

# Abertawe Arts Centre Case Study

*by Yulei Bai*

---

|                |                                                                                     |                 |       |
|----------------|-------------------------------------------------------------------------------------|-----------------|-------|
| FILE           | 25246_YULEI_BAI_ABERTAWE_ARTS_CENTRE_CASE_STUDY_315194_524086<br>266.DOCX (186.98K) |                 |       |
| TIME SUBMITTED | 15-JAN-2021 05:39PM (UTC+0000)                                                      | WORD COUNT      | 2808  |
| SUBMISSION ID  | 142302065                                                                           | CHARACTER COUNT | 17655 |

**ASSIGNMENT**

**ON**

**Database Systems**

**ACCA7002**

**BY**

**YuLei Bai**

**2006323**

**UNIVERSITY OF WALES TRINITY SAINT DAVID**

**SUBMITTED TO**

**Dr Beverley Holland**

**Faculty of Architecture, Computing & Engineering**

**School of Applied Computing**

**UNIVERSITY OF WALES TRINITY SAINT DAVID**

**January, 2021**

# **Abertawe Arts Centre Case Study**

## **Abstract**

Based on the actual situation of Abertawe Art Center, this paper analyzes its design from several aspects such as index and transaction, which meets all the requirements of task one and task two and designs a distributed database on the basis of a centralized database.

**Keywords:**Distributed, indexed, transaction

## Content

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Chapter 1 Background and Analysis.....                               | 1  |
| 1.1 Case background.....                                             | 1  |
| 1.2 System function requirement analysis.....                        | 1  |
| Chapter 2 NoSQL and SQL.....                                         | 2  |
| 2.1 Database selection.....                                          | 2  |
| Chapter 3 Abertawe Art Center Database Design.....                   | 3  |
| 3.1 INDEX.....                                                       | 3  |
| 3.2 Analysis and solution of transaction-based A and B problems..... | 3  |
| Chapter 4 Abertawe Art Center Distributed Database Solution.....     | 6  |
| 4.1 Introduction to Distributed Database System.....                 | 6  |
| 4.2 Demand analysis of distributed database design.....              | 6  |
| 4.3 Design goals of distributed databases.....                       | 6  |
| 4.4 E-R diagram of global conceptual model.....                      | 6  |
| 4.5 Global operation mode.....                                       | 7  |
| 4.5.1 Reservation of venues.....                                     | 7  |
| 4.5.2 Program scheduling.....                                        | 7  |
| 4.5.3 Customer booking.....                                          | 8  |
| 4.6 Fragmentation design and analysis.....                           | 10 |
| Chapter 5 Summary.....                                               | 13 |

|                |    |
|----------------|----|
| REFERENCE..... | 14 |
|----------------|----|

# **Chapter 1 Background and Analysis**

## **1.1 Case background**

The Abertawe Art Center is about to be completed. The company hopes to provide venues for various organizations to perform various entertainment works. Abertawe Arts provides venue reservations for interested entertainment production companies, and then Abertawe Arts is responsible for arranging performances and selling tickets to customers. However, the company's current data is centralized, management is scattered, there is no connection, data redundancy is large, and retrieval is complicated and slow. Therefore, a system needs to be redesigned to meet the requirements of rapid information retrieval and processing.

## **1.2 System function requirement analysis**

Abertawe Art Center has three venues, each of which can host various performances. The company proposes to design and analyze the requirements of the entire system based on distributed. According to the analysis of the company's current business, it can be concluded that the company's current needs mainly implement the following functions:

- (1) Customer information management function: mainly the customer's information entry, related information modification, deletion and query, authority setting, customer ticket management, etc.
- (2) Company information management function: It is mainly the management of the basic information of the predetermined company, including the functions of changing, deleting and querying its information
- (3) Reservation site information management function: It mainly includes functions such as the entry, modification, deletion, query and report generation of information such as the number of the reservation company, the name of the selected theater, the reservation time, the performance of the program and the price.
- (4) Reservation ticket information management system: It is mainly responsible for the entry, modification, deletion, query and report generation of ticket formulation and customer order information.

## Chapter 2 NoSQL and SQL

### 2.1 Database selection

There are two main choices for commonly used databases in the market, one is SQL, which is a relational database, and the other is NoSQL, which refers to a non-relational database. Below we combine the requirements of the case and actually analyze which of them to use.

The difference between SQL and NoSQL:

1 SQL database tables are connected through relationships, and NoSQL databases use JSON to store documents. SQL has strong constraints, so it is not easy to make mistakes. Compared with SQL, NoSQL is more flexible.

2 SQL will define the table mode in advance before logical operation, and adding data requires certain requirements. There are no restrictions on adding data in NoSQL and no prior definition is required.

3 SQL can obtain the relevant data of multiple tables through a single statement, which is not available in NoSQL.

4 SQL data is kept intact through constraints, while NoSQL can store any data.

5 SQL performs more than two data updates through transactions. NoSQL can only maintain consistency when operating on a single document, and it cannot be achieved when multiple documents are used.

In the choice of SQL database and NoSQL database, we need to consider indicators such as data volume, concurrency, real-time performance, and consistency requirements. The case is an art center's management system for booking venues and

selling tickets. From these aspects, it is analyzed as follows:

Data volume: It has only three sites that do not require high data volume, and at the same time, the amount of concurrency is small, which is suitable for relational databases, while NoSQL is more suitable for systems with large traffic.

Real-time: The orders for the venue are pre-booked, so the real-time requirements are not high.

Consistency requirements: The tables in the database should be related to each other. For example, the number set in the seat will continue to be used in other tables, so there is a high requirement for consistency.

Considering these points, it is decided to use SQL database instead of NoSQL data.

## **Chapter 3 Abertawe Art Center Database Design**

### **3.1 INDEX**

The performance of a database is a key indicator to measure the quality of a database. The company uses the database to store business-related information for ease of use. After the database is established, frequent queries are required, so we need a structure to help us quickly find the part of the information we think of. The index can help us solve this problem.

An index is a structure that provides accelerated access to the table based on the values of one or more columns in the table. There are two main types of common indexes, which are clustered indexes and non-clustered indexes (that is, secondary indexes). Clustering is generally based on primary key clustering. A table can only have one clustered index, but there can be many secondary indexes.

The main business of the Art Center is the rental of venues, the arrangement of tickets and the sale of tickets. Considering that customers want to retrieve which shows will be released on a certain day, a secondary index is created in the TheatrePlay table to help customers find shows on a specific date. The code is as follows:

```
create NONCLUSTERED INDEX BYL_TheatrePlay_Time_Name ON
TheatrePlay(Play_DateTime,Play_Space_name);
```

If you want to query which specific types of programs the company has, we can create an index in the table to meet the needs of fast retrieval. The code is as follows:

```
create NONCLUSTERED INDEX BYL_TheatrePlay_Company_Type ON
TheatrePlay(CompanyNo,Play_Type);
```

Sometimes customers want to know their seat number before picking up the ticket. At this time, a secondary index of customers and seats can be established in the system to meet their needs. The code is as follows:

```
create NONCLUSTERED INDEX BYL_Ticket_Customer_Seat ON
Ticket(CustomerID,Seat_ID);
```

### 3.2 Analysis and solution of transaction-based A and B problems

Problem A: Venue bookings are recorded but the details of the specific date(s) requested by the production companies are not registered in the system. Insert data into the Renting table('The main auditorium','CO123','2021-1-15 08:00','2020-1-16 22:00',1000)和('The main auditorium','CO520','null','null',1000)

Code follow:

```
BEGIN TRANSACTION AddNewRenting

BEGIN TRY -- add try to catch any errors

-- -- Add the new DATA for new Renting

INSERT INTO Renting
(Theatre_name,CompanyNo,RentStartDate,RentEndDate,RentalPriceDays) VALUES
('The main auditorium','CO123','2021-1-15 08:00','2020-1-16 22:00',1000)
```

```

-- Add the new DATA for new Renting

INSERT INTO Renting
(Theatre_name,CompanyNo,RentStartDate,RentEndDate,RentalPriceDays) VALUES
('The main auditorium','CO520','null','null',1000)

-- Commit ALL changes, inserts successful

COMMIT TRANSACTION AddNewRenting;

END TRY

BEGIN CATCH -- add catch to handle error

-- Rollback ALL inserts, an error has occurred.

ROLLBACK TRANSACTION AddNewRenting;

RAISERROR('Rolling back, an error occurred!', 16, 1);

END CATCH;

```



```

SELECT * FROM [Renting] WHERE Theatre_name = 'The main auditorium';

```

Problem B: Customers are able to book tickets items that are not registered as event:

```

BEGIN TRANSACTION AddNewPlay

BEGIN TRY -- add try to catch any errors

-- Add new record

INSERT INTO TheatrePlay
(PlayID,Play_Name,Play_Type,Play_Space_name,Play_Category,Play_DateTime,Co
mpanyNo) VALUES ('P206', 'GOOD DAY','musicals','The main

```

```
auditorium','Professional','2021-1-15 19:00','CO123')
```

```
-- Add new record
```

```
INSERT INTO TheatrePlay  
(PlayID,Play_Name,Play_Type,Play_Space_name,Play_Category,Play_DateTime,Co  
mpanyNo) VALUES ('P207', 'happy new year','musicals','Burton  
2','Amateur','null','CO520')
```

```
-- Commit ALL changes, inserts successful
```

```
COMMIT TRANSACTION AddNewPlay;
```

```
END TRY
```

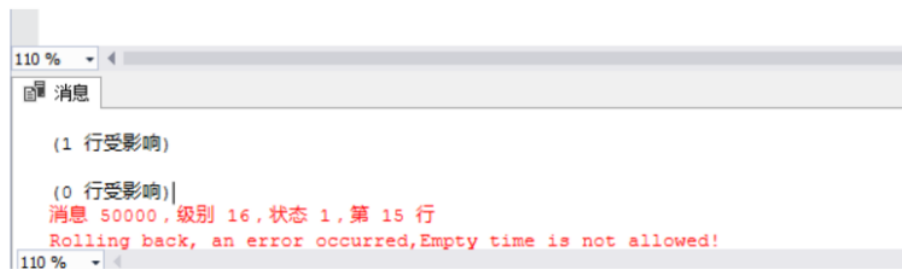
```
BEGIN CATCH -- add catch to handle error
```

```
-- Rollback ALL inserts, an error has occurred.
```

```
ROLLBACK TRANSACTION AddNewPlay;
```

```
RAISERROR('Rolling back, an error occurred,Empty time is not allowed!',  
16, 1);
```

```
END CATCH;
```



```
select * from TheatrePlay;
```

The screenshot shows a results window with the following data:

|   | PlayID | Play_Name                 | Play_Type  | Play_Space_name     | Play_Category | Play_DateTime               | CompanyNo |
|---|--------|---------------------------|------------|---------------------|---------------|-----------------------------|-----------|
| 1 | P200   | The Phantom Of the Opera  | musicals   | The main auditorium | Professional  | 2021-01-10 19:00:00.0000000 | C0123     |
| 2 | P201   | Death of a Salesman       | plays      | Burton 2            | Amateur       | 2021-01-11 14:00:00.0000000 | C0144     |
| 3 | P202   | Gentlemen of Nerve        | pantomimes | Dylan 1             | Amateur       | 2021-01-13 18:00:00.0000000 | C0520     |
| 4 | P203   | London Symphony Orchestra | bands      | The main auditorium | Professional  | 2021-01-12 16:00:00.0000000 | C0123     |
| 5 | P204   | Romeo and Juliet          | plays      | Burton 2            | Amateur       | 2021-01-14 16:00:00.0000000 | C0477     |
| 6 | P205   | Les Miserables            | musicals   | Dylan 1             | Professional  | 2021-01-14 19:00:00.0000000 | C0520     |

## **Chapter 4 Abertawe Art Center Distributed Database Solution**

### **4.1 Introduction to Distributed Database System**

Distributed database system includes distributed database management system and distributed database. The distributed database is actually a whole, but it is stored on different physical nodes in reality, and the database in different places can be accessed through the network. For the user experience, there is no difference between a distributed database and a centralized database.

Distributed database is developed on the basis of the network, and it is suitable for decentralized departments. The department improves the response speed by storing frequently used data locally. Distributed database has many advantages, such

as it is more suitable for distributed management and control, its structure is more flexible, its reliability is high and it is easier to expand.

## **4.2 Demand analysis of distributed database design**

The Art Center has three venues. For all venues, a unified system is responsible for their information recording in the centralized database. The venue reservations have a unified reservation system, and the booking system is also responsible for booking tickets. But in a distributed system, the originally unified system can be separated. The original ticket booking system can be separated into its own ticket booking system for each venue, so that the pressure on the data center can be shared. Each venue manages its own user information. Logically, the entire system is divided into several parts, such as ticket booking, venue reservation, and customer information maintenance. Physically, they are three different venues and departments within each venue. They are connected through the same structure of the database management system to share their data.

## **4.3 Design goals of distributed databases**

When designing a distributed database, the designer should ensure that the database data and applications have the greatest degree of locality, while ensuring its reliability. On the basis of the centralized database of the Art Center, the segmentation strategy is used to reasonably divide it. The main structure of the art center is three venues, so the distributed design is based on these three venues. Each venue is equivalent to a small centralized database. .

## **4.4 E-R diagram of global conceptual model**

The theater database is distributed in three theaters. The database stores customer information, ticket booking information, program scheduling and venue rental data. According to this case, three physical objects can be easily found: the venue, the customer, and the company. Their basic attributes are as follows:

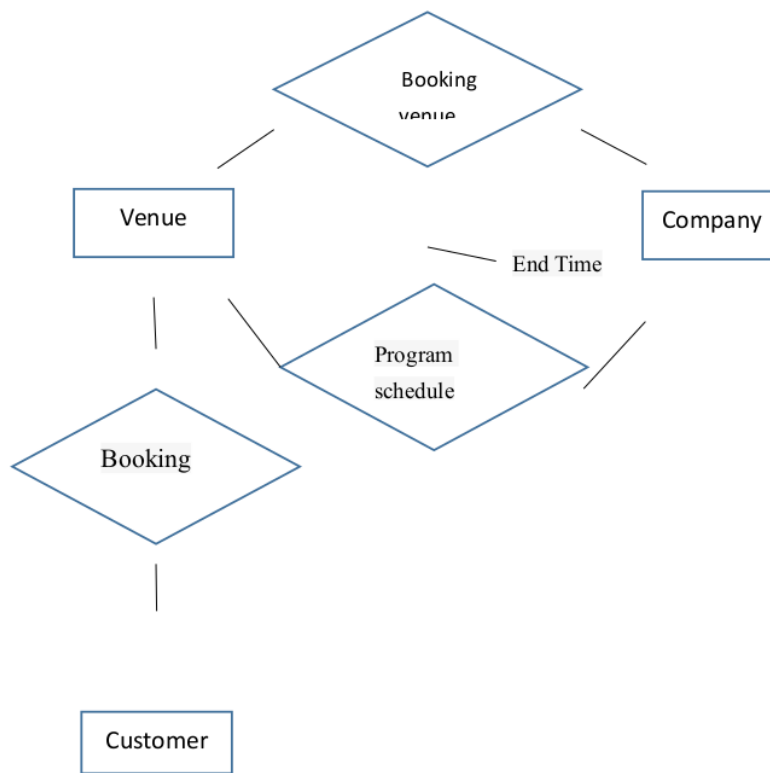
Customer: name, phone, age, email

Venue: seats, shows,

Company: name, phone, email

According to the three entities and applications, the global conceptual model E-R diagram can be obtained

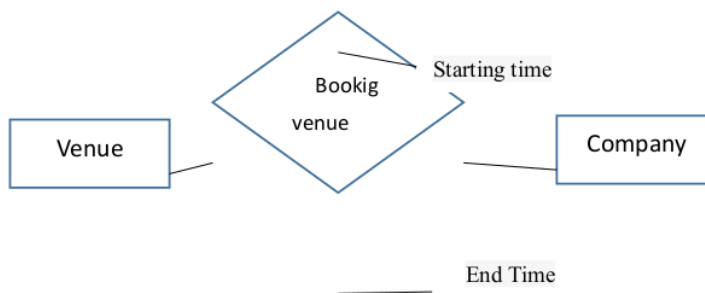
— Starting time



## 4.5 Global operation mode

### 4.5.1 Reservation of venues

Used for company scheduled site operations



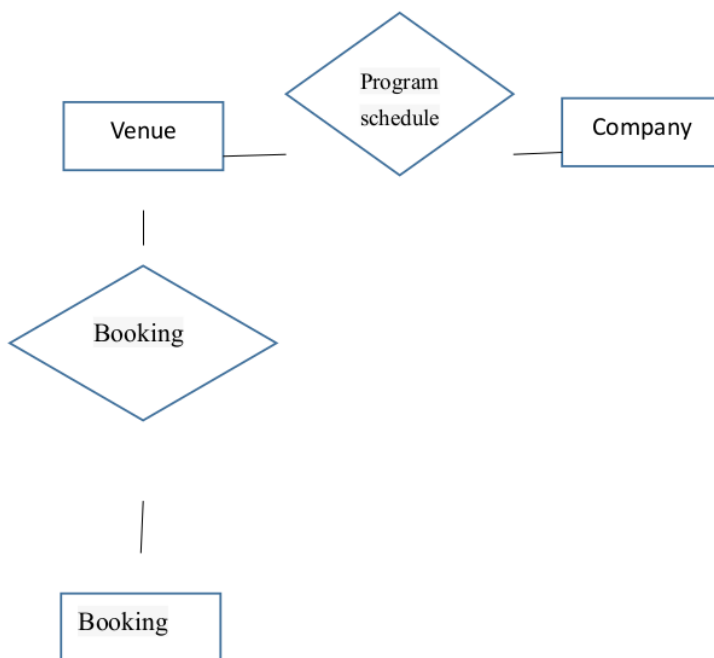
### 4.5.2 Program scheduling

Show the specific arrangement of different programs.



### 4.5.3 Customer booking

Used to display information about customer bookings



Through the above analysis combined with the E-R diagram, the entities in the system, the attributes of the entities, and the associations between the entities can get a simple relationship model:

Customer (Customer\_ID, FirstName, Surname, Age, seat\_venue\_name)

Company(CompanyNo,Company\_Name,PhoneNo,rent\_venue\_name)

TheatrePlayPlay(PlayID,Play\_Name,Play\_Type,Play\_venue\_name,Play\_Category,Play\_DateTime,CompanyNo)

Seat(Seat\_ID,Space\_name,Seat\_Type,Seat\_Number)

Ticket(Ticket\_ID,PlayID,Seat\_ID,TicketPrice,CustomerID)

Renting(Theatre\_name,CompanyNo,RentStartDate,RentEndDate,RentalPriceDays,NumberDays)

From the above analysis and requirements, we can build a database, the table of the database is as follows

| Column          | Datatype | Length | Null Or Not | Comment |
|-----------------|----------|--------|-------------|---------|
| Customer_ID     | CHAR     | 5      | N           | PK      |
| FirstName       | VARCHAR  | 6      | N           |         |
| Surname         | VARCHAR  | 6      | N           |         |
| Age             | VARCHAR  | 2      | N           |         |
| seat_venue_name | VARCHAR  | 21     | N           |         |

Table 1: Customer table

| Column          | Datatype | Length | Null Or Not | Comment |
|-----------------|----------|--------|-------------|---------|
| CompanyNo       | VARCHAR  | 5      | N           | PK      |
| Company_Name    | VARCHAR  | 10     | N           |         |
| PhoneNo         | SURNAME  | 12     | N           |         |
| rent_venue_name | VARCHAR  | 21     | N           |         |

Table 2: Company table

| Column          | Datatype  | Length | Null Or Not | Comment |
|-----------------|-----------|--------|-------------|---------|
| PlayID          | VARCHAR   | 4      | N           | PK      |
| Play_Name       | VARCHAR   | 30     | N           |         |
| Play_Type       | SURNAME   | 10     | N           |         |
| Play_venue_name | VARCHAR   | 21     | N           |         |
| Play_Category   | VARCHAR   | 20     | N           |         |
| Play_DateTime   | DATETIME2 |        |             |         |
| CompanyNo       | VARCHAR   | 5      | N           |         |

Table 3: TheatrePlayPlay table

| Column      | Datatype | Length | Null Or Not | Comment |
|-------------|----------|--------|-------------|---------|
| Seat_ID     | VARCHAR  | 5      | N           | PK      |
| Space_name  | VARCHAR  | 21     | N           |         |
| Seat_Type   | VARCHAR  | 12     | N           |         |
| Seat_Number | INT      |        | N           |         |

Table 4: Seat table

| Column    | Datatype | Length | Null Or Not | Comment |
|-----------|----------|--------|-------------|---------|
| Ticket_ID | VARCHAR  | 5      | N           | PK      |

|             |             |   |   |  |
|-------------|-------------|---|---|--|
| CompanyNo   | VARCH<br>AR | 4 | N |  |
| Seat_ID     | VARCH<br>AR | 5 | N |  |
| TicketPrice | INT         |   | N |  |
| CustomerID  | CHAR        | 5 | N |  |

Table 5: Ticket table.

| Column          | Datatype      | Length | Null Or Not | Comment |
|-----------------|---------------|--------|-------------|---------|
| Theatre_name    | VARCH<br>AR   | 21     | N           | PK      |
| PlayID          | VARCH<br>AR   | 5      | N           | PK      |
| RentStartDate   | DATETI<br>ME2 |        |             | PK      |
| RentEndDate     | DATETI<br>ME2 |        |             |         |
| RentalPriceDays | INT           | 5      | N           |         |
| NumberDays      |               |        |             |         |

Table 6: Renting table

After designing these six tables, it can basically meet the needs of the art center, and the table itself also meets the third paradigm. This case is the database design of the affairs management system of the Abertawe Art Center. It needs to be designed according to the logical structure of the table, combined with the actual location of the performance venue and distributed design requirements, and the data is distributed to each venue.

#### 4.6 Fragmentation design and analysis

Fragmentation design is to allow each entity to choose the appropriate fragmentation strategy. In our case, horizontal sharding is the most suitable case. Because the business of the three venues is basically the same, the data they store is basically the same. Only when their applications are different and the required data are different, will vertical sharding be considered.

The first customer table can be analyzed after inserting data.

| Customer_ID | Firstname | Surname | Age | seat_venue_<br>name    |
|-------------|-----------|---------|-----|------------------------|
| C0001       | Lebron    | James   | 33  | The main<br>auditorium |
| C0002       | Naomi     | Scott   | 22  | Burton 2               |
| C0003       | Jose      | Burdis  | 24  | Dylan 1                |
| C0004       | John      | Evens   | 41  | The main<br>auditorium |
| C0005       | Zoe       | Harris  | 26  | Burton 2               |
| C0006       | Justin    | Bieber  | 27  | Dylan 1                |

Table 7 Customer Data table

In the Customer table with data inserted as above, according to the design principle, the information center of each venue stores its own part of customer information, uses seat\_venue\_name in the table as the basis for fragmentation, and performs horizontal segmentation to obtain different logical segments. There are three different values of seat\_venue\_name, which respectively refer to three different venues. After classification according to this value, the fragmented result can be obtained.

| Customer_ID | Firstname | Surname | Age | seat_venue_<br>name    |
|-------------|-----------|---------|-----|------------------------|
| C0001       | Lebron    | James   | 33  | The main<br>auditorium |
| C0004       | John      | Evens   | 41  | The main<br>auditorium |

Table8 Customer For Main Auditorium table

| Customer_ID | Firstname | Surname | Age | seat_venue_<br>name |
|-------------|-----------|---------|-----|---------------------|
| C0002       | Naomi     | Scott   | 22  | Burton 2            |
| C0005       | Zoe       | Harris  | 26  | Burton 2            |

Table9 Customer For Burton 2 table

| Customer_ID | Firstname | Surname | Age | seat_venue_<br>name |
|-------------|-----------|---------|-----|---------------------|
| C0003       | Jose      | Burdis  | 24  | Dylan 1             |
| C0006       | Justin    | Bieber  | 27  | Dylan 1             |

Table10 Customer For Dylan 1 table

For the Ticket table, there is no direct venue information in the table, and the seat numbers are different for each, so horizontal slicing is not suitable here. Through analysis, it can be found that the program ID and the ticket price can form a vertical segment, and the remaining Ticket\_ID, Seat\_ID and CustomerID can form a vertical segment. They represent the price of the performance and the customer's seat number query.

| Ticket_ID | PlayID | Seat_ID | TicketPrice | CustomerID |
|-----------|--------|---------|-------------|------------|
| T0001     | P200   | S001    | 25          | C0001      |
| T0002     | P201   | Bu015   | 11          | C0002      |
| T0003     | P202   | Dy001   | 9           | C0003      |
| T0004     | P203   | S255    | 30          | C0004      |
| T0005     | P204   | Bu078   | 9           | C0005      |
| T0006     | P205   | Dy032   | 20          | C0006      |

Table10 Ticket Data table

| PlayID | TicketPrice |
|--------|-------------|
| P200   | 25          |
| P201   | 11          |

|      |    |
|------|----|
| P202 | 9  |
| P203 | 30 |
| P204 | 9  |
| P205 | 20 |

Table11 Play\_Price table

| Ticket_ID | Seat_ID | CustomerID |
|-----------|---------|------------|
| T0001     | S001    | C0001      |
| T0002     | Bu015   | C0002      |
| T0003     | Dy001   | C0003      |
| T0004     | S255    | C0004      |
| T0005     | Bu078   | C0005      |
| T0006     | Dy032   | C0006      |

Table12 Customer\_Seat table

## **Chapter 5 Summary**

In this case, we first discussed and analyzed whether the database uses SQL or Nosql, and compared the two in terms of data volume, concurrency, and real-time performance based on the actual situation of the Art Center. The second is that the case-based database has designed three secondary indexes from different perspectives. Then use transaction rollback and commit to solve the two problems of Abertawe Art Center. Finally, the fragmentation design and analysis of the entire database system from a distributed perspective. Although the basic requirements are met, there are still many problems and defects in this system, and I hope that it can be improved even better in the future.



## REFERENCE

- [1] 高萌.基于分布式 NoSQL 的医疗大数据档案检索系统设计[J].电子设计工程,2020,28(21):24-28.
- [2] Andrew C Oliver. SQL lessons for NoSQL developers[J]. InfoWorld.com,2020.
- [3] Vamsi Krishna Myalapalli and Bhupati Lohith Ravi Teja, "High performance PL/SQL programming," 2015 International Conference on Pervasive Computing (ICPC), Pune, 2015, pp. 1-5.
- [4] A. H. Al-Sanhani, A. Hamdan, A. B. Al-Thaher and A. Al-Dahoud, "A comparative analysis of data fragmentation in distributed database," 2017 8th International Conference on Information Technology (ICIT), Amman, 2017, pp. 724-729.
- [5] X. Sun, B. Jiang and X. He, "Database Query Optimization Based on Distributed Photovoltaic Power Generation," 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), Xi'an, 2018, pp. 2382-2386.

# Abertawe Arts Centre Case Study

## GRADEMARK REPORT

### FINAL GRADE

63/100

### GENERAL COMMENTS

#### Instructor

Suitable report presentation though you must improve your use of referencing. You have clearly research SQL / NoSQL but use more referencing to reflect your points. Unusual way of presenting and ERD why not use the one generated with MS sql. Good technique linking customer to seat however, more testing to demonstrate this. You have analysed the distributed database requirement bit try to follow the representation method we use on lectures.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6



#### Comment 1

JSON

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

---

PAGE 15

---

PAGE 16

---

PAGE 17

---

PAGE 18

---

PAGE 19

---

PAGE 20

---

PAGE 21

---

PAGE 22

---

PAGE 23

---



**Comment 2**

please translate to English