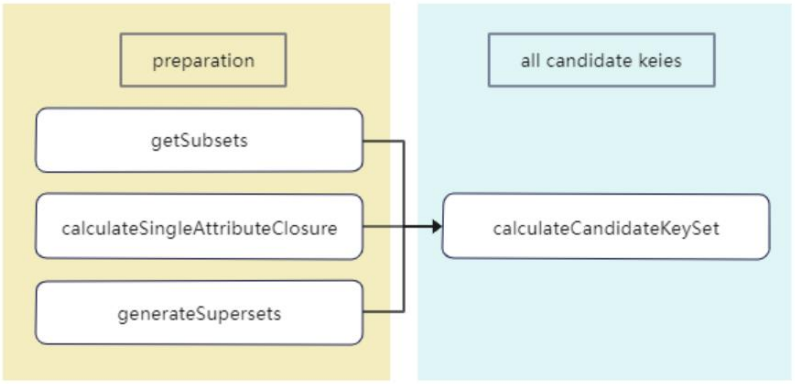
b. Compute the function dependency closure F+.

1. $F^+ \leftarrow F$
2. *For each* $\gamma \subseteq R$,
   2.1 *we compute* γ^+.
   2.2 for each $S \subseteq \gamma^+$, *we compute* $F^+ \leftarrow \{\gamma \rightarrow S\} \cup F^+$.
3. outputs F^+



c. Compute candidate codes.

```
SortByLength ( subsets( R ) )
for each subset Si in R do
    begin
        If (! mark( Si ) ) and ( R ⊆ closure( Si ) )
            candidateKeySet := candidateKeySet ∪ Si
            mark ( supersets( Si ) )
    end
```



Proof

For the candidate key ***K*** determined by our algorithm, it is assumed that ***K*** is not the true candidate key. According to this assumption, the real candidate key is either traversed before ***K*** or after ***K***.

-> If the real candidate key is traversed before ***K***, then after we traverse the real candidate key, ***K*** should be marked and should not actually be traversed again.

-> On the other hand, the real candidate key cannot be traversed after ***K*** either, because the length of ***K*** is shorter than the real candidate key, and by the definition of candidate key, the set of attributes longer than ***K*** cannot be a candidate key.

Therefore, according to the above reasoning, the assumption that ***K*** is not a real candidate code is not valid.

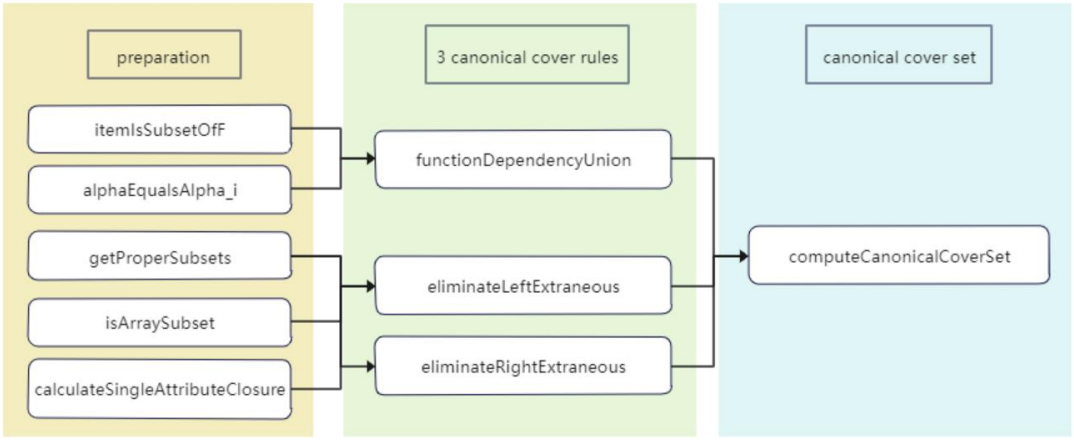
d. Compute the canonical cover set

- How to check attribute x is extraneous, given $\alpha x \rightarrow \beta$ and F ?
  1. *computes* α^+ *under* F
  2. $\alpha^+ \supseteq \beta$?

- $\alpha \rightarrow \beta x$ in F. How to test if attribute x is extraneous
  1. $F' = (F - \{\alpha \rightarrow \beta x\}) \cup \{\alpha \rightarrow \beta\}$
  2. *computes* α^+ *under* F'
  3. $x \in \alpha^+$?

$F_c = F$
repeat
  Use the union rule to replace any dependencies in F_c of the form $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$.
  Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β.
    /* Note: the test for extraneous attributes is done using F_c, not F */
  If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$ in F_c.
until (F_c does not change)

Figure 8.9 Computing canonical cover.



4.1.3 Key Algorithm Functions

a. **isArraySubset(listA:any[], listB:any[]):bool** To determine whether listA is a subset of listB

b. **getSubsets(sArg:any[]):any[]** To generate subsets

c. **getProperSubsets(sArg:any[]):any[]** To generate proper subsets

d. **generateSupersets(subsets:any[],input:any[]):any[]** To generate supersets

e. **itemIsSubsetOfF(item:any[], F:any[]):bool** To determine if item is a subset of F

f. **alphaEqualsAlpha_i(arr1:any[], arr2:any[]):bool** To determine whether arr1 and arr2 are equal

g. **calculateSingleAttributeClosure(alphaArg:any[],F:any[]):any[]** To calculate a single attribute closure

h. **calculateAttributeClosureSet(F:any[]):any[]** To calculate all attribute closures

i. **calculateCandidateKeySet(R:any[], F:any[]):any[]** To calculate candidate codes.

j. **functionDependencyUnion(F:any[]):any[]** To merge the same relations on the left side of function dependencies

k. **eliminateLeftExtraneous(unionF:any[]):any[]** To eliminate the redundant **properties on the left side of the function dependency**

l. **eliminateRightExtraneous(unionF:any[]):any[]** To eliminate redundant properties on the right side of function dependencies

m. **computeCanonicalCoverSet(F_Arg:any[]):any[]** To compute the regular cover set

4.2 BCNF Module

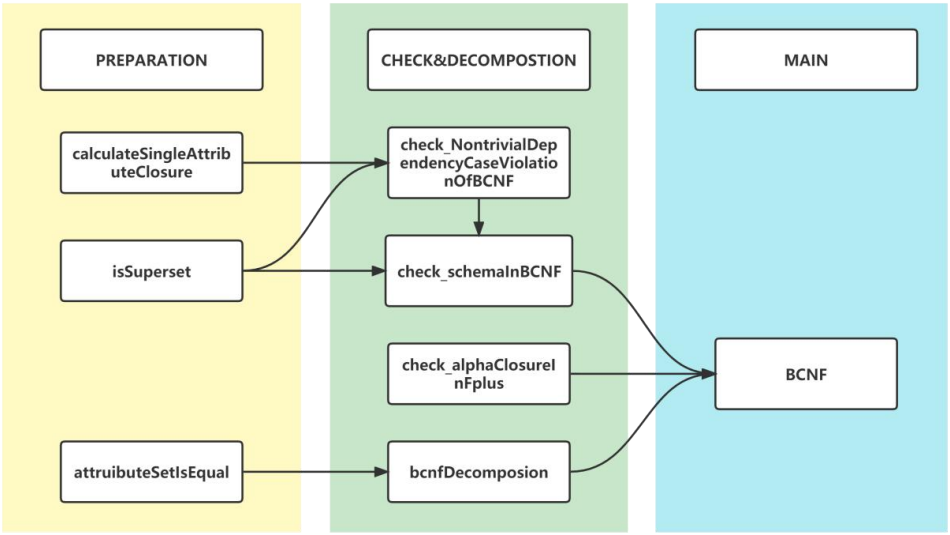
4.2.1 Function Overview

a. Determine whether R belongs to BCNF

b. BCNF decomposition

c. Show the computational steps

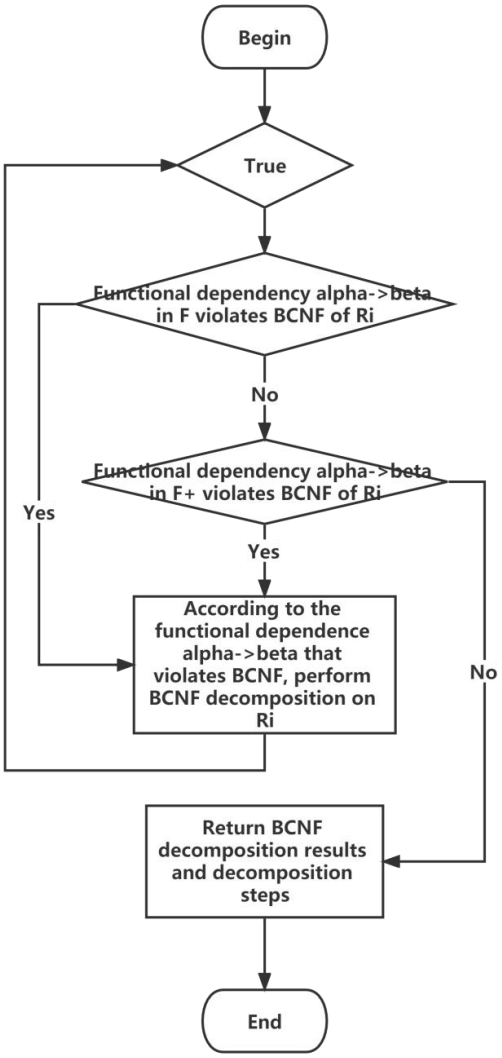
4.2.2 Key algorithm logic



a. Determine whether R belongs to BCNF

```
For((α → β) in F){
    If(α ⊇ β  and  Ri ⊇ (α ∪ β)){
        If(not (α+⊇ β)){
            return Functional dependencies that violate BCNF;
        }
    }
}
```

b. BCNF decomposition



4.2.3 Key Algorithm Functions

a. **check_NontrivialDependencyCaseViolationOfBCNF(s:any[],schemaR:any[],F:any[]):bool** To determine whether a non-trivial dependency violates BCNF of Ri

b. **Check_schemaInBCNF(schemaR:any[],F:any[]):bool** To determines whether Ri is a BCNF

c. **BCNF(R:any[],F:any[]):any[]** Given an attribute set R and a function dependency set F, perform a BCNF decomposition on the attribute set R

4.3 3NF Module

4.3.1 Function Overview

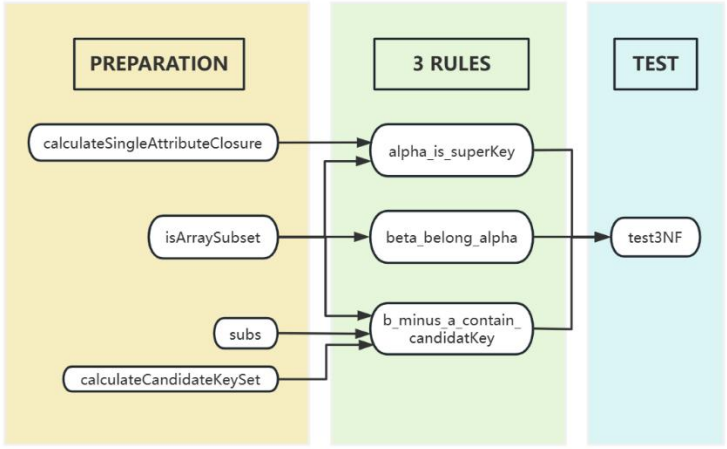
a. Determine if R is a 3NF.

b. 3NF decomposition.

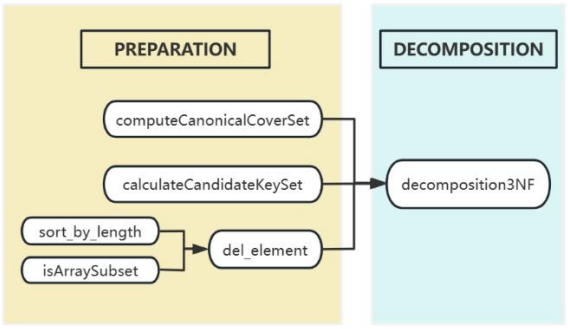
c. Show the calculation steps

4.3.2 Key algorithm logic

a. Determine if R is a 3NF.



b. 3NF decomposition.



4.3.3 Key Algorithm Functions

a. **beta_belong_alpha(alpha: any[ ], beta: any[ ]): bool** To determine if beta belongs to alpha

b. **alpha_is_superKey(alpha: any[ ], F: any[ ], R: any[ ]): bool** To determine whether alpha is a super code

c. **subs(A: any[ ], B: any[ ]): bool** To implement set subtraction

d. **b_minus_a_contain_candidatKey(alpha: any[ ], beta: any[ ], F: any[ ], R:**

any[ ]): bool To determine whether beta-alpha is contained in a candidate code

e. **test3NF(F: any[ ], R: any[ ]): any[ ]** This function tests whether a table conforms to 3NF. the output breaks the 3NF function dependencies; when the output is [ ], the table conforms to 3NF.

f. **sort_by_length(result: any[ ]): any[ ]** To sort by the length of the elements in ascending order. The sorted table is output.

g. **del_element(result: any[ ]): any[ ]** To delete a table with duplicate elements. Outputs the deleted table.

h. **decomposition3NF(F: any[ ], R: any[ ]): any[ ]** To split the table according to 3NF. Output the decomposed table.

i. **areListsEqual_noorder(list1: any[ ], list2: any[ ]): bool** To determine whether two lists have the same elements, regardless of order.

j. **removeDuplicates(arr: any[ ]): any[ ]** To remove duplicate elements from a two-dimensional array. The output is the table after de-duplication.

k. **showStep(F: any[ ], R: any[ ]): any[ ]** To show step 2 of the UI. output [Fc,selected candidate code, candidate code that needs to be a separate table].

l. **ThreeNF(F: any[ ], R: any[ ]): any[ ]** This function integrates the function of 3NF, check first, and then generate a table that conforms to 3NF. The output is whether 3NF is broken, and if so, which functions depend on breaking 3NF. the output conforms to the 3NF table.

5 Software Testing

5.1 Testing of basic functions

Attribute closure

Calculate F+

Candidate code

Judge whether R in BCNF

BCNF decomposition

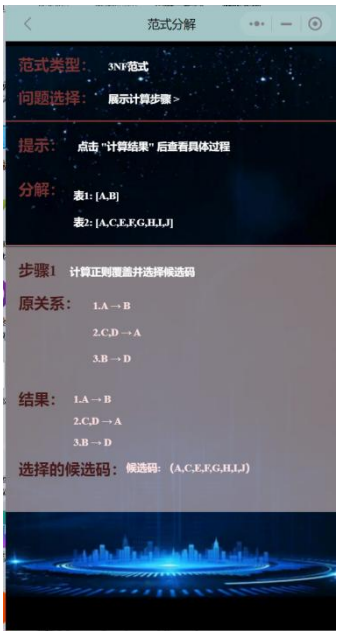
Show computation step (BCNF)

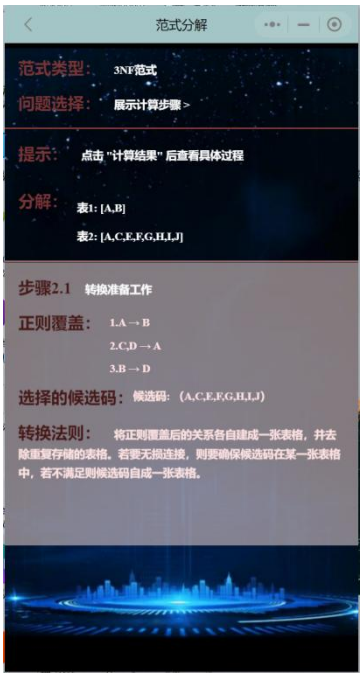
Canonical cover Set

Judge whether R in 3NF

3NF decomposition

Show computation step (3NF)



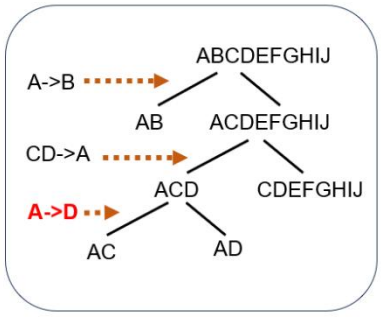


5.2 Testing of advantageous functions

5.2.1 Integrity of BCNF decomposition

The solution to the problem of "BCNF is not completely decomposed when using only F". In the case shown below, we have achieved a complete decomposition by considering F+, and we have taken into account the possibility of incomplete decomposition on both left and right sides. For example, in the figure below, we have further decomposed the left table.

situation Manual decomposition diagram Program decomposition results



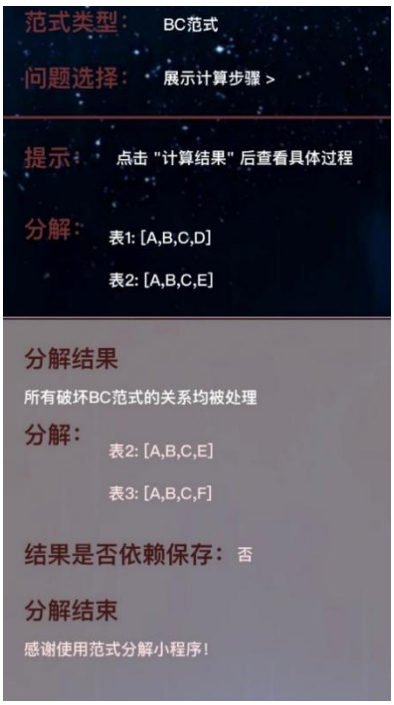
5.2.2 3NF decomposition by teaching purpose

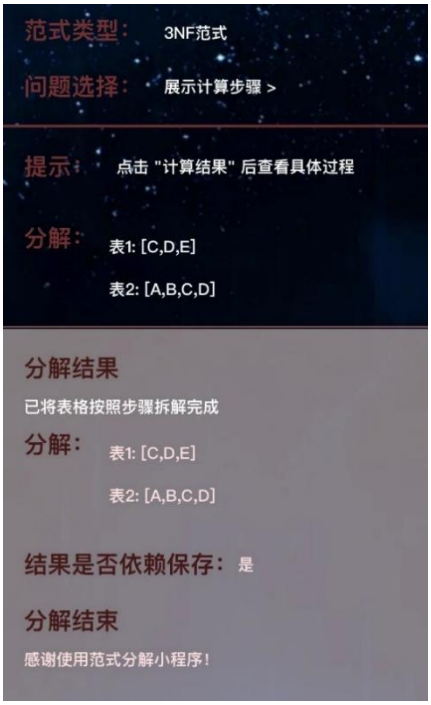
Satisfy 3NF, but still decompose, in order to show the teaching effect.

5.2.3 Data Configuration Synchronization

a. Difference in conformity: Explain that 3NF is the minimum relaxtion of BCNF

b. Differences in Dependency Preservation: BCNF is not necessarily preservation-dependent, 3NF is definitely preservation-dependent





5.2.4 Quick solution to "lots of properties and lots of function dependencies"

a. For the computation of F+, we replaced Armstrong's theorem with attribute closure computation, which greatly reduces the time complexity. In the following figure with more than 60,000 data, the computation time of F+ is close to 0.

b. On the BCNF decomposition, we process F first, and if the decomposition result does not satisfy the dependency preservation, we then process other function dependencies in F+, which greatly reduces the time complexity.

As shown in the figure below, the computation time of the whole display process is about 1 second.

6. Software Demo

6.1 Introduction of the WeChat Mini Program module

The mini program offers both BCNF and 3NF decomposition. Their UI designs have the same internal logic and are divided into 3 modules: Introduction, Configuration and Results.

The introduction module introduces the basic content of the mini program. The configuration module is used for user-defined input relationship pattern R and function dependency set F. The result module allows users to select a problem and calculate the result.

6.2 Data Configuration

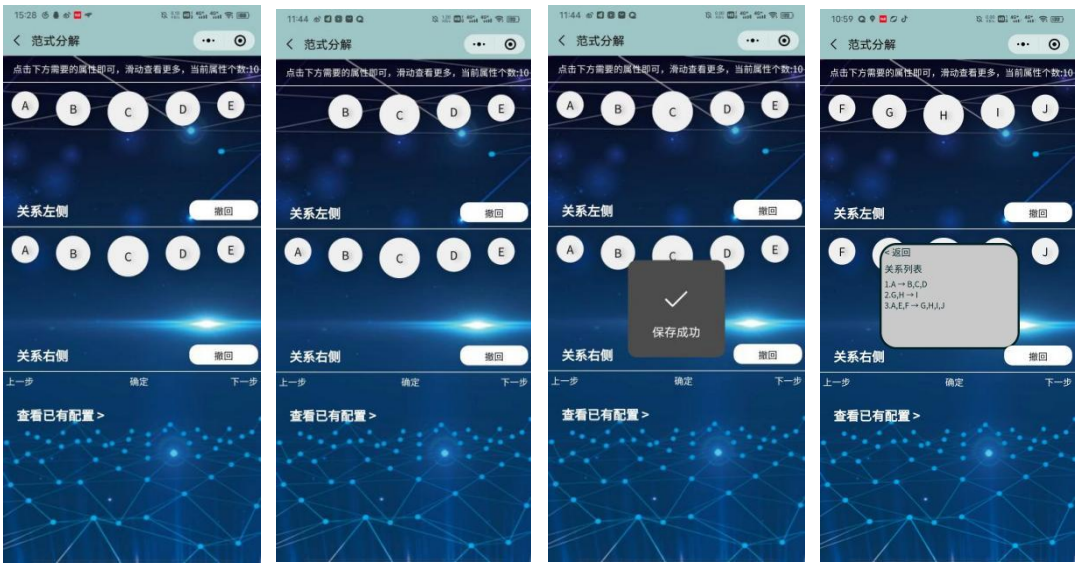
The BCNF and 3NF data configuration logic is the same. It can be divided into the following steps: (BCNF is used as an example here)

6.2.1 On the configuration module page, click "配置".

6.2.2 Select the number of attributes in the relationship schema R. Drag the scroll bar to select the number of attributes. Click "下一步" when you are done.

6.2.3 Select the relationships in the function dependency set F. The top attribute is the function dependency on the left, and the bottom attribute is the function dependency on the right. If the selection is wrong, click "撤销". If the selection is complete, click "确定" to save the function dependencies. Click on "查看已有配置" to see the configured function dependencies. Click "下一步"

when the selection is complete.



6.2.4 In the configuration module page, click "查看" to see the data you have just configured.

6.3 Synchronizing Data Configuration

The BCNF and 3NF data Synchronization logic is the same. After completing the configuration of R and F, in the "配置" module, click on "同步". This button will synchronize the data configuration of the current module to another module. If you

are in the BCNF module, then the data will be synchronized to the 3NF module. And vice versa. (BCNF is used as an example here)

6.4 BCNF Calculation Module

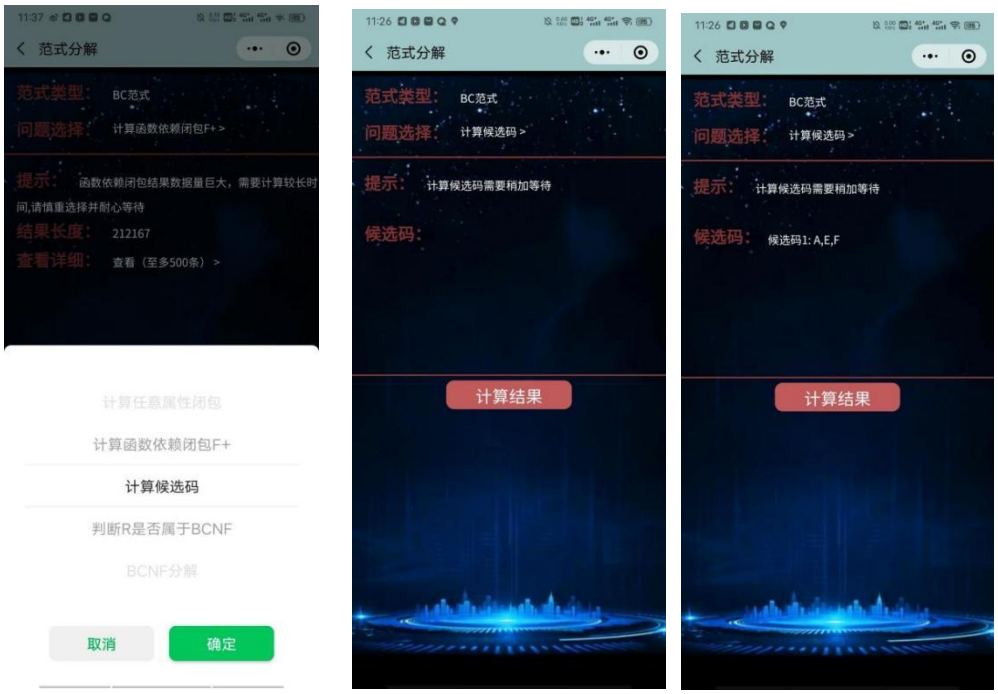
In the "Results" module, click "View Results" to jump to the page. Click on "Problem Selection" to select the problem you want to calculate.

The BCNF module provides 6 problems. The details are as follows

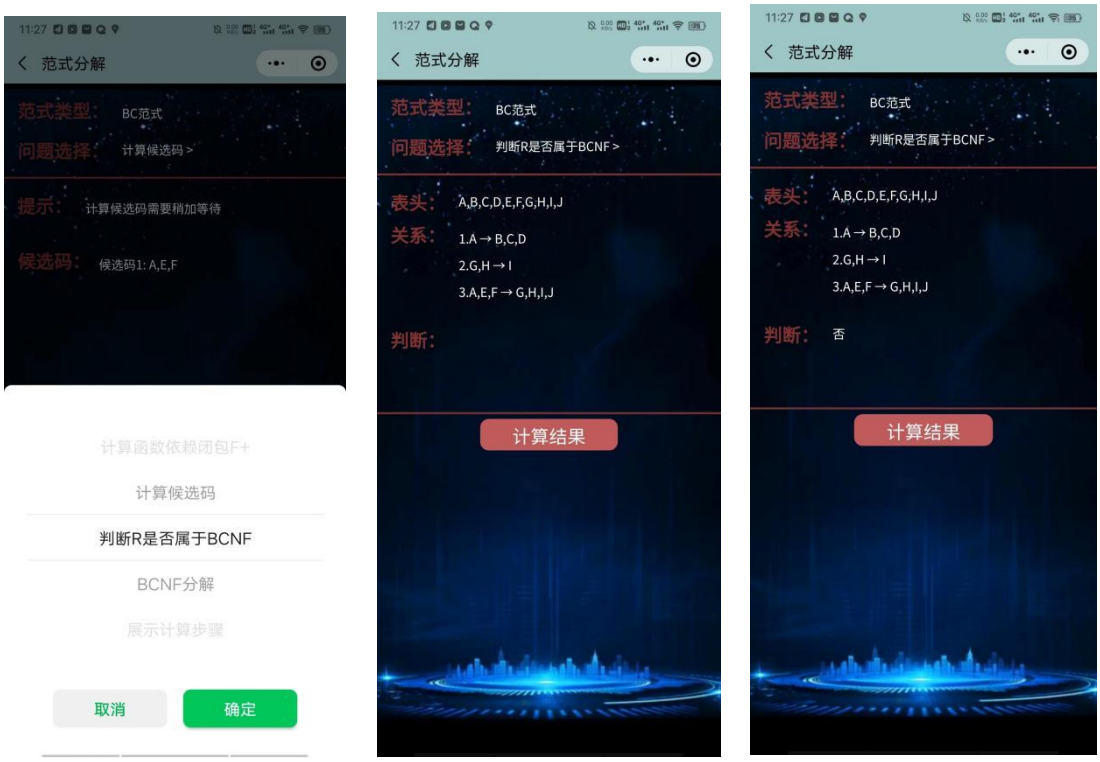
6.4.1 Calculate any attribute closure. Click "选择" to check the attributes. Click "计算结果" to view the results.

6.4.2 Calculate the function dependency closure F+. Click "计算结果" to start the calculation. The "结果长度" shows how many function dependencies are in F+, click "查看（至多 500 条）" to see up to 500 function dependencies.

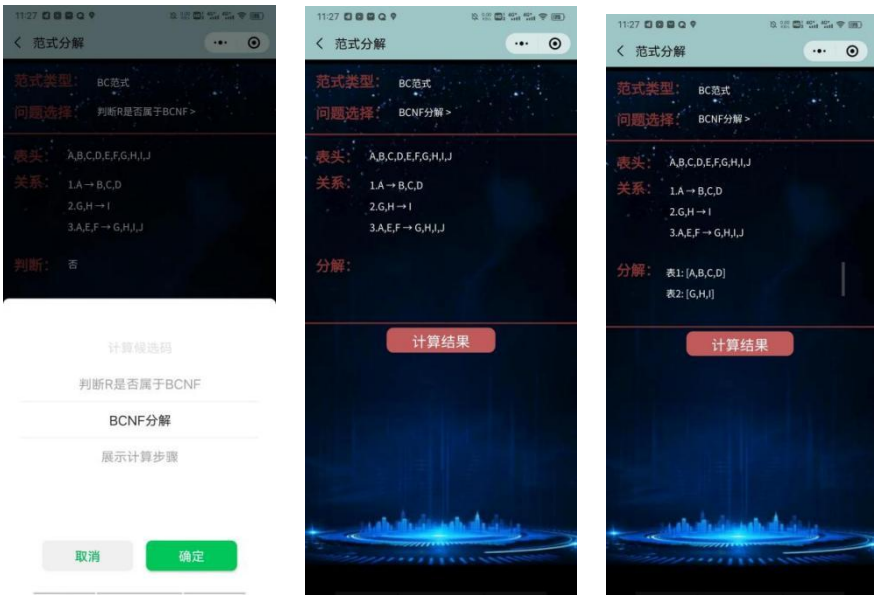
6.4.3 Calculating Candidate Codes. Click "计算结果" to see the results.



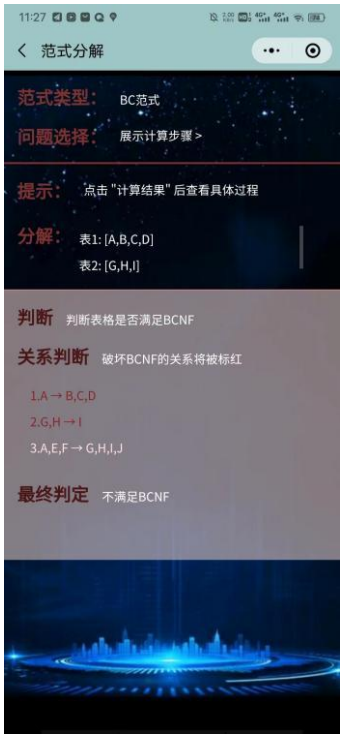
6.4.4 Determine whether R belongs to BCNF. Click "计算结果" to see the results.

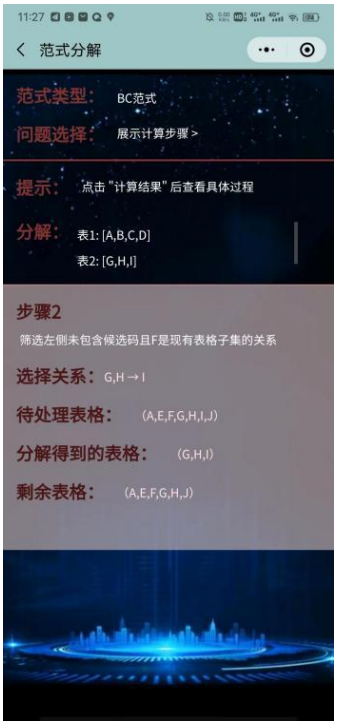


6.4.5 BCNF decomposition. Click "计算结果" to view the decomposition table. If the BCNF is satisfied, the decomposition is not necessary.



6.4.6 Show calculation steps. Click "计算结果" to slide through the results. The result page first shows the basic information of the decomposition, including R, F and candidate codes. Then, the function dependencies that break the BCNF are marked in red. Then, function dependencies are selected in turn for BCNF decomposition. Finally, the judgment of dependency preservation is provided.





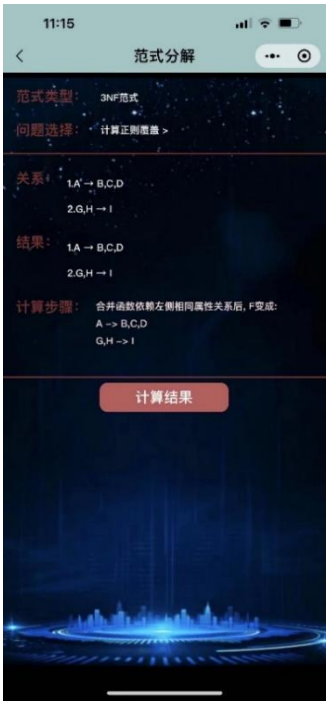
6.5 3NF Calculation Module

In the "结果" module, click "查看结果" to jump to the page. Click on "问题选择" to select the problem you want to calculate.

The 3NF module provides 6 problems. The details are as follows

6.5.1 Calculate arbitrary attribute closures. Same as BCNF, omitted here.

6.5.2 Compute a canonical cover set. Click "计算结果" to start the computation. You can see the result and the calculation steps.

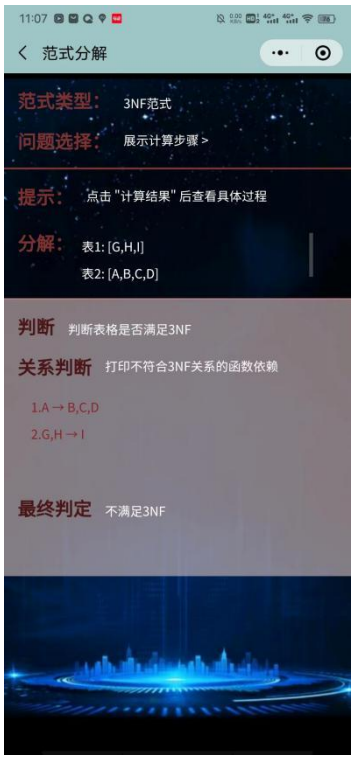


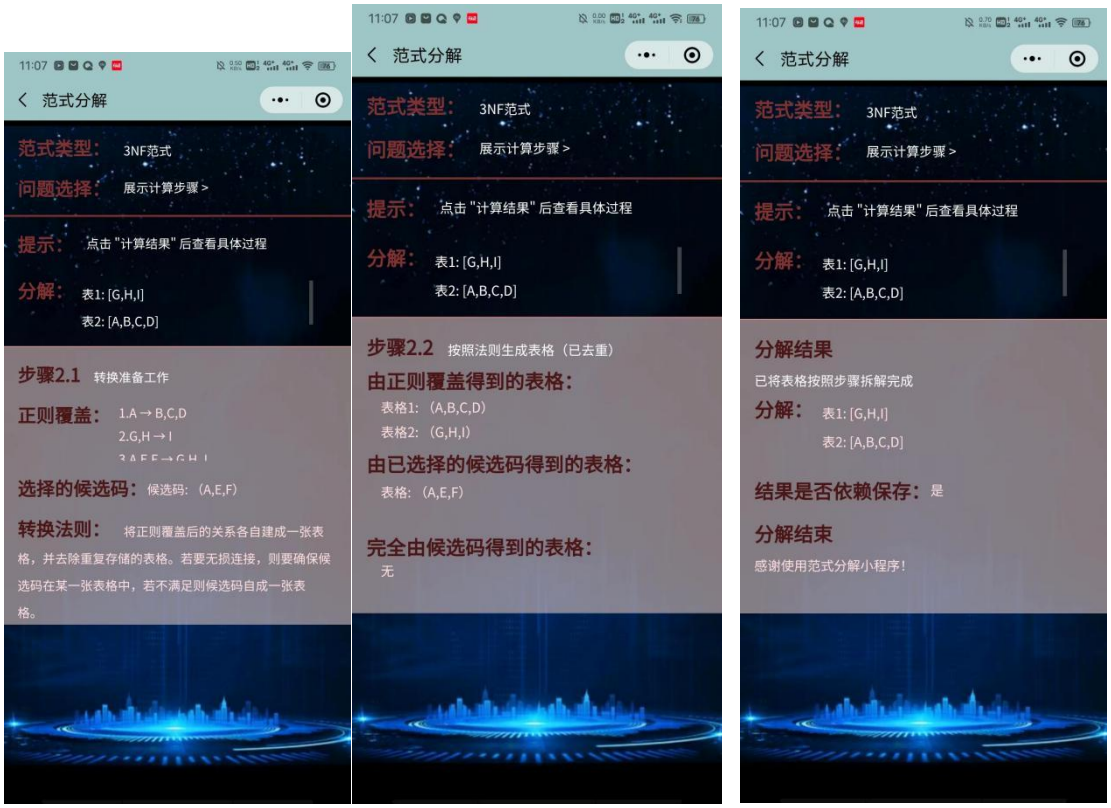
6.5.3 Calculating Candidate Codes. Same as BCNF, omitted here.

6.5.4 Determine whether R belongs to 3NF, and click "计算结果" to see the result.

6.5.5 Decompose 3NF. Click "计算结果" to view the decomposition table. Tables that satisfy 3NF are still decomposed for teaching purposes.

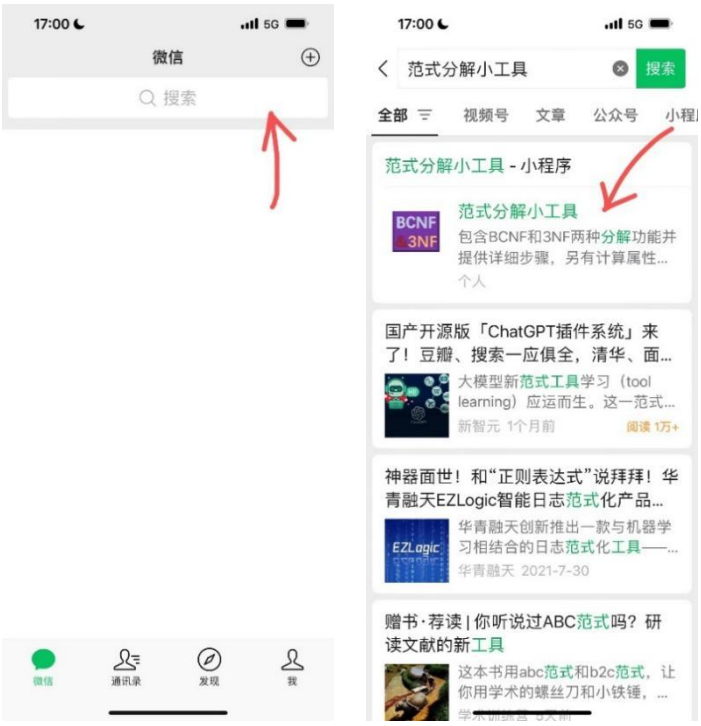
6.5.6 Show calculation steps. Click "计算结果" to slide through the results. The result page first shows the basic information of the decomposition, including R, F and candidate codes. Then, the function dependencies that break 3NF are marked in red. Next, the transformation law is applied to generate the table. It is also necessary to determine whether the table generated by using the candidate code is needed. Finally, the judgment of dependency preservation is provided.





7. Software tutorial

Search "范式分解小工具" in the search bar of WeChat to use it for free



8. Summary of the group discussion

The first discussion

Time: 2023-04-17

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. determined the group members as Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan, and the group leader is Yunyang Luo.

b. The topic of paradigm decomposition gadgets was decided.

Second Discussion

Time: 2023-05-04

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. determined the front-end as WeChat applet

b. determined the specific division of labor: Yunyang Luo is responsible for the base function code, Weijie Li is responsible for the BCNF decomposition code, Qinxiao Quan is responsible for the 3NF decomposition code, and Yuyang Li is responsible for the front-end.

c. determined what needed to be completed before the first interim report: the back-end code part was basically fully implemented.

The third discussion

Time: 2021-05-21

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. summarized the completion of tasks before the first interim report.

b. completed the first interim report.

c. confirmed what needs to be completed before the second interim report: update the

F+ algorithm to achieve the second out of F+; update the candidate code algorithm from finding the candidate code in F to finding the candidate code in R. Implement more front-end interfaces

Fourth discussion

Time: 2021-06-03

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. summarized the completion of tasks before the second interim report.

b. Finished the second interim report.

The fifth discussion

Time: 2021-06-10

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. assigned debug tasks to the front-end to improve the front-end functions

b. determined the division of labor for the final report, PPT, and using the introduction video.

Sixth discussion

Time: 2021-06-17

Members: Yunyang Luo, Yuyang Li, Weijie Li, Qinxiao Quan

Main content:

a. summarized the completion of all tasks.

b. determined the division of labor for the final presentation.

9. Personal Summary

Yunyang Luo

The smooth running of this project was due to the concerted efforts of the team members. I would like to thank the team members for their hard work. Yuyang Li was in charge of the front-end, and although the front-end required many complex modifications, he not only did not complain, but also completed the work efficiently and well. Weijie Li and Qinxiao Quan are responsible for BNCF and 3NF decomposition code, they can balance the efficiency and quality of writing code, and work seriously and responsibly. They not only fix the bugs they find quickly, but also have a comprehensive and detailed understanding of various documentation requirements and rules, which makes everyone feel at ease and solid. Qinxiao Quan also did an excellent job defending the project and getting our group recognized by our teachers. I was responsible for the construction of basic functions, such as calculating property closures, F+, candidate codes, etc. Through this project, I have deepened my understanding of databases, enhanced my programming skills, cooperation and coordination skills, and benefited a lot.

Yuyang Li

In this big project, I am responsible for the UI interface design of Mini Programs, interface code writing, front-end and back-end docking, and the In this big project, all the team members have paid a lot and gained a lot. In teamwork, I also learned how to connect different modules and realized I hope to have opportunity to exercise myself in the future.

Weijie Li

In this project, I was responsible for the BCNF decomposition part. Although no database was used in this project, the innovative use of WeChat applet instead of website as the front-end of the project allowed me to understand the full-stack

development process, front and back-end cooperation, communication and exchange. In the process of modifying and optimizing the BCNF algorithm time and again, I also became more proficient in the BCNF decomposition process and grasped the knowledge in the book more firmly.

Qinxiao Quan

In this project, I was responsible for the back-end part of 3NF. I further familiarized myself with the relevant syntax of 3NF and made algorithm innovations to reduce the algorithm time complexity according to the actual needs. Meanwhile, it was my first time to use WeChat applet as a carrier for software development, and I also used JavaScript language to implement the back-end of the applet for the first time. In this process, I understood the development process of the applet and participated in the front and back-end connection part. At the same time, this cooperation was also a pleasant one, our team actively communicated and exchanged ideas. At the end, I also acted as a tester to check the boundary special cases of the software to ensure the robustness of the applet, and in this process, I understood the kernel of the algorithm more deeply. All in all, it was a very rewarding project for me.

10. Project Schedule

Theme	Date	Plan to achieve	Practical achivement
Define the topic	2023/4/16-2023/4/28	1.define the topic; 2.learn BCNF,3NF,JS in advanced; 3.use Github to share codes.	1.define the topic -- NormalFormTool 2.learn BCNF,3NF,JS in advanced; 3.use Github to share codes.
complete the basic software functions	2023/4/29-2023/5/18	1.complete the back-end algorithm; 2.complete the front-end prototype.	1.complete the back-end algorithm; ·attribute closure ·canonical cover set ·candidate code ·F+ ·BCNF decomposition ·3NF decompisition 2.complete the prototype of WeChat Mini program
optimization	2023/5/19-2023/6/04	1.optimize F+ algorithm; 2.optimize the computation of candidate code; 3.optimize the computation of BCNF decomposition; 4.complete the front-end.	1.F+ algorithm is much faster; 2.optimize the computation of candidate code; 3.optimize the computation of BCNF decomposition; 4.complete the front-end.
debug	2023/6/4-2023/6/18	front-end and back-end debug.	finished.
project closing	2023/6/19-2023/6/30	1.write report; 2.design ppt; 3.prepare for academic defense	finished.