
The Database Design of Abertawe Art Center

Xin Yu

2006647

Figure Catalogue	2
Table Catalogue.....	2
Abstract.....	3
1. Introduction.....	3
2. Design of the database	3
2.1 The Design of ER diagram	3
2.2 The design of attributes.....	5
2.3 Constraints in database.....	6
2.4 Index in database	6
2.5 Transaction in Database.....	7
3. Database of SQL and NoSQL.....	7
3.1 The advantages of NoSQL.....	8
4. Distributed Database Design.....	9
Conclusion	10
Reference	10

Figure Catalogue

Figure 1. The E-R diagram of the performance	4
Figure 2. The E-R diagram of the performance	4

Table Catalogue

Table 1. Performance Venue information table.....	5
Table 2. Seat information table.....	5
Table 3. Customer information table.....	5
Table 4. Performance group information sheet.....	5
Table 5. Performance information sheet	6
Table 6. Seat booking information table.....	6

Abstract

This report can be divided into three parts. First of all, according to the information provided by Abertawe theater, the database system is designed to meet the needs of customers and theater staff for reasonable booking management and theater seat management. Secondly, SQL and NoSQL databases are compared. Finally, according to the information provided by Abertawe theater, the distributed database is designed.

1. Introduction

With the development and promotion of the network, the data volume also increases explosively. For individuals and businesses, it is particularly important for data storage and management. And the university management and storage of data can not be separated from a good database. Therefore, it is very important to design different levels of database for different problems to meet the daily needs of the group or enterprise.

According to the information given, Abertawe Arts Centre has two major needs: the reservation of actors for the theatre and the reservation of the audience to the seats. This report focuses on the design of the database for both requirements. First, I use Er graph to represent the logical relationship of the database system, and then I use SQL statements to build the whole database., I added indexes to the database to make the database retrieval more efficient. In the second part, this report discusses the difference between SQL and NoSQL databases. The third part, also for the requirements of Abertawe Arts Centre, the distributed database design. Distributed database can make good use of the distribution relation of data storage in space, which makes the database storage more efficient and cost lower.

2. Design of the database

Before designing the database, we need to analyze the requirements of Abertawe Art Center. Based on the existing information, we can draw the following requirements:

- (1) Abertawe Art Center has three locations for professional and amateur performance groups. Groups can book three venues and schedule their performances
- (2) Each theater has a different number of seats for the audience to choose.
- (3) Each audience can book seats for different performance.

2.1 The Design of ER diagram

Through the above analysis, we can think that the database mainly contains three entities and two functions. Requirements can also be expressed as relationships between entities. The entities here are: performance venue, performance group and seats. The requirement is mainly divided into venue reservation and seat reservation. According to the two functions summarized above, I adopt the bottom-up design structure and get the ER diagram of two parts, as shown in Figure 1 and Figure 2.

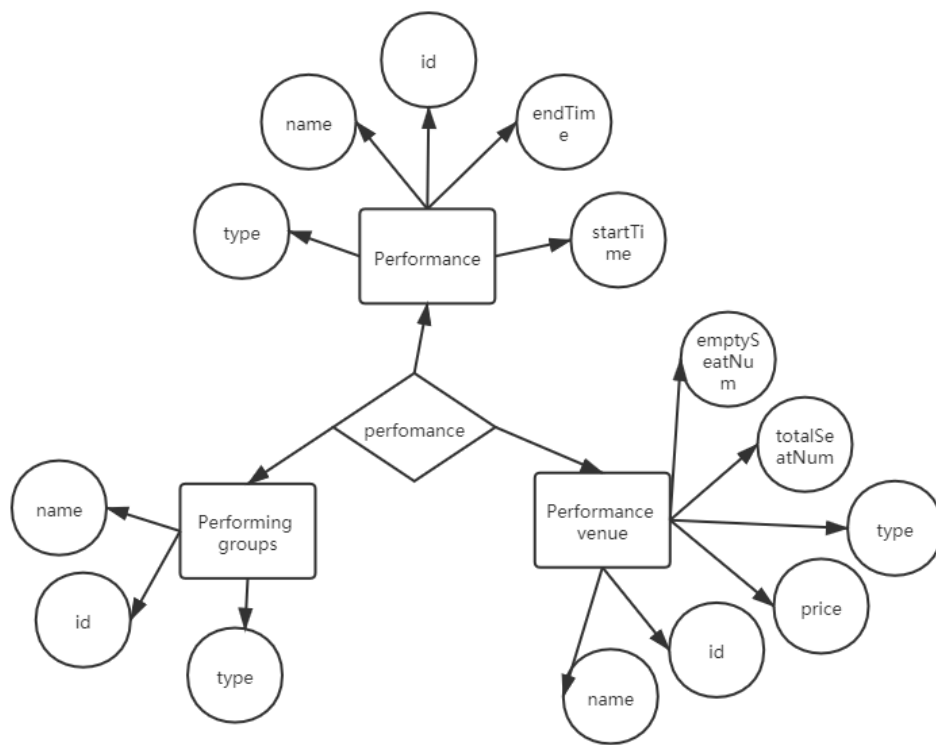


Figure 1. The E-R diagram of the performance

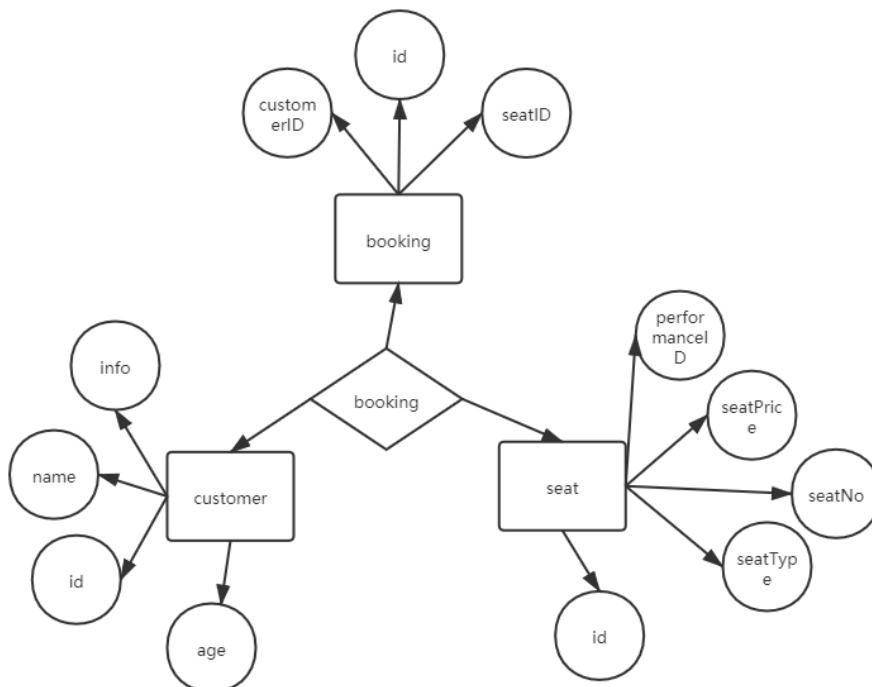


Figure 2. The E-R diagram of the performance

2.2 The design of attributes

According to the information in Figure 1 and Figure 2, we can see that this database design pattern conforms to the 3NF database pattern. According to the above information, it can be divided into the following six data tables:

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
Venue_id	Yes	varchar	No	32	No
name	No	varchar	No	32	No
Type	No	int	No	650	No
totalSeatNum	No	int	No	650	No
emptySeatNum	No	int	No	650	No
venuePrice	No	int	No	65535	No

Table 1. Performance Venue information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
id	Yes	varchar	No	32	No
performanceID	No	varchar	No	32	Yes
Seat_type	No	varchar	No	32	No
Seat_No	No	varchar	No	32	No
Seat_price	No	varchar	No	32	No

Table 2. Seat information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
customer_id	Yes	varchar	No	32	No
name	No	varchar	No	32	No
age	No	varchar	Yes	32	No
info	No	varchar	No	32	No

Table 3. Customer information table

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
id	Yes	varchar	No	32	No
name	No	varchar	No	32	No
type	No	varchar	No	32	No

Table 4. Performance group information sheet

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
show_id	Yes	varchar	No	32	No
name	No	varchar	No	32	No
type	No	varchar	No	32	No
companyID	No	varchar	No	32	Yes
venueID	No	varchar	No	32	Yes

startTime	No	date	No	32	No
endTime	No	date	No	32	No

Table 5. Performance information sheet

Name	Is it primary key	Data type	Is it empty	Length	Is it foreign key
book_id	Yes	varchar	No	32	No
customer_id	No	varchar	No	32	Yes
seat_id	No	varchar	No	32	Yes
performance_id	No	varchar	No	32	Yes

Table 6. Seat booking information table

2.3 Constraints in database

Constraints play an important role in relational databases. It means to constrain the data in the table to show the correctness and validity of the table. They are usually played by the primary key ID in the table. In the six tables in the database of this design, there are also constraints. In all six tables, each entity has its own ID, which is also the primary key in the table. In the performance table, companyID and venueID are foreign keys. They represent the ID of the performance group and the ID of the performance venue respectively. They represent which performance group will perform this performance and where they will perform. In the seat table, the performance ID is used as the foreign key to associate the performance with the information. In the more important reservation table, it contains a primary key and three foreign keys, and expresses which customer is watching which performance in which seat.

2.4 Index in database

After the database goes online, we may find that with the increase of performances and the number of audience watching performances, the query efficiency of the database may decrease with the increase of data volume. In order to improve the efficiency of database retrieval, index is essential. However, adding indexes regardless of the situation does not necessarily change the query efficiency of the database, but may reduce it. This is because the index itself will occupy a certain storage cost. Therefore, we need to consider which tables need to be indexed. In the performance group information table and audience information table, the information is relatively fixed, which needs to be added, deleted and modified frequently, and the query demand is relatively small, so it is not suitable to add index. For some information in other tables, I think we can add indexes to improve query efficiency.

First of all, for some frequently queried information in the table, such as price, it is helpful to establish an index. For the seatID and seatPrice in the information table, the index is helpful to quickly query the price of seats in different locations in the theater, which is often concerned by the audience. Secondly, for the booking table, we think we need to add indexes for both CustomerID, seatID and performanceID. Because it is obvious that when customers come to the theater to watch the performance, although they own their own information, for the theater staff, they need to check whether their personal information and seat information are the same on the spot, so as to be their access conditions. The index creation statement is as

follows:

```
CREATE NONCLUSTERED INDEX seatID on SeatInformation(seatID);
CREATE NONCLUSTERED INDEX seatPrice on SeatInformation(seatPrice);
CREATE NONCLUSTERED INDEX bookingInformation on Booking(customerID,seatID);
```

2.5 Transaction in Database

Transaction is a user-defined sequence of database operations. These operations are either all done or none done. It is an indivisible unit of work. For example, in a relational database, a transaction can be a single SQL statement, a group of SQL statements, or the whole program. When you want to modify two different tables in the database at the same time, if they are not a transaction, when the first table is modified, but the second table fails to be modified due to an exception, only the second table returns to the state before modification, and the first table has been modified. However, when we set them as a transaction, when the first table is modified, but the second table is not modified due to an exception, both the first table and the second table will return to the unmodified state. This is called transaction rollback. In our database design, all the records in the seat table have been registered, and the customer corresponds to the act in the booking table. Assuming that there are unregistered tickets, the same transaction can detect the unregistered seats and perform rollback. The sentence is as follows:

```
BEGIN TRANSACTION AddPerformance;
BEGIN TRY
INSERT INTO Performance(id, name, type, companyID, venueID, startTime, endTime)
VALUES (1,'You and Me',' modern drama ',1,1,'2021-5-22 5:23:00', '2021-5-22 6:23:00' );
COMMIT TRANSACTION AddPerformance;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION AddPerformance;
RAISERROR('Rolling back, an error occurred!', 16, 1);
END CATCH;
ROLLBACK;
```

3. Database of 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 public fields, separates it from the application, and uses SQL query. In other words, a relational database puts data into a table, and SQL creates an interface to interact with it. NoSQL is not built based on the concept of tables, nor does it use SQL to operate or analyze data (some NoSQL databases have SQL like query languages). NoSQL includes structured data, unstructured data and polymorphic data.

SQL usually deals with structured data.

3.1 The advantages of NoSQL

NoSQL database has many advantages. Its main advantages are high scalability and high availability. The high scalability of NoSQL database is mainly manifested in horizontal expansion through fragmentation. It is not complete to partition the data and place it on multiple computers in a way that keeps the order of the data. Vertical scaling means adding more resources to existing computers, while horizontal scaling means adding more computers to process data. First, vertical scaling is easier to implement than horizontal scaling. The automatic replication function in NoSQL database makes it highly available, because in case of failure, the data will be automatically copied to the previous consistent state. NoSQL also has some disadvantages, such as weak transaction management and inconsistent standards [2-4].

Through the analysis of the above advantages and disadvantages, NoSQL scenarios 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

3.2 Is the Abertawe Arts Center suitable for NoSQL database?

For Abertawe Arts Center, the above five points are analyzed respectively:

- (1) The scale and growth of data volume are within the control range of structured database.
- (2) There are relationships among the data in each table, which can't be fully expressed by NoSQL database.
- (3) Whether it's the booking of actor's venue, seat management or booking function, the data is structured and needs structured query.
- (4) Similarly, there are constraints and foreign key links between tables.
- (5) There is no need for regular expansion of the database. Because for the existing data, you can set a deadline to delete it.

To sum up, in my opinion, there is no need to introduce NoSQL database for the time being. If the requirements change in the future, I think the combination of SQL and NoSQL database will be more meaningful.

4. 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.

According to the needs of theater, this paper also gives a set of design scheme of distributed database. The segmentation methods of distributed database can be divided into horizontal segmentation and vertical segmentation. The horizontal segmentation method is based on the segmentation of the display position value in the field information table, which divides the data into three different positions. These three sites store the corresponding information for each site.

First of all, let's assume that there are already performance {ID, name, type, companyID, venueID, startTime, endTime}, and booking {ID, CustomerID, seatID}. We can divide the database into two parts: E1, E2 and E3. E1 and E2 represent the situations of performance groups with company ID 1, 2 and 3 respectively. According to the definition of horizontal segmentation, the database satisfies $E1 \cup E2 \cup E3 = P$, and has $E1 \cap E2 = \text{null}$, $E2 \cap E3 = \text{null}$, $E1 \cap E3 = \text{null}$. The specific segmentation statements are as follows:

P1: $\sigma \text{ customerID} = '1'$ (Performance) // The main auditorium

P2: $\sigma \text{ customerID} = '2'$ (Performance) // Dylan 1

P3: $\sigma \text{ customerID} = '3'$ (Performance) // Burton 2

Next, the possibility of vertical segmentation of database is discussed. I don't think the vertically segmented database is suitable for the needs of the current theater. If a table has many columns and only a few columns are commonly used data, we can split the table to avoid various addition, deletion, modification and query operations, which will access a lot of rarely used data each time. Obviously, there are not so many columns in the theater table for us to divide.

To sum up, horizontal segmentation is more suitable for the current needs than vertical segmentation.

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 Abertawe Art Center, we 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.
- [5] Moniruzzaman ABM, Syed Akhter Hossain. NoSQL database: New era of databases for big data analytics-classification, characteristics and comparison. arXiv Preprint arXiv:1307.0191; 2013.
- [6] Băzăr Cristina, Cosmin Sebastian Iosif. The transition from RDBMS to NoSQL. A comparative analysis of three popular non- relational solutions: Cassandra, MongoDB and couchbase. Database Systems Journal. 2014;5(2):49-59.