Journal of Positive School Psychology http://journalppw.com
2022, Vol. 6, No. 8, 7033-7041

Design And Implementation Of Database For Shared Facility Reservation System In School

Jae Moon Lee[1], In Hwan Jung[2], Kitae Hwang[3]

[1,2,3]*School of Computer Engineering, Hansung University, Korea*
[1]*jmlee@hansung.ac.kr*, [2]*ihjung@hansung.ac.kr*, [3]*calafk@hansung.ac.kr*

Abstract: This paper is a study on database design and implementation of reservation system for small shared facilities in schools. The reservation system has the characteristics that changes in the database must be communicated to the relevant users in real time. The database was designed using ERM from the requirements of the reservation system at school, and the database schema was derived from ERD, the result of ERM. In particular, it was implemented using Firestore provided by Google to notify related users of database changes in real time. For this, users and facilities data, which are core data, were designed and implemented to support efficient search by designing as the root of the hierarchical database.

Keywords: Database, Firebase, NOSQL, Booking, Shared Facility, Real Time

1. Introduction

Facilities that are difficult for individuals to own are built in common, so facilities shared by many people are increasing. These facilities generally have the characteristics of being used by far more people than the number of built facilities. For example, tennis courts and soccer fields in parks are typical facilities. These facilities allow many people to use them at their allotted time by making a reservation. There are many facilities shared by several students at the school. As in parks, sports facilities such as soccer fields and tennis courts are representative, and many students share and use library seats, computers in the computer room, and books in the library with limited facilities.

Reservations are essential to share these limited facilities among multiple students (Teng, 2014). In the existing reservation system, it is common to use an application program running on the web. The disadvantage of these reservation systems is that they do not have a push service function, so it is difficult to deliver the reservation details in real time. Recently, the reservation system is being developed as an application program running on a mobile phone (McTavish et al., 2010). This is because not only many students want to make a reservation on a mobile phone, but also real-time reservation information by push service can be easily implemented on a mobile phone (Sagar et al., 2016). A representative database system based on the Push service is firebase supported by Google (Mokar et al., 2019).

This paper is to design a database for a reservation system for relatively small shared facilities such as schools and study rooms. This is a database design for a system that only manages reservations for shared facilities without interworking with other management systems because the facility is small. The designed database is implemented in firestore in firebase. Since it is implemented in Firestore, the results of bookings made by other users can be displayed on all users' mobile phones in real time.

The chapter 2 investigates studies related to database design of reservation system, and the chapter 3 describes database design. The chapter 4 discusses the database implementation result

using Firestore characteristics and the reservation system implementation.

2. Related Works

2.1 Design of Database

Designing a database refers to the process of creating a database to support the operation and purpose of an organization. The purpose is to express the data required by the application and the relationship between the data. Developing a database follows a typical project life cycle process. Good database design represents all aspects of data over time, minimizes duplication of data items, and provides efficient access to the database. It also provides the integrity of the database and should be easy to understand.

Database design is largely divided into conceptual design and logical design. Conceptual design is primarily about finding data items that need to be stored and managed and discovering their relationships. The greatest characteristic of conceptual design is that it is completely independent of logical design. The most used method in this stage is ERM(Entity-Relationship Model). Logical design is the stage of finding out how to represent data. In other words, it is to design how data will be stored and managed using the results of conceptual design. The relational data model has been widely used at this stage since the 1970s (Han et al., 2011).

2.2 ERM

The conceptual design of the database is to understand the infinity and continuity of the real world in order to obtain the structure of information, and to express the perception of the real world as an abstract concept in order to communicate with others. Abstract concepts are expressed as entities and relationships, which is called ERM (Badia, 2004). Therefore, finding entities in ERM is very important. ERM refers to a data model that describes information obtained from requirements as entities, attributes, and relationships (Badia, 2004; Omar, 2021).

An entity means an object that exists alone, and the same object does not exist. For example, when student information includes student number, name, and grade, if there is only one student with the same three information, it is called an entity. In other words, each student becomes an object. The elements that make up an entity are called attributes. Relation refers to the relationship between entities. For example, when there is an entity called student and subject, there is a relationship called taking between student and subject.

The result of the ERM process is represented to as an Entity-Realtionship diagram (ERD). In ERD, entities are typically represented by rectangles, and properties are represented by ellipses. Also, the relationship is expressed as rhombus (Omar, 2021).

2.3 Realtime Database: Firestore

Since the 1980s, the relational database has been mainly used as the database system. The main reason for this is that it not only minimizes data redundancy, but also provides powerful query functions like SQL. A major weakness of relational databases is that they require a standardized format for all data. Recently, data appearing on SNS, etc., is more unstructured than standardized data. For such data, more efficiency of sharing rather than minimization of data duplication is required (Haritsa et al., 1992; Chai et al., 2019).

A database that satisfies these requirements is a NOSQL database. NOSQL (original meaning: non SQL or non relational) databases provide a mechanism for storing and retrieving data using a less restrictive consistency model than traditional relational databases. Motives for these approaches include design simplification, horizontal scalability, and fine-grained control. A NOSQL database is a highly optimized key-value storage for simple search and append operations, with the goal of yielding significant performance gains in terms of latency and throughput. NOSQL databases are widely used for commercial use of big data and real-time web applications. Also, the NOSQL system is sometimes called "Not only SQL" in the sense

that it emphasizes the fact that a SQL-like query language can be used (Haritsa et al., 1992).

A representative product of NOSQL is Firebase provided by Google. Firebase is a "toolset for developing, improving, and growing apps". Without these tools, developers typically have to build a significant portion of the service themselves (Mokar et al., 2019; Varshney et al., 2019). However, developers do not like to make all those details because they have to focus on the user experience (UX) of the app. These include analytics, authentication, databases, configuration settings, file storage, and push messages. When these services made with Firebase are hosted in the cloud, developers can scale the app without much effort. Firebase is a third-generation database based on NOSQL. Rather than relational databases such as Oracle and MySQL, which are currently widely used, Firebase introduced a fast and simple NOSQL-based database in document format. Also, unlike other databases, Firebase supports RTSP (Real Time Stream Protocol) database. RTSP is literally a method of transmitting data in real time. If you use this method, the amount of code is drastically reduced compared to making and communicating with a socket-based server, so you can create the configuration you want with just a few lines of code. Both Firestore (new version database) and Realtime Database (old version database), which are parts of Firebase, have very poor query functions. People familiar with SQL will be very confused when using a Firebase database. It is not searched with the usual OR statement, nor does the LIKE statement exist, so similar characters or data cannot be searched for. So users who use Firebase are using a method that receives all this data and filters it on the device.

3. Design of database for facilities in school

3.1 Requirements for database

The school has a variety of facilities. Many of these facilities require several students to share and use them at different times. For example, a computer lab may have 30 computers and 300 users. The same is true for tennis courts and soccer fields in schools. If there are many users but the number of facilities is limited, these facilities should be shared through reservation.

Restricted sharing of facilities must be made by reservation. In order to support such a reservation, there should be restrictions on usage time for each facility. This is because one person cannot occupy one facility indefinitely. In the same sense, one person should be restricted in the use of the same type of facility for a certain period of time. For reservations, the facilities available at a specific time should be shown to users. Reservation opening hours should be limited according to the characteristics of each facility.

3.2 Main entities

The most important thing in database design is finding entities. This paper found the entities for the reservation system for shared facilities for school facilities as follows:

- User: Users are all people including students and teachers who have access to shared facilities in the school. Of course, facilities available for each user may be limited. That is, user U1 can use facilities F1, F2, and F3, and user U2 can use facilities F2, F3, and F4. In addition, since the user has a grade, a discount may be applied differently depending on the grade. For example, even on the same computer, teachers have higher ranks than students, so they should be able to make reservations in advance.

- Facilitiy: A facility is an object to be shared. For example, school computers, tennis courts, and soccer fields are typical facilities. These facilities are segregated by specific time units, and are usually reserved for each separate unit. It must also be stored in quantities for the same facility. For example, if there are 30 identical computers in a computer room, a quantity of 30 may be displayed in a facility

object called Computers. This system is a database for reservations only. If it is a database for facility management, it should be stored as each different entity for 30 computers.

- Schedule: A schedule refers to a schedule that can be reserved for each facility. For example, if an on-campus tennis court opens at 10 clock and closes at 18 clock, reservations can be made in two-hour increments, four reservation schedules can be created for 10, 12, 14, and 16 clock.

- Discount rate: Most of the facilities in the school are free, but expensive facilities may charge a fee, and the usage fee may vary by time of day. For example, if you use your computer early in the morning, you can give it a 10% discount. The discount rate can make you to operate a variety of these charges.

3.3 ERD

ERD is a schematic representation of the results of ERM, a conceptual modeling. It is to find the relationship between each entity based on the previously found entity and to represent them schematically. Figure x is the ERD for the entities found earlier. Here, the rectangle refers to entities and the rhombus refers to relationships. N means that the mapping cardinality is multiple, and 1 means that the mapping cardinality is at most 1.

The Rank relationship indicates the user's priority in using a specific facility. For example, suppose you have students and teachers, and your facility has a tennis court. For an example, when the teacher ranks 1 and the student is 2, the teacher is opened two weeks in advance and the student is one week in advance. By applying rank in this way, the reservation right for teachers can be higher than that of students. These ranks will vary from facility to facility. Therefore, each user must be assigned a rank for each facility. Therefore, one user should be given a rank for several facilities, and one facility should be given a rank for several users. Therefore, the mapping cardinality of the relationship becomes N:N. The Allocate relationship is the relationship between the reservation schedule and the facility. One facility must make one schedule for each reservation time unit. Therefore, one facility is related to several schedules, whereas one schedule is always related to one facility. Therefore, their mapping cardinality becomes 1:N. The The Discount relationship applies a discount to a specific time. That is, it is to apply different rates for each facility and each time period. Since one facility has different discount rates for each time period, it is related to several discount rates, and since one discount rate is related to only one facility, it has a 1:N relationship. The Apply relationship applies a discount rate to the reservation schedule. Therefore, one discount rate can be applied to several schedules, and only one discount rate can be applied to one schedule. Therefore, their mapping cardinality becomes 1:N. Finally, there is the Booking relationship. One user can reserve several schedules, and one schedule can be reserved for several users. This is because it is necessary to manage the case where a reservation is reserved for user U1, then user U1 cancels the reservation, and user U2 makes a reservation for the canceled reservation. Therefore, their relationship becomes N:N. If there is no need to store the information that user U1 cancels, one reservation is reserved for only one user, so then there is a 1:N relationship.

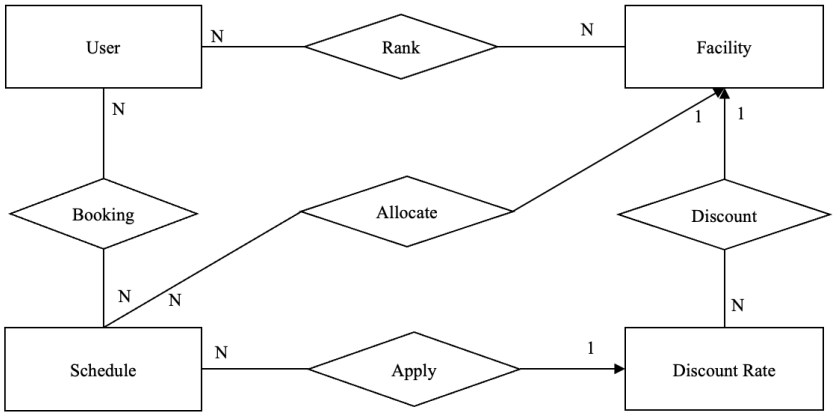

Figure 1. ERD for reservation of facilities in school

3.4 Attributes and schema of database

A database schema is to express entities and relationships as tables and to define properties necessary for configuring each table. In addition, a primary key should be defined as a table requirement, and referential constraints should be applied to foreign keys. In Figure 2, 4 entities and 5 relationships are defined. Each entity and relationship must be converted into a table. Before being converted to a table, each of the necessary attributes must be found and the key attribute must be found. In Figure 2, the bold and underlined attributes or attribute pairs indicate primary keys, and the attributes or pairs of attributes indicated in italics are foreign keys. Foreign keys must satisfy referential constraints.

The User entity must have attributes such as ID, password, name, phone number, contact information, registration date and time, and approval date and time. The Facility entity has more various attributes. Attributes are ID, name, quantity, open time, close time, reservation unit, and cost. The Discount Rate entity is a Weak entity and must be associated with a facility. Also, the discount rate may vary by time period. The attributes of this entity are the ID of the facility, the confirmation start time, and the discount rate. The Schedule entity has a price according to usage time, reservation status, and usage time. The Rank relationship includes the date and time the rank was applied, user ID, facility ID, grade, and grade application date. The Discount relationship is the start date of the discount application and the facility ID. The Apply relationship has properties of usage time and facility ID. The Booking relationship has the properties of schedule time, facility ID, user ID, and reservation time. The schema of the tables with these characteristics is as shown in Figure 2.

```
User(userid, password, name, phone, registerDate, allowDate)
Facility(id, name, quantity, openTime, closeTime, bookingUnit, price)
Schedule(facilityId, usingTime, booked, scheduledPrice)
DiscountRate(facilityId, discountTime, rate)
Rank(userId, facilityId, grade, appliedDate)
Discount(facilityId, discountTime)
Apply(facilityId, usingTime)
Booking(userId, bookingTime, facilityId, usingTime, canceled, contact)
```

Figure 2. Database schema reservation of facilities in school

In the above tables, a table created by a relationship can be combined into a table created by an entity in a range where data duplication does not occur. Typically, data duplication does not occur even if a relationship table having a 1:N relationship is merged with the N-side entity

table. By applying these rules, the Apply table can be included in the Schedule table, and the Discount table can be included in the Discount Rate table. Therefore, it is finally reduced to 6 tables as shown in Figure 3. Attributes or pairs of attributes underlined indicate primary keys.

```
User(id, password, name, phone, registerDate, allowDate)
Facility(id, name, openTime, closeTime, bookingUnit, price)
Schedule(facilityId, usingTime, booked, scheduledPrice)
DiscountRate(facilityId, discountTime, discountRate)
Rank(userId, facilityId, grade, startingDate)
Booking(userId, bookingTime, facilityId, usingTime, canceled, phone)
```

Figure 3. Reduced database schema without data duplications

Figure 4 is an example of a database according to Figure 3. The value of the primary key must be unique in the corresponding table and must satisfy the referential constraint on the foreign key. For example, the userId value in the Booking table must be one of the id values in the User table.

User Table

id	password	name	phone	registerDate	approveDate
jmlee	123456	Jae Moon Lee	8210123456	2022-06-30 12:12:12	2022-06-30 14:00:00
kitae	234567	Ki Tae Hwang	8210234567	2022-07-01 10:00:00	2022-07-01 10:00:20
ihjung	345678	In Hwan Jung	8234567891	2022-07-01 11:00:00	2022-07-01 12:00:00

Facility Table

id	name	quantity	openTime	closeTime	unit	price
iMac	MacOS	2	10:00	18:00	2:00	5
PC	Windows11	2	10:00	18:00	2:00	3

Booking Table

userId	bookingDate	facilityId	usingDate	canceled	contact
jmlee	2022-06-30	iMac01	2022-07-10 12:00	false	82101234567
kitae	2022-07-02	iMac01	2022-07-10 14:00	false	82101234568
kitae	2022-06-30	iMac01	2022-07-10 16:00	true	82101234568
ihjung	2022-07-01	PC01	2022-07-10 16:00	false	82101234568

Figure 4. Example database for User, Facility, Booking tables

4. Implementation of database in Firebase

The biggest difference between a booking system and a system that handles other data is that the booking contents must be delivered to the user in real time. For example, suppose someone is looking at a list of currently available schedules to book. Assume that F1, F2, and F3 are currently appearing in the booking available list. However, if someone else has booked F2 in that state, F2 should no longer appear in the available booking list at that moment.

A real-time database is a system in which data changes are sent in real time to all users viewing the data. This technology is based on push service. A typical real-time database product is Firebase provided by Google. There are three databases in Firebase: Realtime Database, Firestore, and Storage. Realtime Database is a database that informs data changes in real time. Firestore is very similar to Realtime Database, but provides more query capabilities than Realtime Database.

The biggest weakness of Firestore is that there are many restrictions on Query support. Searching for desired data from one table provides a function similar to SQL, but there is no function such as join operation of SQL. If you

need to query over two or more tables, read the data from the corresponding tables into the device and perform functions such as join operation. Firestore supports a hierarchical database structure to compensate for the shortcomings of such query.

Firestore natively supports hierarchical databases. Firestore consists of a series of Collections and Documents. In order to use Firebase efficiently, an optimal database structure is required for the designed database. As an example, consider Figure 5. In Figure 5-(a), one table can be converted into one collection. In this case, query on one collection can be performed easily like SQL. For example, if you want to find all booking details reserved with the ID 'kitae', simply pass the keyword 'kitae' to Firestore, and it will find and return the details to find.

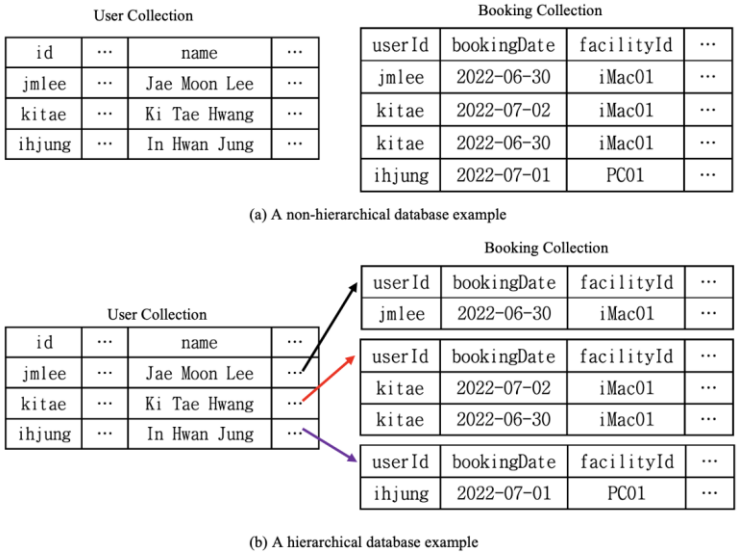

Figure 5. Various database configuration examples

However, if User and Booking collections are hierarchically connected as shown Figure 5-(b), it is efficient because only the Booking collection connected to 'Kitae' needs to be read. This is the advantage of hierarchical databases.

There are various configurations for building a hierarchical database from a given database. Figure 6 is the connection diagram of the previously designed database. Since this connection diagram appears as a graph rather than a tree, it cannot be applied as a hierarchical database as it is. Figure 7 is the decomposition of figure 6 into two possible trees. Figure 7-(a) is the decomposition of Figure 6 into two trees, and Figure 7-(b) describes Figure 6 into one tree. Each is organized as a tree and can be mapped directly into a hierarchical database. There are innumerable tree shapes that can be obtained from Figure 6, and Figure 7 shows two possible among them. This tree decomposition should be designed from the point of view of the lowest query cost considering the characteristics of the database. In this paper, it is decomposed and implemented as shown in Figure 7-(a). The reason for this design is that Booking occurs from the user's point of view. Therefore, it is because the search for the booking contents of the logged user occurs frequently. So there is a booking as a sub-collection of the user collection. In addition, The Schedule are periodically generated according to the facility. Therefore, the search for schedules for each facility will occur frequently. From this point of view, by designing the schedule to be a sub-collection of the facility collection, the search was made more efficient. Finally, the Rank is most likely one user will have a rank for all facilities. That is, a search will occur frequently for each user to determine which facility will have which rank. Conversely, we assume that it will not be queried for which users

have which ratings for a particular facility. Therefore, when designing as shown in Figure 7-(b), a lot of overlap occurs because facility data must be related to all ranks per user. So it would be more appropriate to break the link between rank and facility.

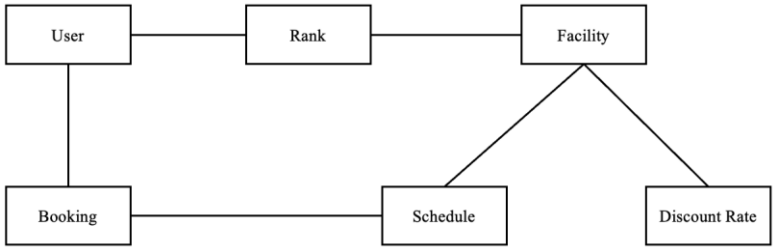

Figure 6. Connections for tables

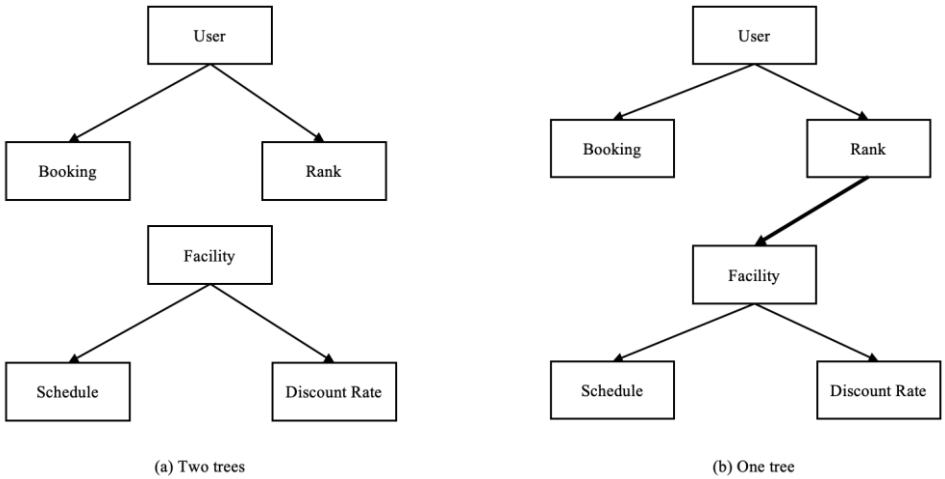

Figure 7. Example of partitioning into a hierarchical database

5. Conclusions

This paper is a database design and implementation for a reservation system for a small shared facility. A characteristic of the reservation system is that the changed reservation contents must be delivered to the relevant users in real time. In this paper, to reflect these characteristics, the database was designed and implemented to be suitable for Google's Firestore.

In this paper, the database was designed using ERM, and the database schema was derived from ERD, the result of ERM. In particular, Firestore was used to notify related users of database changes in real time. However, to implement a database in Firestore, which is a hierarchical database structure, it must be implemented to match the data access scenario. To this end, the user and facility tables, which are core data, were designed and implemented as the root of the hierarchical database to support efficient search.

6. Acknowledgements

This work was financially supported by Hansung University.

7. References

[1] Teng, P. (2014). Online room-booking system based on database. In Applied Mechanics and Materials, 513, 1748-1751.

[2] McTavish, C., & Sankaranarayanan, S. (2010). Intelligent agent based hotel search & booking system. In 2010 IEEE International Conference on Electro/Information Technology, 1-6.

[3] Sagar, S. V., Balakiruthiga, B., & Kumar, A. S. (2016). Novel vehicle booking system

using IOT. In 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 1-5.

[4] Mokar, M. A., Fageeri, S. O., & Fattoh, S. E. (2019). Using firebase cloud messaging to control mobile applications. In 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 1-5.

[5] Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. In 2011 6th international conference on pervasive computing and applications, 363-366.

[6] Badia, A. (2004). Entity-Relationship modeling revisited. Acm Sigmod Record, 33(1), 77-82.

[7] Omar, M. A. (2021). Performance Evaluation Of Supervised Machine Learning Classifiers For Mapping Natural Language Text To Entity Relationship Models. Journal of Pure & Applied Sciences, 20(1), 6-10.

[8] Haritsa, J. R., Carey, M. J., & Livny, M. (1992). Data access scheduling in firm real-time database systems. Real-Time Systems, 4(3), 203-241.

[9] Chai, C., Fan, J., Li, G., Wang, J., & Zheng, Y. (2019). Crowdsourcing database systems: Overview and challenges. In 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2052-2055.

[10] Varshney, H., Allahloh, A. S., & Sarfraz, M. (2019). IoT Based eHealth Management System Using Arduino and Google Cloud Firestore. In 2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON), 1-6.