Applied Science and Innovative Research
ISSN 2474-4972 (Print) ISSN 2474-4980 (Online)
Vol. 8, No. 1, 2024
www.scholink.org/ojs/index.php/asir

*Original Paper*

# Exploration of Problems and Key Points in Database Design in Software Development

Yitian Chen[1], Yiran Li[1], Lihui Mao[1], Jiaxu Huo[1], Haolin Ning[1]

[1] University of Science and Technology Liaoning, School of Applied Technology, Anshan, Liaoning, China

Received: December 16, 2023 Accepted: January 25, 2024 Online Published: February 2, 2024

doi:10.22158/asir.v8n1p78 URL: http://doi.org/10.22158/asir.v8n1p78

***Abstract***

*Starting from the necessity and principles of database design, this article explores the optimization issues. Firstly, analyze the necessity of database design, elaborating on effective management, maintainability, resource utilization, and running speed; Then, a series of issues in database management were discussed, such as user management, data object design specifications, and overall design ideas; Finally, the optimization issues such as normalization rules, inter table redundancy handling, query optimization, indexing, and transactions were elaborated in detail. In the software development lifecycle, database design is indispensable. Its role is not only to ensure the safety and reliability of data, but also to ensure the overall stability and speed of the system. Strengthening the rationality and optimization of design is the key to improving software quality.*

***Keywords***

*Software development, Database design principles, Database optimization*

## 1. Introduction

In the software development process, database design belongs to the backend category, mainly analyzing and designing the storage structure and algorithms of data, in order to achieve the rational use of data in the database (Xu, 2023). Here, the database not only carries the function of data storage, but also ensures the safe and stable operation of the entire software system while meeting user needs. Therefore, the quality of database design has an undeniable impact on software systems. When designing, principles and norms must be reflected. While implementing basic functions, code volume should be controlled, code readability and maintainability should be enhanced, and code computation speed should be improved as much as possible to ensure that the system has sufficient applicability and practicality. This article focuses on the necessity and principles of database design, studies efficient

data storage methods, and lays the foundation for a reasonable connection between the front-end and back-end.

## 2. Necessity of Database Design

### *2.1 Effective Management of Databases*

The application of database technology lies in the effective management of resources and the rational utilization of data. The access to data is reflected in how it is stored and read. Good access methods are beneficial for software development, while bad access methods not only hinder the development process but also cause incalculable losses (Zhang & Wang, 2023). Currently, most of the popular data is structured. For this type of data, database technology can use database management systems to effectively analyze and process the data, such as adding, modifying, deleting, querying, etc. At the same time, the management system can also analyze the data. Therefore, database technology effectively solves the problems in managing a large amount of data, and can also facilitate the separation of front-end software development and back-end data. This allows for separate design and development, greatly reducing software development costs and improving software development efficiency. In terms of data migration, database technology also solves compatibility issues. Data can be conditionally migrated to other systems, as long as appropriate adjustments are made to the development software (Zhang, 2022). At the same time, there are also good solutions in terms of security. Through encryption control and other security measures, database security can be effectively guaranteed to prevent malicious theft and destruction.

### *2.2 Maintainability and Resource Utilization Issues*

The proportion of database design in software development is increasing, so it is necessary to examine the design process from different perspectives, such as the actual purpose of system development, the architecture of design, the professional ability of developers, etc. These factors can affect the quality of design. Different database systems should be designed for different practical purposes, and the design of the database should also be corresponding. The architecture construction, whether in size or design, should be suitable for the corresponding industry requirements. The bias of designers is also an issue that cannot be ignored. Backend design is not dependent on front-end development, but should be designed separately. If security vulnerabilities occur, the consequences will be unbearable for the system, not only during the development phase, but also leading to system crashes during operation (Zhang, Yang, & Li, 2022). The rationality of the data storage structure also needs to be considered, otherwise the maintenance costs in the later stage will increase and the resource utilization rate will decrease. Therefore, database design must be elevated to a higher level, and designers need to recognize this to avoid causing significant problems.

### *2.3 System Operating Speed*

The quality of database design not only affects development speed, but also affects the overall running speed of the system. On the premise of ensuring stable operation of the system, on the one hand, it is

necessary to meet the basic operating coefficient of the system, and on the other hand, it must also have a certain degree of scalability. A good database design can quickly obtain corresponding results and return them to the front-end when receiving messages sent by the front-end, and the retrieved information can meet the user's needs to the greatest extent, and complete the updating and deletion of data resources themselves, thereby achieving system optimization (Qi, Liu, Li, Liu, Ren, & Zhu, 2023).

## 3. Principles of Database Design

### *3.1 Preparation for Database Design*

Database design plays a crucial role in software development and directly affects the overall performance of the system. Sufficient design time should be reserved for necessary and detailed preparation (Li & Song, 2023). Database design plays a crucial role in software development and directly affects the overall performance of the system. Sufficient design time should be reserved for necessary and detailed preparation. Firstly, in the preparation stage of database design, sufficient communication should be conducted with the client to understand their specific requirements and strive to be as detailed as possible. This is the fundamental work of software development; Then, comprehensively consider the details of software development, maximize the effectiveness of the application, analyze from both the perspectives of designers and users, communicate with users in a timely manner during design, emphasize the scalability and flexibility of the design, revise the design content at any time, and decompose the entire database into multiple modules, confirm with customers one by one, and plan enough time to complete the design.

### *3.2 System Efficiency and User Needs*

The efficiency issue of database systems has always been a challenge in database design. Developers need to find a balance point, coordinate data storage while meeting functional requirements, and accelerate data computation. In the process of database design, accurate calculations should be made for some basic tables that need to be searched, and a reasonable storage method should be designed. If the amount of data stored in the basic tables is too large or too small, it needs to be controlled, otherwise redundancy problems will occur, which will affect the overall performance of the database. Tables with excessive data volume need to be conditionally split, which is beneficial for subsequent searches and can reduce some redundancy; For tables with small amounts of data, some field contents may become empty, and improper handling can result in a significant waste of spatial resources. At this point, merging tables can be used to reduce vacancy. If there are complex correlations between multiple tables in the database, the physical correlation can be appropriately changed to a logical correlation, which can be explained in relevant documents to reduce the complexity between tables (Fang, Xie, & Wang, 2023).

The relationship between user needs and system efficiency is both contradictory and unified, because user needs are sometimes very general, requiring designers to communicate repeatedly, and there are many demand points. Meeting their needs while ensuring system efficiency is an important issue that

designers need to consider. Based on the characteristics of different users and combined with actual development opinions, user needs can be divided into two aspects: system performance and security. System performance varies among different industries and users with different needs. Some require high performance, while others only require basic functions. Designers should develop systems that meet the needs of users based on their actual situation; Security, whether it is physical integrity or reference integrity, is a requirement for information security by users. As the amount of data processed by the system increases, users will have higher requirements for data security during transmission and storage. Designers need to have a detailed understanding of these requirements and make necessary designs.

*3.3 Standardized Data Table*

A database object includes tables, views, queries, stored procedures, triggers, indexes, etc. Tables are the basic elements of the database, and the system requires a large amount of data support, which is stored in different tables. The data stored in these tables often have connections, which forms correlations. Designing reasonable inter table associations is the key to improving system performance, which not only effectively enhances the linkage between data, but also enhances the overall architecture stability of the system. The design of a table structure is particularly important as it serves as the foundation for undertaking other data objects. The design of table structure needs to follow two principles. Firstly, the table structure should have a certain degree of recognizability, that is, it can be understood from the table, the table can describe itself, and its purpose can be roughly understood without special explanation or annotation; Secondly, while meeting the current system requirements, it meets the scalability requirements to meet the subsequent maintenance and performance expansion of the system. A reasonable table name is important for identifying tables and is also applicable to other data objects. The use of standardized naming rules can not only effectively identify data objects, but also facilitate future maintenance work. For example, using a combination of letters, numbers, and underscores, separating letters with underscores, using nouns or verb object phrases, or adding some descriptive information, simple and clear, easy to remember and understand. For some table structures that require storing a large amount of data, under the premise of normalization, it is necessary to strengthen the correlation between tables, which can be established through logical or physical methods. Both methods have their own advantages and disadvantages, with strong standardization of physical correlation but poor development flexibility; The difficulty of logical association development is reduced, but it is prone to dirty data. The two need to be balanced and designed as clearly as possible to provide convenient conditions for subsequent design work.

*3.4 Overall Design Concept*

In the software development lifecycle, maintenance in the later stages cannot be ignored, and its overall cost is relatively large. As an important part of it, database design and development also need to follow this principle. When designing a database, not only should user functional requirements be considered, but also maintenance and updates in the later stage should be taken into account. Although designers

consider scalability, any software product cannot avoid the problem of insufficient consideration. With the extension of system launch time, and due to the unreasonable design of designers, there may also be insufficient space, and user needs may also change. Therefore, in order to enhance the scalability and fault tolerance of the system, designers need to leave sufficient expansion space for some tables and fields when designing the database, in order to facilitate subsequent database manipulation. This is very beneficial for subsequent database maintenance operations and provides a foundation for further optimization of the database.

## 4. Database Optimization Design

### *4.1 Third Normal Form*

For general information management systems, data tables should conform to the Third Normal Form (3NF), which means that all non primary attributes do not have a transitive dependency, meaning that non primary attributes only rely on the primary key. Its advantages are: (1) greatly reducing redundancy and saving a lot of storage space; (2) Ensured data integrity constraints, namely entity integrity and reference integrity. In this way, data migration and updates become easier; (3) The accuracy of the query is improved, and there will be no Cartesian product when performing join queries, which means there will be no data duplication or missing data; (4) Optimized for transaction processing and physical structure changes, able to meet the changing needs of users.

### *4.2 Redundancy Handling between Tables*

In order to improve system development efficiency and reduce I/O consumption, it is necessary to tolerate some data redundancy. In stored procedures, if multiple tables are frequently accessed and there is a master-slave relationship between these tables, it can be considered to add data from sub tables in the master table or temporary table for data redundancy operations. This can effectively reduce I/O operations and improve efficiency. However, this behavior violates paradigm norms and correspondingly increases data maintenance difficulties. If the original table is updated, a trigger or triggering mechanism must be used to update the redundant table data.

### *4.3 Query Optimization*

When accessing data tables, the overall requirement is to minimize sorting, joining, and subquery operations as much as possible. The specific points are as follows: (1) Try to minimize sorting during the query process. When sorting a large amount of data, the data should be placed in a temporary table and sorted using simple data types, such as integers or short characters; (2) Try to avoid nested queries within tables as much as possible; (3) Avoid using complex calculation expressions or truncated strings for concatenation operations in where conditions; (4) In the where condition, reduce the use of "or" and use more "and"; (5) When establishing multi table connections and complex sub queries, temporary tables should be used as much as possible to store intermediate data, in order to reduce I/O, but this will increase space loss.

*4.4 Index*

Using indexes can greatly improve system performance. Typically, it is necessary to find columns commonly used for querying or sorting in the data table, establish indexes on them, and set indexed fields in the where conditional expression to avoid time loss caused by large-scale scanning of data. Directly accessing indexed fields has better performance. Meanwhile, the primary key itself is an index and a unique index, ensuring the uniqueness of the data. The index columns should also be updated regularly. Indexing has many benefits, but there are also drawbacks, such as the need for space and time cost to establish an index. Moderate maintenance is required during addition, deletion, and modification operations. There are generally two types of indexes: clustered indexes and non clustered indexes. A table can have and can only have one clustered index, but there can be multiple non clustered indexes. The corresponding clustered index is faster in search speed than non clustered indexes.

*4.5 Transactions and Locks*

A transaction is an operation of a unit, either all of which is done or none of which is done, with the main purpose of maintaining data consistency. Locking is an important part of concurrency control. When a transaction is established to operate on a data object, a request needs to be issued to lock the object. The transaction then monopolizes the object until the operation is completed, and other transactions cannot update the object. The system utilizes this feature to achieve data integrity. However, there are also some issues with transactions, such as deadlocks. Due to its exclusivity, if a long transaction is set to monopolize an object for a long time, it will cause deadlocks. Therefore, when designing transactions, efforts should be made to avoid long transactions, reduce the number of exclusive data objects in the transaction, and avoid long-term interaction with users; For the same data object, batch data processing should be avoided to reduce time consumption. In addition, the use of locks may also cause a phenomenon of "fake death". In transactions, if a data object is monopolized, other users can only wait for requests. When there are many requesting users, system performance will decrease, resulting in a large number of waiting users "fake death" problems. The solution can be as follows: (1) Downgrade the lock from page level lock to row level lock, reducing the exclusive duration; (2) Using clustered indexes to distribute data to different pages is equivalent to creating multiple redundant tables for a table, which is also a form of demotion; (3) Try to minimize the length of transactions and minimize interaction transmission.

*4.6 Physical Storage of Database Objects*

The physical storage strategy for database objects is uniform allocation, and I/O should avoid being too large or too small. The specific rules are as follows: (1) Distributed storage, which disperses data into multiple storage devices, allowing users to jump between different storage devices when accessing, avoiding I/O bottlenecks; (2) Separate the system database from the user database and place temporary tables on idle storage; (3) Placing log files on independent storage not only improves overall system security but also reduces unnecessary I/O losses; (4) For high-frequency accessed data tables, they

should be stored separately, especially in cases of multi table connections, and even some fields can be stored separately to effectively avoid I/O congestion.

## 5. Conclusion

Only by ensuring good performance of the database can effective data support be provided for front-end development. When completing database design, it is necessary to fully consider the connection between the front-end and back-end, start from applicability, and balance scalability and security to ensure the overall stability of the system.

## Fund Project

This paper is supported by the Innovation and Entrepreneurship Training Program of University of Science and Technology Liaoning.

## References

Fang, M., Xie, J., & Wang, H. M. (2023). Research on Multi directional and Progressive Database Experimental Teaching. *Computer Education*, *2023*(04), 140-145.

Li, M. X., & Song, Y. (2023). Design and implementation of a database based data mining experimental platform. *Technological innovation and application*, *13*(24), 46-50+55.

Qi, J., Liu, Z. Q., Li, X. L., Liu, P., Ren, Y. F., & Zhu, X. L. (2023). Basic Teaching Design of Database Application Based on Data Analysis of Blended Teaching Process. *Software Guide*, *22*(11), 199-204.

Xu, L. (2023). Research on Database Design Issues in Software Development. *Wireless Internet Technology*, *20*(15), 87-89.

Zhang, H. Y. (2022). Research on the Reform of Engineering Teaching for Applied Undergraduate Database. *Jiangsu Science and Technology Information*, *39*(17), 53-56.

Zhang, R., & Wang, H. (2023). Project based teaching design for database courses. *Electronic technology*, *52*(11), 118-120.

Zhang, R., Yang, S. L., & Li, B. W. (2022). Key Technology of Application System Database Development in the "Internet plus" Era. *Information and Computer (Theory Edition)*, *34*(11), 164-166.