

Design of Abertawe Theater Database System

by Yu Xin

FILE	24330_YU_XIN_DESIGN_OF_ABERTAWE_THEATER_DATABASE_SYSTEM_315 194_1076544360.DOCX (105.07K)		
TIME SUBMITTED	15-JAN-2021 11:38AM (UTC+0000)	WORD COUNT	2415
SUBMISSION ID	142228973	CHARACTER COUNT	13143

Design Of Abertawe Theater Database System

Abstract

This report designs the database system based on the information provided by abertware theater. The design is divided into three parts: requirement analysis, function design and database system design.

In the demand analysis part, according to the theater information given by the title, it analyzes the database system requirements of the theater from various aspects to meet the needs of actors' group performance, customer booking and other functions. In the functional design part, I design the function of the database system to meet the needs of the theater according to the requirements analysis results of the first part. In the database design part, I designed the database in detail according to the function analysis of the second part.



1. Demand analysis

According to the given case information, the theater should not only manage its own venues effectively, which is mainly reflected in the need for the theater to perform for various groups. In this step, I think the administrator's authority can be set. The group that needs to perform can only inquire whether the venue is free through the theater management personnel, and make reservation through the administrator. At the same time, the theater should reserve seats for the external audience. The audience can query the current performance and the current performance as idle information through another database. At the same time, the audience can modify and cancel the canceled information under their own account ID. Administrators can do the same operation to modify and delete many false information and problem orders.

The database management system is divided into the following functional modules from the functional structure.

- 1) Actor reservation module: This module is for theater managers, and can realize actor reservation. Specific can be achieved: site query, site reservation, cancel reservation, reservation information modification.
- 2) Audience seat reservation module: This module faces all the audience, and can realize the audience's seat reservation. Specifically, it can realize: performance information query, seat status query, seat reservation, cancellation of reservation and reservation information modification.

According to the above analysis, the use case diagrams of the two modules are given as follows:

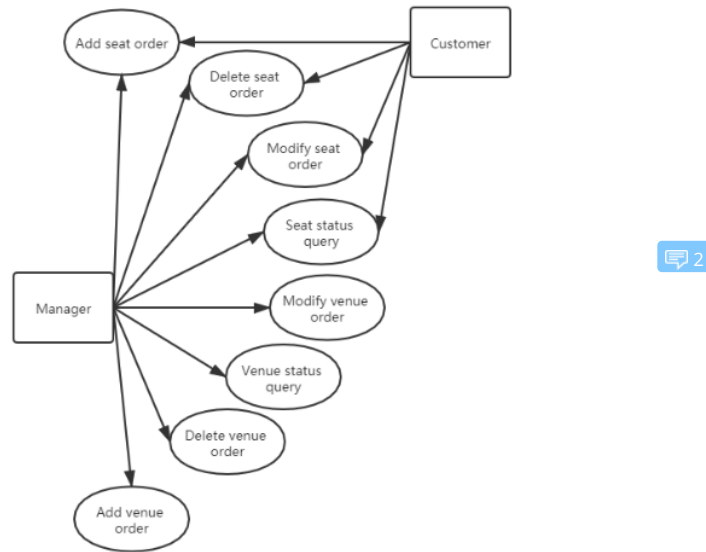


fig1. use case diagram

In addition to the above necessary functional requirements of the theater management system, the system should also include non functional requirements. For example, security requirements, performance requirements, interface requirements and system deployment requirements. When considering the performance requirements of the system, we mainly consider the response time of the system to ensure that the response time of the system is within the acceptable range of users. When considering the deployment requirements of the system, in order to facilitate the use of administrators, users only need to have a browser to log in to the system and carry out corresponding operations. We can use the B / S architecture mode. However, the B / S architecture is beyond the scope of this paper, which will not be discussed in this paper.

2. Function Design

The management system adopts B / S mode. This can not only improve the response time of the enterprise material management system, but also ensure the security of the system. Secondly, the B / S architecture mode can reduce the pressure of the server. The architecture of theater management system is shown in the figure. The front end of the system is the presentation layer, including system login interface, customer reservation interface, administrator theater management interface and administrator seat management interface. The business layer mainly realizes the function design of each module, including the addition, deletion, modification and query of each module. The bottom layer of the system is the database layer. The user submits the command in the display layer, and the command in the display layer is passed to the business layer. The business layer realizes the functions that the user needs by interacting with the database, and then displays the results to the interface layer.

The overall framework of theater management system is as follows:

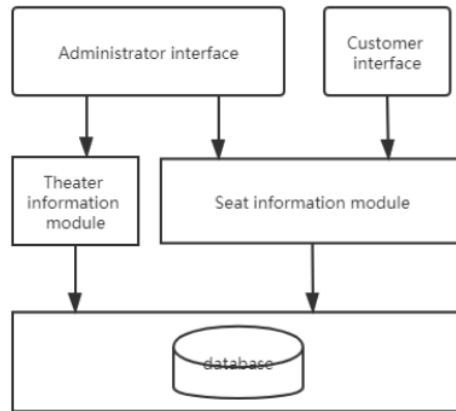


fig2. System structure

2.1 Function Detailed Design

2.1.1 function design of theater information module

Theater information module mainly completes the management of theater rehearsal information.

- 1) New theater rehearsal information: the administrator can add new theater rehearsal information after logging in to the system. Each information includes performance serial number, performance time, performance place, performance group name, performance group category, etc.
- 2) Query theater rehearsal information: the administrator can obtain the corresponding information by querying the three theaters, the query time and the name of the performance group, so as to arrange the reservation of the new performance group.
- 3) Delete theater rehearsal information: after logging in, the administrator can delete the corresponding performances according to the conditions.
- 4) Modify theater rehearsal information: the administrator modifies the existing performance information, such as time adjustment and location adjustment.

2.1.2 Functional Design Of Seat Information Module

Seat information is mainly used to manage the seats of each theater, so as to facilitate the audience to book tickets.

- 1) Seat query: the audience can obtain the remaining seats in a specified period of time and theater through query
- 2) Seat reservation: the audience can reserve the existing empty seats for the specified session.
- 3) Cancellation of seats: the audience can cancel the seats that have been reserved for the specified session.

4) Seat change: the audience can change the seats already reserved, only for the seats at the same price in the current session.

The above operations can also be performed by theater administrators.

3.Database Design

3.1 SQL and NoSQL

NoSQL database does not follow all the rules of relational database. Relational database arranges data in tables that can be connected or associated through common fields, separates it from application and uses SQL query. In other words, relational databases put data into tables, and SQL creates an interface to interact with them. NoSQL is not built on the concept of table, and does not use SQL to manipulate or analyze data (some NoSQL databases have SQL like query languages). NoSQL includes structured data, unstructured data and polymorphic data. SQL generally deals with structured data. In general, NoSQL differs from SQL in the following ways:

- 1) Type
- 2) Language
- 3) Scalability
- 4) Follow attributes
- 5) Community

NoSQL database has many advantages. The main advantages are high scalability and high availability. The high scalability of NoSQL database is mainly manifested in its horizontal expansion by using fragmentation. Partitioning data and placing it on multiple computers in a way that preserves the order of data is fragmented. Vertical scaling means adding more resources to existing computers, while horizontal scaling means adding more computers to process data. First of all, vertical scaling is easier to achieve than horizontal scaling. The automatic replication function in NoSQL database makes it highly available, because in case of any failure, the data will copy itself to the previous consistent state. NoSQL also has some disadvantages, such as weak transaction management and inconsistent standards.^{[2][3][4]}

Through the analysis of the above advantages and disadvantages, the scenarios of NoSQL mainly include:

- (1) When large amounts of data need to be stored and retrieved.
- (2) The relationship between the stored data is not that important.
- (3) Data changes over time and is not structured.
- (4) There is no need to support constraints and joins at the database level
- (5) With the continuous growth of data, the database needs to be expanded regularly to process data

To sum up, in view of the theater information provided, I think the traditional SQL database is adequate for this demand. The main reasons are as follows: the high scalability of NoSQL

database is not shown in this case. When the amount of data is large, the speed of NoSQL data with graph structure is slower than that of structured SQL database.

3.2 Database Conceptual Design

System requirements analysis is the most important part of the system design and implementation. Before the system design, we need to understand the specific needs of users, and then develop after fully understanding the needs of users. Only in this way can we develop a financial management system that meets the expectations of users.^[1]

This chapter mainly studies the process of theater customer-oriented and actor oriented, briefly describes the business processing, mainly analyzes the various functional modules of the theater, and establishes the model using the idea of object-oriented. In this case, each function module contains several obvious entities. For the venue reservation module, there are obviously two entities: Theater and performance team. For each theater and performance team, a performance entity will be generated. For seat reservation module, there are three entities: customer and seat information and performance entity composed of venue reservation module. According to the above requirement analysis and function design, the following entity information is given for the above modules :

- 1) Site entity. The site entity includes site ID, site name and the number of seats in the site.
- 2) Seat entity. Seat entities include seat ID, seat location and seat category.
- 3) Customer entities. Customer entities include customer ID, customer name and customer telephone number.
- 4) Performance entity. Performance entity includes performance ID, performance location, performance time and performance personnel.

3.3 Database Logic Design

3.3.1 site information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
place_id	Yes	varchar	No	32	No
place_name	No	varchar	No	32	No
seat_num	No	int	No	650	No

3.3.2 seat information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
seat_id	Yes	varchar	No	32	No
seat_place	No	varchar	No	32	place_id
Seat_type	No	varchar	No	32	No

3.3.3 customer information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
customer_id	Yes	varchar	No	32	No
customer_name	No	varchar	No	32	No
Customer_tele	No	varchar	No	32	No

3.3.4 actor information sheet

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
------	-------------------	-----------	-------------	--------	-------------------

actor_id	Yes	varchar	No	32	No
actor_name	No	varchar	No	32	No
actor_type	No	varchar	No	32	No

3.3.5 performance information sheet

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
show_id	Yes	varchar	No	32	No
show_actor_id	No	varchar	No	32	actor_id
show_place_id	No	varchar	No	32	place_id
show_date	No	varchar	No	32	No

3.3.6 seat reservation information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
book_id	Yes	varchar	No	32	No
book_customer_id	No	varchar	No	32	customer_id
book_seat_id	No	varchar	No	32	seat_id
book_show_id	No	varchar	No	32	show_id

As you can see from the information table, show is the foreign key in the information table_ Show in ID and session information table_ Date is set to be non null. Through the setting of SQL constraints, two problems in the problem are easily solved, that is, the lack of time information in the session information and the reservation of non-existent sessions in the seat reservation.

3.4 Index Design

Index is a data structure to help SQL get data efficiently. Indexes are usually added to important columns that are frequently queried in a database. In this case, the index can also help us find quickly. The index keys to be added are: performance time, performance group name and audience name in the performance information table. When a group makes a reservation for a theater, it often makes a query according to its own time and the free time of the theater to determine the final performance time. For the same performance group, administrators can query their scheduled performance information to perform. For the audience, the audience can query their scheduled performance information and seat information through the external interface. The SQL statement for creating an index is as follows:

```
CREATE INDEX Show_Date
ON Show (show_date)
```

```
CREATE INDEX Customer_name
ON Customer (customer_name)
```

```
CREATE INDEX Actor_name
ON Actor (actor_name)
```

3.5 Distributed Database Design

Distributed database refers to the database system in which data are physically distributed and logically centralized. There are three main characteristics of distributed database: one is transparency. For users, distributed database is equivalent to a stand-alone database, shielding the bottom multi node, data physical dispersion, copy consistency and other details; second, data redundancy. Distributed database realizes system reliability and availability through multiple copies. When the data in a node is not available, other data copies can continue to ensure business continuity, and can also calculate the data nearby to reduce network consumption and improve performance. Third, it is easy to expand. Distributed database can improve the overall processing capacity through horizontal expansion. Data can be dynamically distributed to new nodes to eliminate data skew.

The core technologies of distributed database include data replication, that is, the data synchronization mechanism between different copies; data partition, that is, how to distribute massive data to different nodes; distributed transaction, that is, to solve the atomic, consistency, isolation and persistence challenges faced by multiple nodes.

In response to the requirements of the theater, I also give a set of distributed database design. After multi-faceted thinking, I think the horizontal segmentation scheme is the most suitable for the current system. The horizontal segmentation method is based on the segmentation of show in the field information table. Place value to divide the data into three different sites. The three sites store the corresponding information of each site. The split statement is as follows:

P1: σ show_place_id = ' 1' (Performance) // The main auditorium
P2: σ show_place_id = ' 2' (Performance) // Dylan 1
P3: σ show_place_id = ' 3' (Performance) // Burton 2

Conclusion

This report designs the database system based on the information provided by abertware theater. The main content of the report is the design of the database. In view of the reality of abertware theater, I designed six tables to store the corresponding information. For the existing problems given by theater, I also standardize the database by adding constraints. Finally, I choose the most suitable distributed storage solution according to the requirements.

Reference

- [1] Grace. Design and implementation of material management system for Inner Mongolia National Art Theater [D]. Jilin University, 2016
- [2] Chen Qifan. Research on distributed storage technology of public security monitoring data based on NoSQL [D]
- [3] Lei Yuhui, Zhong Wen, he Qing, Tang manling, Liu Chaoying. Literature review on NoSQL database research [J]. Electronic world, 2017 (04): 11 -12
- [4] CHEN Zhong-ju. Research and Application of NoSQL Database [J]. Computer Programming Skills and Maintenance, 2020, No.423(09):83-85.

Design of Abertawe Theater Database System

GRADEMARK REPORT

FINAL GRADE

44/100

GENERAL COMMENTS

Instructor

Some of your report content was irrelevant for this assignment you begin with an overview of the functional requirement and a use case diagram you are required to produce and ERD. You have presented some information on SQL and NoSQL but you need to improve your referencing techniques. Basic SQL design for your solution and not enough testing. No consternation of testing transactions.

PAGE 1



Comment 1

You should write in 3rd person try to avoid I eg I designed

PAGE 2



Comment 2

Ok But for this assignment an ERD is more relevant

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7
