

2006796_Hao_Ke_Database Design for Abertawe Arts Centre

by Hao Ke

FILE	25260_HAO_KE_2006796_HAO_KE_DATABASE_DESIGN_FOR_ABERTAWE_ARTS_CENTRE_315194_310991831.DOCX (409.33K)		
TIME SUBMITTED	15-JAN-2021 04:01PM (UTC+0000)	WORD COUNT	2778
SUBMISSION ID	142287483	CHARACTER COUNT	14683



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Application of Machine Learning Report

Database design for Abertawe Arts Centre

Student Number: 2006796

Student Name: Hao Ke

January, 2021

TABLE OF CONTENTS

ABSTRACT	2
1 Introduction	3
2 Requirements Analysis	3
3 Database Design	5
4 Distributed Database Solution.....	7
Reference.....	10

ABSTRACT

In today's increasingly prosperous theater performance market and increasingly fierce competition, traditional performance management and ticketing systems can no longer meet the needs. The operating mode of the theater has gradually changed from renting out venues to purchasing performances and sharing the box office with the performing parties. Therefore, higher requirements are put forward on the theater ticketing system. Instead of relying solely on the ticketing system of other ticketing companies to achieve ticket sales, it is shifting to the establishment of a ticketing operation support system centered on the theater itself. The establishment and use of the theater-centric ticketing operation support system enable the theater to master complete ticket sales data and customer information, and on this basis, achieve diversified marketing and in-depth customer relationship management, thereby improving the operating level. This report uses Visio for database design and SQL server management studio to create the database.

1 Introduction

With the intensification of cultural demand, theater construction is also frantically increasing, and the demand for theater and entertainment production companies to book theater venues is also increasing. Therefore, between performance and performance project data management, The Abertawe Arts company needs to have better maintenance and management plan. The database is the foundation of the whole system, which is very important in the system design. The first step is to choose how to store the data. SQL and NoSQL are both ways to store data, but they store data in different ways. SQL databases provide relational tables to store data and data cannot be added to a table unless the schema for the table and field is predefined [1]. NoSQL databases use a key-value pair like JSON to store documents and data can be added at any time without having to define documents and collections in advance [1]. SQL has data integrity and NoSQL does not have data integrity [2]. SQL requires custom transactions. NoSQL is transactional when operating on a single document, but not when operating on multiple documents [2]. In conclusion, SQL database can use tables to store relationships and can be used on a large scale, which is more suitable for the Abertawe Arts Centre. Because the better database design can greatly reduce the difficulty of system development and design [3], the following article first analyzes and designs the database according to the given case, and then discusses and analyzes the database design and the problems which should be solved, and then creates the final database with SQL server. This task uses Microsoft SQL Server Management Studio as the compilation tool and SQL Server is a relational database management system launched by Microsoft.

2 Requirements Analysis

The construction of the new theater complex means that the Abertawe Arts company will have more customer groups, including art organizations that book theaters and audiences who want to watch art performances. The traditional paper record method will consume a lot of time to search and record, so it is necessary to use the database to solve some problems. To put it simply, the core requirement of the Abertawe Arts company is to handle the lease of theater venues and audience reservations for theater venues. Therefore, it is unquestionable to use theater venue information as the first step in creating a database table. Considering that the theater venue may have compound information, and it is uncertain whether the venue will be expanded, the venue-related information table is divided into two tables (venue and space) when designing the database. The "T_theatre_venue" table stores the number of the theater as the primary key, and stores daily rental of the venue and remarks information of the scheduled venue. Its primary key is used as the parent key of the "T_theatre_spaces" table. The "T_theatre_spaces" table stores the information of the divided theater spaces. It stores the venue id as the primary key, and the venue name and venue description as general attribute.

For the design of the "seat" table, the seats and the number of seats in the three theater venues are not the same in the requirements, so the table structure includes the

seat id, venue number, and seat number. The seat id and seat number are not the same property. The seat number is the only sign of the same space and the uniqueness constraint will also be designed here. This will be explained in the constraint summary below.

Because Abertawe art company need to record information about booking space organization, reservations require theatre company name, contact name and contact telephone, but considering the requirements of the production company, the details of the specific date will not store in the database. Information related to venue reservations is stored in two tables, one table does not include the reservation date, and the remaining attributes are the same, including the order number, number, theater reservation organization, name of the agent and phone number, type of performance (amateur or professional performance) and the price for the performance.

A table is also set up for the relevant information of the customer booking tickets, where the venue reservation id and seat id are set as the joint primary key, and the remaining attributes include the customer's name and phone number. The information for the initial ticket purchase and reservation of the venue is only the phone number and name, so there is no need to register the system. If the company or customer has too many attributes, a table must be created and saved, otherwise, there will be a lot of duplicate data. The registration system is not considered here, and a simple design is adopted. The database design is shown in Figure 1.

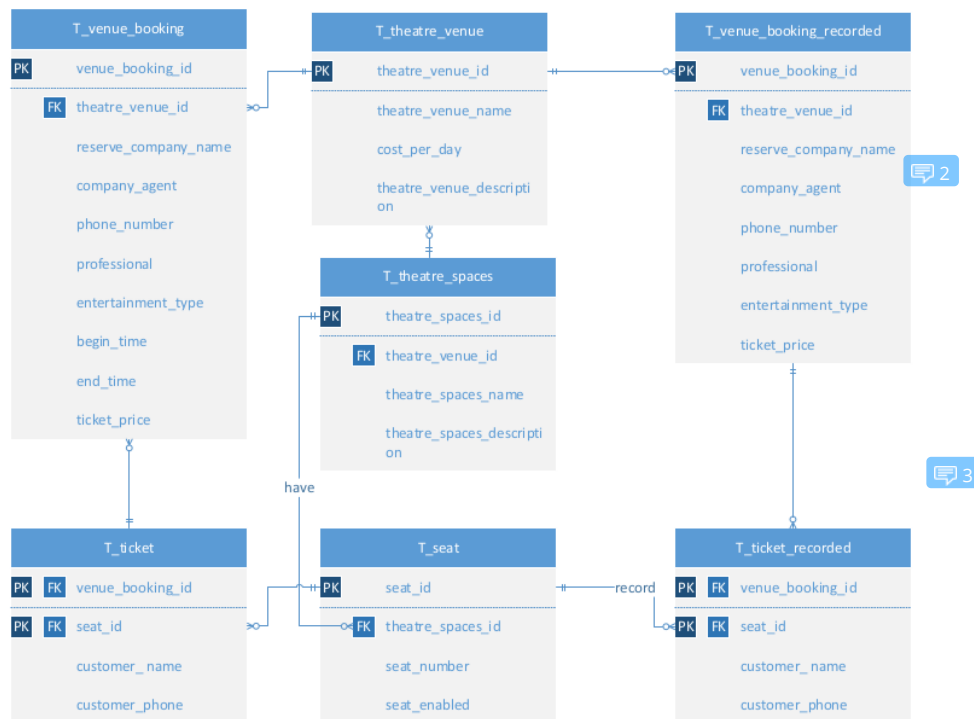


Figure 1 The database design

3 Database Design

Due to the limited description of actual case requirements, the theater venue is divided into two tables as analyzed above, and information is inserted in advance instead of using enumerated types. The ID of the venue is set to integer. It may be reasonable to use a short integer as the data type of the ID, but for safety reasons, it is set to integer and no short integer is used. Considering extreme conditions, for example, some venues may have 60,000 Seats, the Abertawe Arts company may like to start counting seats from seventy thousand, and there may even be seats numbered with letters. The primary key seat_id in the T_seat table uses bigint storage. The seat number in the "T_seat" table is the same as the venue ID. Although the seat numbers in the venue are consecutive, it is not certain whether there is a difference between the seats. At first, the seat number range (the upper and lower bounds of the seat number of a venue) was used to reserve the seat number, and now each seat is used instead. Keep as a separate record in the database. The seat number considers a variety of situations. The seat number is now changed to use a string to record char(10). This unique constraint will be explained in the following constraint summary. In addition to the ID and number, there is also a flag indicating whether the seat is available or not, namely seat_enabled. Of course, this attribute does not indicate whether the seat is reserved or not, because the relevant information about the reservation is kept in the ticket purchase table, here it just indicates whether the seat appears. What's the problem? For example, the naming of venues that are broken and cannot be used is unified as a string of 40 characters or less. The customer name and company name are both stored in char(50). Although I see university such institutions seem to have over 70 characters, but temporary or use 50 characters, not enough can also be through code change). The booking time in the "T_venue_booking" table is a full day, so it is recorded as a date. The appointment time can be expressed as the start time plus the end time, or as the start time plus the duration. Although the latter is more space-efficient, two time points are used to facilitate time overlap checking.

The ticket price in the "T_venue_booking" table is determined by the type of performance according to the demand description. It cannot be seen from the description whether the ticket price of the same time will vary according to the seat position. In addition, although the ticket price is the data type of money, the ticket price is relatively cheap (relative to the data type of money), so use float data records. The data type for storing the rental price of the space is money. Performance types (live band, drama, musical, or pantomime) are saved using strings. In MySQL, you can use an enumeration type, but SQL Server does not have an enumeration type, so you use the string char (12) plus check constraint to specify it.

In Task 2, if there is a reservation with an undetermined time, there are two solutions. One is to allow data exist empty, and the other is to store this reservation separately in a table that does not save time. The last one is selected. The main reason is that if there are too many reservations with uncertain time, there will be too many null data in one table.

A preview of the database is shown in Figure 2 and the data type of each attribute in each table is shown above. Figure 3 is an overall screenshot of the database and database tables

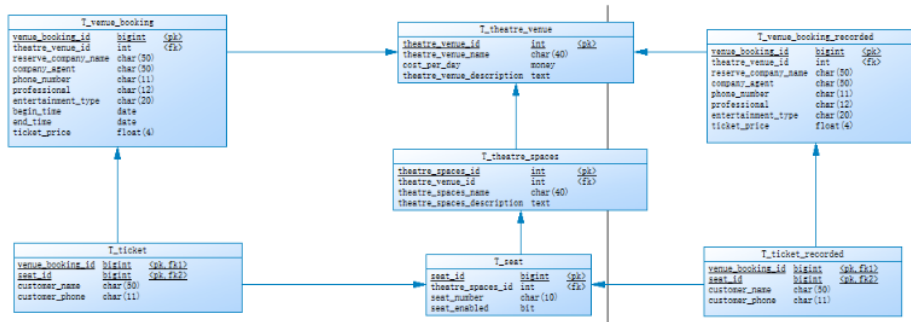


Figure 2 Database preview



Figure 3 Databases and database tables

The primary key constraint requires that the constraint column not be allowed to have duplicate values. This is usually done by setting the primary key to the column when designing the table. Like a primary key constraint, a unique key constraint guarantees the uniqueness of data in a column. Unlike a primary key constraint, a unique key constraint can have more than one, and a column setting a unique key constraint can have a NULL value, but only one NULL value[4]. Check constraints enforce domain integrity by limiting the values of one or more columns that can be entered or modified. In a relational database, tables are connected by primary and foreign keys, and the two tables that are connected are divided into parent tables (the table in which the primary key column is located) and child tables (the table in which the foreign key column is located). Foreign KEY constraints are designed to enforce referential integrity between tables. It defines relationships between columns in different tables in a relational database (parent and foreign keys) and requires that the values of a column or group of columns in one table (child table) must match the values of the related column or group of columns in another table (parent table) [5].

The constraints are summarized as follows.

1. Primary key constraints and foreign key constraints

Primary keys include theater_venue_id, theater_spaces_id, seat_id, and venue_booking_id. These are also used as foreign keys for other tables, especially in the T_ticket table where venue_booking_id and seat_id are used as joint primary keys.

2.The only constraints

- 1) Venue name is unique.
- 2) The space name of the same venue is unique.
- 3) The seat number of the same space is not repeated.
- 4) Tickets in a performance cannot have duplicate seats (the two fields are not duplicated at the same time, but the index is used here, in order to retrieve tickets faster, it is also a clustered index).
- 5) There is an index, which is the time index of venue reservation.

3.Check constraints:

- 1) The site rent is greater than 0.
- 2) Ticket price greater than 0(same if not officially registered).
- 3) The field of performance can only be Professional or Amateur (the same for those not officially registered).
- 4) The end time of site lease is longer than the start time.
- 5) The same site lease time does not overlap (this need to be specified in the trigger or in the stored procedure because of table operation).
- 6) The Venue location is located in the corresponding Venue location. (Note: The Venue location is located in the corresponding Venue location.)
- 7) Validate two prices, cost_per_day and ticket_price require validation with a check constraint greater than 0.

Transaction processing includes: when booking a space that doesn't have reservation time, there is only one set of operations, so there are few conflicts; the reserved site is converted to the confirmed site, read the recorded data first, then delete the unwanted data (you can insert and then delete), but to demonstrate rollback delete first and then insert if not insert will roll back the original delete, and advance tickets for unscheduled performances can be booked and canceled for unscheduled venues, and ticket information printed if anyone has bought a ticket.

4 Distributed Database Solution

Distributed database system is developed on the basis of centralized database system, it is suitable for the departments of decentralized units, allowing each department to store its commonly used data in the local, the implementation of local storage local use, so as to improve the response speed, reduce communication costs [7]. Assuming that each theater of the Abertawe Arts Company has separate Spaces, although their physical spaces may be close together, they operate as separate bookable locations, so the database information is separate for each theater, and the Abertawe Arts Company acts as the corporate headquarters, holding all the information. The various shards are defined as follows:

- 1) Horizontal fragmentation: data tables are divided into tuples, determined by fragmentation conditions

2) Vertical fragmentation: data tables are divided according to relationship attributes. Except for keywords, a certain attribute of the same relationship is not allowed to appear in multiple fragmentation of a relationship.

3) Mixed fragmentation: both horizontal and vertical fragmentation are included.

By definition, the “T_theatre_venue” table is suitable for horizontal fragmentation. The ID of different theaters is unique and can be retrieved according to the number of different theaters. The only predicate is theater_venue_id.

The “T_theatre_spaces” table is the space division information of each theater. It is relatively fixed for each theater. Unless the venue needs to be expanded, the information in the table remains basically unchanged. It is reasonable to use horizontal fragmentation and the set of predicates is theater_venue_id and theater_spaces_id. The “T_seats” table is appropriate for mixed fragmentation because each seat is retained as a separate record in the database, which conforms to the definition of horizontal fragmentation, and the only predicate is seat_id. Seat_number and seat_enabled properties can also be vertically partitioned out theatre_spaces_id, seat_number and seat_enabled properties to check the seat number and availability.

The suitable fragmentation schemas are shown in Table 1 to Table 6.

Table 1 T_venue_booking_recorded_A

T_venue_booking_recorded_A	
Venue_booking_id	Theatre_venue_id

Table 2 T_venue_booking_recorded_B

T_venue_booking_recorded_B									
Venue_booking_id	Theatre_venue_id	reserve_company_name	company_agent	phone_number	professional	entertainment_type	begin_time	end_time	ticket_price

Table 3 T_venue_booking_A

T_venue_booking_A	
Venue_booking_id	Theatre_venue_id

Table 4 T_venue_booking_B

T_venue_booking_B						
Theatre_venue_id	reserve_company_name	company_agent	phone_number	professional	entertainment_type	ticket_price

The “T_ticket” table and the “T_ticket_recorded” table used for ticket booking information can be divided vertically into seat booking and reservation information and the attributes are shown in Table 5 and Table 6.

Table 5 T_ticket_A/T_ticket_recorded_A

T_ticket_A/T_ticket_recorded_A	
Venue_booking_id	seat_id

Table 6 T_ticket_B/T_ticket_recorded_B

T ticket_B/T ticket recorded_B		
Venue_booking_id	customer_name	customer_phone

Figure 4 shows the global relationship reconstructed by the attributes of the individual tables and tables with the same attributes are represented in the diagram in a simplified way.

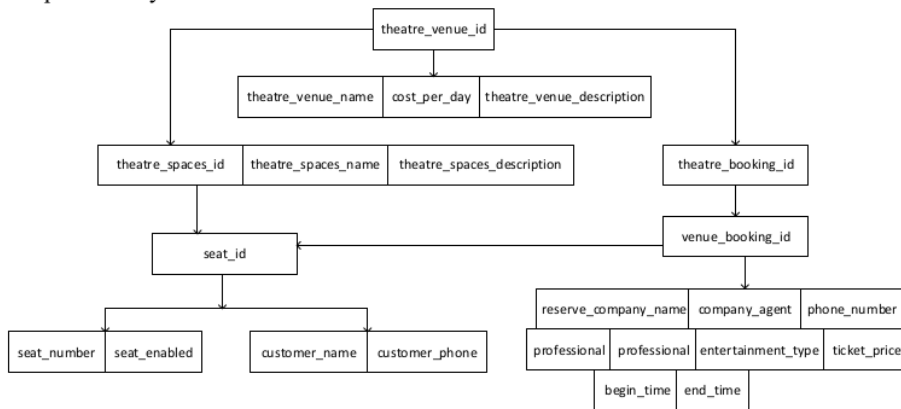


Figure 3 Reconstruction of the global diagram

Reference

- [1]Jing Han, Haihong E, Guan Le and Jian Du, "Survey on NoSQL database," 2011 6th International Conference on Pervasive Computing and Applications, Port Elizabeth, 2011, pp. 363-366, doi: 10.1109/ICPCA.2011.6106531.
- [2]Cattell R.Scalable SQL and NoSQL data stores[J]. ACM SIGMOD Record, 2010, 39(4):12-27.
- [3]Kießling, Werner, Köstler, Gerhard. Preference SQL—Design, Implementation, Experiences[C]// International Conference on Vldb. DBLP, 2006:990-1001.
- [4]赵洁.信息系统开发中的数据库完整性维护技术[J].信息技术,2013(07):137-139.
- [5]王洪香.基于 SQL Server 数据库完整性约束的几点探讨[J].办公自动化,2006(15):25-27.
- [6] Kikuchi S , Ito H , Nakane K , et al. Distributed database management system including logical database constituted by a group of physical databases[J]. 2002.
- [7] Kikuchi S , Ito H , Nakane K , et al. Distributed database management system including logical database constituted by a group of physical databases[J]. 2002.

FINAL GRADE

GENERAL COMMENTS

55/100

Instructor

A suitable report but try to use more references throughout. You have provided an ERD but be mindful of data duplication for example t-venue-booking and t-venue-booking-record would either a) contain the same data or b) the system would lose data integrity. You have provided suitable evidence of analysis of distributed databases and looked at transactions within the context of the case study.

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

Json

PAGE 5



Comment 2

data duplication



Comment 3

Good use of Visio for initial design

PAGE 6

PAGE 7

PAGE 8

