

Databse Systems-Jiaji Tao- 2200521

by Jiaji Tao

Submission date: 05-Jan-2023 05:49PM (UTC+0000)

Submission ID: 193722924

File name: 40680_Jiaji_Tao_Databse_Systems-Jiaji_Tao-2200521_474126_1274478325.docx (230.83K)

Word count: 3948

Character count: 20344

Database Systems

Assignment Title: Abertawe Leisure Centre
Design & Implementation

Student name: Jiaji Tao

Student ID number: 2200521

Abstract

Database technology is now widely used in information systems, and powerful information systems rely heavily on database technology. The definition, manipulation, management and maintenance of data in information systems in all industries require database technology to support them, and the basis for these operations is the design of the database. This paper firstly introduces PostgreSQL and then takes ALC as the background, using PostgreSQL and then completing the design and implementation of the ALC database through requirements analysis, conceptual design, logical design and physical design.

Catalog

Abstract	2
1 PostgreSQL Overview	4
1.1 Analysis of PostgreSQL features	4
1.2 Fundamentals of PostgreSQL	4
2 Design of the database	6
2.1 Background and requirements of the project	6
2.2 Database design principles	6
2.3 System logic model.....	7
2.4 Entity-relationship diagram of the database	8
3 Database implementation.....	9
3.1 Triggers.....	9
3.1.1 Classification and role of triggers	9
3.1.2 Creation and application of triggers	9
3.2 Index	10
3.2.1 Classification of indexes	10
3.2.2 Selection and application of indexes	13
3.3 Transactions	14
3.3.1 Characteristics of transactions	14
3.3.2 Types of transactions	14
3.4 Normalised to 3NF	14
4 Conclusion	15
References.....	16

1 PostgreSQL Overview

The object-based relational database management system now known as PostgreSQL evolved from the Postgres package designed and developed at Berkeley. After two decades of development, PostgreSQL has become one of the most advanced open source database systems in the open source community, providing multi-version parallel control, support for almost all SQL components, and access to a very wide range of development language bindings. "Version 1 was released in June 1989 and was primarily available to external users. Version 2, released in June 1990, redesigned the rules system. Version 3, released in 1991, added support for multiple storage managers, improved the query executor and rewrote the rules system. Version 4.2 was released in 1993 and the Postgres project was discontinued.

1.1 Analysis of PostgreSQL features

PostgreSQL is a free database management system software, which has distinctive features compared to other commercial databases.

(1) Object-oriented PostgreSQL contains a number of object-oriented technical features, mainly classes and their inheritance mechanisms. For example, when creating a table, the user does not have to start from scratch, but simply inherits some properties from an existing table and then adds some new ones.

(2) Rich in data types, PostgreSQL supports not only traditional numeric, character and date data types, but also special data types such as point, line, polygon and other geometric data types. In addition, PostgreSQL also allows user-defined data types.

(3) Full support for SQL. PostgreSQL has the most comprehensive support for the SQL language of any free database management system, supporting not only SQL89 but also most of the features of SQL92.

(4) Integration with the Web. Through scripting languages such as PHP3 or Perl, PostgreSQL can be tightly integrated with web servers, providing users with a free, high-performance web solution. In addition, using ODBC and JDBC, PostgreSQL also supports traditional client/server applications.

(5) Large databases. While some database products suffer from a dramatic drop in performance on microcomputers when the size of the database exceeds 2GB, PostgreSQL maintains good performance, even when the size of the database reaches 100GB.

1.2 Fundamentals of PostgreSQL

The PostgreSQL database system environment consists of 3 main components: the

main Postmaster process, the client front-end application, and the Postgres back-end process. Within a PostgreSQL session, the Postmaster process, as the monitoring daemon, manages a certain collection of databases on the host, and its main task is to manage all database input requests and establish connections between the client and one or more database backends. If a client front-end application wants to access one of the databases in a node, Postmaster confirms that the request has legitimate access rights and it makes a library (function) call to start a new back-end service process, Postgres, and connects the front-end process to this new service process. Each client connection request has a unique Postgres process, and any operations on the database are initiated by the client. From this point on, the front-end process and the back-end service will no longer communicate through Postmaster but directly. Thus Postmaster is always running, waiting for requests, while the front and back-end processes are on and off.

The architecture of PostgreSQL is flexible in that it allows database administrators to use different technologies to manage multiple databases running on the same computer at the same time. In addition, each PostgreSQL installation provides a standard set of configuration files, which also conform to the PostgreSQL database format, often called `template1`, which can be modified by the user to suit a particular server and a particular application. This configuration information can also be copied to a new database, using the `createdb` program.

2 Design of the database

2.1 Background and requirements of the project

Abertawe Leisure Centre (ALU) is situated in the Manselton area of Swansea. They offer membership schemes to members of the general public. As the ALC continues to grow the centre's current database provision will not support their business model needs. Analyse the requirements of Abertawe Leisure Centre (ALC) and propose a database design by creating a database schema prototype through a suitable Entity Relationship Diagram. Create a PostgreSQL database (using SQL statements) that will enable ALC to manage member requirements.

2.2 Database design principles

Database design is a core part of program development, and standard database design principles and methods can effectively improve program development progress and efficiency.

(1) 3NF Third Paradigm

Paradigm, as one of the important concepts in database design. It refers to the process of creating a database that must follow certain guidelines, which are called paradigms. The third paradigm refers to the principle of adding attributes that do not depend on other non-primary attributes to a database table design that satisfies the first and second paradigms, in order to reduce the amount of data redundancy. It is broadly specified as follows.

A table can only describe a single entity, each value in the table appears only once in the entire database (except for foreign keys), each row in the table should be uniquely identified, and no information unrelated to the primary key is stored in the table. The third paradigm is generally considered to provide the best balance of performance, scalability and data integrity.

(2) Ensure that there are no computed fields in the table

The value of a computed field, which is calculated from other fields, can easily be overlooked when updating other fields while updating this computed field, resulting in data errors.

(3) Ensure that field values in tables are unique

If a field in a table has more than one value, it can be split into different fields, with each field having a unique value.

(4) Break down fields as much as possible

It is best to subdivide fields for tables with particularly detailed information, so that it is easier for users to edit the information and helps to ensure the integrity of the data.

2.3 System logic model

After observing and analyzing the ALC business system, the company needs.

- (1) the registration of members and the booking of courses.
- (2) the manipulation of membership data by managers.
- (3) An audit of all Insert, Update and Delete activities on the membership table.

The following modules are required to meet the above functions.

(1) Basic data maintenance module. The basic data maintenance module provides a means for members to enter, modify and maintain basic data. For example, the entry and modification of members' personal information, membership types and course selection information.

(2) Basic Business Module. In the basic business module, customers can fill in membership application forms, members can fill in course selection information forms, course booking methods, and staff are responsible for processing these forms. At the same time, staff feedback on the selection of courses to see whether to meet the course opening demand.

(3) database management module. In the ALC system, the information of members, managers and courses should be managed in a unified manner, and the membership joining, updating and termination should also be registered in detail.

(4) Information enquiry module. The information query module is mainly used to query related information, such as member query course information, course selection, etc.

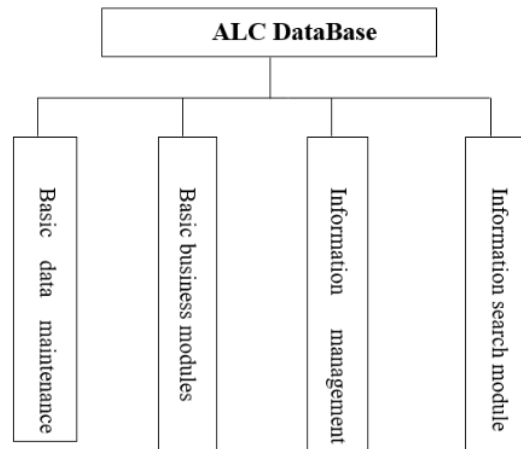


Fig. 1 Functional modules

2.4 Entity-relationship diagram of the database

After analysis, this system will use three basic tables: course table, membership table, and company table. The entity relationship diagram of the database is shown in Figure 2.

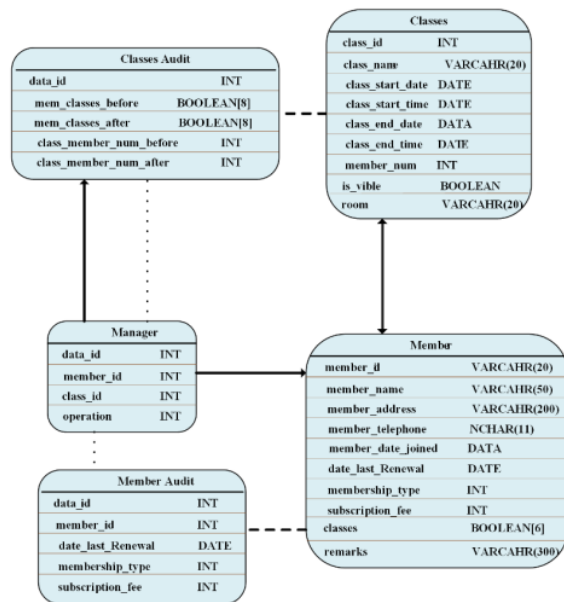


Figure 2 Database entity relationship diagram

3 Database implementation

3.1 Triggers

A trigger is a stored procedure that is automatically executed when an insert, update or delete operation is performed on a table. It is defined on a specific table, related to a table, cannot be called directly and automatically triggers the execution of a transaction.

3.1.1 Classification and role of triggers

According to the timing of trigger activation, it is divided into: post-trigger: a trigger that is triggered only after the execution of the INSERT, DELETE, UPDATE operation. Substitute trigger: instead of executing the insert, delete or update operation defined by the command, only the code defined in the trigger is executed. According to the event that causes the trigger, it is divided into: insert trigger: this type of trigger is triggered when there is an Insert operation on the table on which the trigger is created and automatically creates an Inserted table in memory, which holds a copy of the data at the time of the Insert operation. delete trigger: this type of trigger is triggered when there is a Delete operation on the table on which the trigger is created and automatically creates an Inserted table in memory. update trigger: this type of trigger is triggered when an Updated operation is performed on the table on which the trigger is created and automatically creates the Inserted and Deleted tables in memory, with the Deleted table holding a copy of the data before the modification. The Deleted table holds a copy of the data before the modification and the Inserted table holds a copy of the data after the modification.

The Inserted and Deleted tables are used for this purpose. One of the most powerful features of the trigger code is the ability to compare old and new versions of records, the most critical of which are the Inserted and Deleted temporary tables. When the trigger is triggered, the system automatically creates a Deleted table or an Inserted table in memory, both of which are read-only and do not allow modification, and both of which are automatically deleted when the trigger is executed. The Deleted table temporarily holds the rows of records before the deletion or update, and you can check from the Deleted table whether the data before the deletion or update meets the business requirements.

3.1.2 Creation and application of triggers

1. The higher the value dispersion of a field in the table is, the more suitable the field is to be selected as the keyword for the index.

Syntax for creating triggers:

```
CREATE TRIGGER trigger_name
ON table_name [WITH ENCRYPTION]
FOR [DELETE, INSERT, UPDATE]
AS
T-SQL 语句
GO
```

In the ALC Database's classes table, member_num indicates the number of people who have selected a course, and when the number of members in the course is greater than 6, the course can only be offered, and is_vialbe indicates whether the course can be offered. When we insert or update member_num after the trigger to automatically set is_vialbe to show whether the course is currently open or not.

3.2 Index

3.2.1 Classification of indexes

Common index structures can be divided into three categories according to how the index fields are organised: sequential indexes, hash indexes and tree indexes.

Sequential indexes are those where the index data is stored in the order of the field values. The most common type of sequential index is the Indexed Sequential Access Method (ISAM), which was originally proposed by IBM in 1964 as a file system developed by the IBM Corporation. Since data on disk is stored in blocks, creating a sequential index for such a storage structure requires block indexing. Suppose that a file of length a is divided into b blocks, which can be unequal in size. There is a certain order between the blocks, i.e. the largest data in the previous block is smaller than all the data in the next block. The data in each block can be stored out of order of size. The result is a structure that is unordered within blocks and ordered between blocks. An index node is then generated for each block, which contains an index field and a pointer field. The index field holds the largest element of the node's corresponding block, and the pointer field holds the block's location on disk. The index nodes are stored in order to form an index table. When searching the above index, the index table should be searched first to determine the range in which the target data is located, and then find the location of the corresponding data block based on this range. The data block is then retrieved according to the location. Finally, the target data is obtained by traversing the data blocks.

Hash indexes, ideally, are indexed in such a way that the target data is retrieved by

reading the disk only once, without the need for an additional comparison process. Hash indexing comes very close to this model. Hash indexing is done by mapping all the data into a number of Buckets based on a hash function. Ideally each piece of data will be mapped to a different Bucket. This way, when querying the data, it is only necessary to calculate the Hash function to get the location of the data. In practice it may not be possible to get the location of the data just by calculating the Hash function. It is entirely possible for different data to be mapped into the same Bucket. This phenomenon is called a Hash Conflict. Index keys usually tend to be Poisson distributed over a hash bucket. A Poisson distribution is not uniform. Therefore, the data is not evenly distributed in the Bucket. In one study, the Poisson distribution resulted in about 1/3 being empty buckets, 1/3 of the buckets containing one index key, and the other 1/3 containing two index keys. A large number of hash conflicts will affect index performance. Hash indexes solve the conflict problem mainly through the zip method.

Hash indexes have very obvious advantages, but they also have obvious disadvantages. Hash indexes do not allow for range queries. Data is scattered in Buckets, which are not contiguous with each other, and there is no obvious pattern to follow. Their location can only be obtained one by one by means of a Hash function. To retrieve data based on a range, all possible data in the range must be looked up one by one to see if the corresponding Bucket has data in that range. This process is obviously very inefficient. Hash indexes also have the obvious disadvantage that they cannot be queried using partial index keys. As the Hash function requires the full key to be present in order to calculate the correct value, providing only a partial index key will not give the correct value.

Tree indexes, there are two types of tree indexes B+ trees and LSM trees. Sequential indexes create an index on the data and can avoid the inconvenience of traversing the data. A B+ tree is a good structure for improving query efficiency, and introducing a B+ tree structure into an index can effectively reduce the time consumed by index queries.

B+ tree is a balanced multi-way lookup tree. Most database systems today use a B+ tree or a variation of it as their index structure. A B+ tree of order m has the following characteristics.

- (1) The root node has $[2, m]$ subtrees and non-leaf nodes other than the root node have $[\lceil 2/m \rceil, m]$ subtrees.
- (2) The number of subtrees of a non-leaf node is equal to the number of its

keywords.

(3) Pointers to non-leaf nodes: $P[1], P[2], \dots, P[m]$. where $P[1]$ points to a subtree with keyword less than or equal to $K[1]$ and the other $P[i]$ points to a subtree with keyword greater than $K[i-1]$ and less than or equal to $K[i]$.

(4) All leaf nodes are located at the same level and all keywords occur at the leaf nodes.

(5) All leaf nodes have a pointer to the next leaf node.

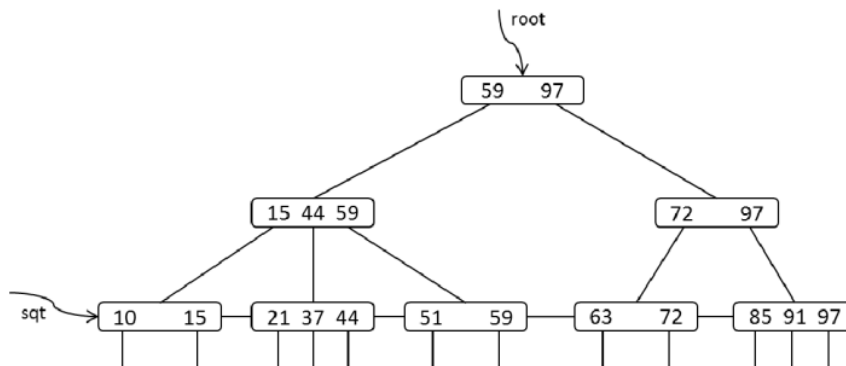


Figure 3 Example diagram of a B+ tree

Figure 3 shows a typical third-order B+ tree. As can be seen from the figure, a B+ tree typically has two head pointers, one of which points to the root node and the other to the first leaf node. By looking at Figure 3 it can be seen that each node of each third-order B+ tree has at most three keywords, each keyword corresponding to a pointer, and the range of subtrees this pointer points to is also specified by this keyword. The last layer of leaf nodes has pointers between them to the next smallest node. All keywords can be traversed sequentially by means of a head pointer to the first leaf node and a pointer between leaf nodes.

A query of the B+ tree should start at the root node and work its way down until the corresponding keyword is found at a leaf node. Each subtree selection is matched against the range of keywords corresponding to the current node. When performing an insert operation on a B+ tree, the first step is to find the leaf node where the data should be inserted. This process is the same as the query process for the B+ tree. Then the number of keywords in the leaf node is checked for m . If the number of keywords is less than m , then the leaf node is inserted directly. If the number of keywords is equal to m , the leaf node should be split in two. Also insert the smallest keyword in the second

node into the parent node. If the number of keywords in the parent node exceeds m , then the parent node should also be split.

To delete a B+ tree, you should first find the leaf node where the data to be deleted is located and delete it. If so, borrow the largest (or smallest) keyword from its sibling node so that the two nodes have the same number of keywords. If the sibling node also does not have enough keywords to borrow, then a new node is created by merging that node, the sibling node and one of the keywords from the parent node.

The LSM tree is an improvement on the B+ tree. The process of inserting data into the B+ tree may involve inserting several consecutive data into different leaf nodes. These leaf nodes are most likely to be scattered in different areas of the disk. This can cause random I/O problems to the disk. This invariably increases the seek time of the disk and makes the insertion process more time consuming. To solve this problem, P. O'Neil proposed the LSM tree in his 1996 article "The log-structured merge-tree", which is most commonly used in applications where the frequency of modifications is much higher than the frequency of queries. A significant increase in write performance is achieved by sacrificing system read performance. The time consuming operations are done in memory, while the disk simply stores the data. The data is first stored sequentially in memory, and then the batches of data are written to disk. This reduces the seek time on disk and gives a high write performance.

3.2.2 Selection and application of indexes

B-Tree indexes are mainly used for equals and range queries, especially when the index column contains the operators "<, >, =, <=, >=" as a query condition, and are considered by the PostgreSQL query planner. By analysing the requirements of ALC, it is clear that the course will only be offered if the number of members in the course is greater than or equal to 6, so the data structure used in creating the index is a B-Tree. The diagram below shows some of the indexes for this database.

```
/* Add index to class id to improve search speed */
CREATE INDEX Ind_class_id ON classes(class_id ASC);
```

Figure 4

```
/* Index SQL*/
-- CREATE INDEX Ind_Member_id ON member (member_id ASC);
CREATE INDEX Ind_Member_name ON member (member_name ASC);
```

Figure 5

3.3 Transactions

3.3.1 Characteristics of transactions

The transaction, as a single logical unit of work, has 4 characteristics.

Atomicity: a transaction is a complete operation. The steps of the transaction are indivisible (atomic); either they are all executed or none of them are executed.

Consistency: the data must be in a consistent state when the transaction completes or fails.

Isolation: the transactional operations performed on the data are isolated from each other, which indicates that the transaction must be independent and that it should not depend on or affect other transactions in any way.

Durability: After a transaction completes, its changes to the database are maintained permanently and the transaction log is able to maintain the permanence of the transaction.

3.3.2 Types of transactions

Transactions in sql server databases are divided into three categories: explicit transactions, implicit transactions, and auto-commit transactions.

Explicit transactions: specify the beginning of a transaction with a begin tran and end with a commit tran or rollback tran statement. This is the most common type of transaction.

Implicit transaction: Set the implicit transaction mode to on by setting the set implicit transactions on statement, and the next statement automatically starts a new transaction. When that transaction completes, then the next T-SQL statement will start a new transaction again.

Auto-commit transaction: This is SQL Server's default mode, which treats each individual T-SQL statement as a transaction that is automatically committed if it executes successfully, or rolled back if there is an error.

3.4 Normalised to 3NF

The transformed relational schema will be optimised to eventually reach the third paradigm.

In order to improve the performance of the database application system, the relational schema should also be modified and adapted appropriately to the needs of the

application, which is the optimisation of the data model. The optimisation results are shown in the entity-relationship diagram above.

4 Conclusion

With the rapid development of technology and computer technology, the use of computers to replace manual management is gradually becoming a reality, and society's requirements for the overall quality of contemporary personnel are becoming higher and higher. In order for us to better adapt to the development of society and the needs of modern enterprises, the school has arranged this course in a timely manner, and the teacher has provided us with a series of comprehensive and applicable coursework.

After a semester of study, I have gained a good understanding of the subject of database application course design. This coursework is based on the design and implementation of an ALC database in the context of ALC using PostgreSQL and then through requirements analysis, conceptual design, logical design and physical design. Through this visual programming course design, I was able to consolidate the theoretical knowledge I had learnt in class, further understand the basic ideas of object-oriented programming and develop my knowledge.

References

- [1] Shen Shu. Research on NoSQL database technology and its application[D]. Nanjing University of Information Engineering, 2012.
- [2] Liu Y. Database design skills and design specifications [J]. Information Technology and Informatization, 2014(5): 250-252
- [3] Lin Shuze. Research on optimization of physical design of database information system [D]. Tianjin: Tianjin University, 2012
- [4] Hou Xiaoling,Feng Lili,Qu Xiaohong. The important role and principles of computer software database design[J]. Electronic Technology and Software Engineering, 2020, No.172(02): 192-193.
- [5] Big data[DB/OL]. http://en.wikipedia.org/wiki/Big_data.

FINAL GRADE

GENERAL COMMENTS

Instructor

Jiaji Tao

62/100

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills	10	
Mark will be awarded on your references to the literature to discuss the importance of the concepts of rollback, commit, functions, triggers, procedures, transactions and indexes. Although a very good and in-depth discussion, there was a lack of engagement with literature to reinforce your statements by providing evidence.	4	
2. Knowledge and Understanding Skills	20	
Guidance specific to this assessment: Suitable		

<i>Entity Relationship design must be employed and justified.</i> A suitable ERD was provided and a comprehensive analysis of the business requirements to justify the ERD design.	18	
3. Cognitive and Intellectual Skills	20	
<i>Guidance specific to this assessment:</i> <i>Marks will be based on your ability to analyse the case study problem scenario and propose an appropriate, normalised to 3NF, database schema.</i> The normalisation was a little less than I would have liked, an example would be to provide a membership type table or a class type to further reduce the repeated data.	10	
4. Practical Application Skills	30	
<i>Guidance specific to this assessment:</i> Marks will be awarded based on your ability to build and deploy the database for the ALC case study evidenced by your submission in SQL File 1 (Task 1, item 1).Note: You may be required to provide a demonstration of the implemented database. <i>Marks will be awarded based on your ability to analyse the case study problem scenario and provide an appropriate practical solution and testing for the ALC database.</i>	20	

normalised to 3NF; appropriate attributes to demonstrate the design structure for assignment purposes; Primary key for each table. Your database schema should feature at least one composite primary key; Referential integrity constraints (foreign keys); 2/3 check constraints. A few Functions; Triggers with an auditing feature; No stored Procedures; No Transactions; 1/3 Indexes.		
5. Transferable Skills for Life and Professional Practice	20	
<i>Guidance specific to this assessment:</i> Marks will be awarded based on the written and presentational quality of the report. This will be assessed based on the: · Readability of the report.	10	

· Written quality, including spelling and grammar and the use of appropriate formal language. · Structure and presentation of the report. · Critical evaluation of the selected database methods, tools and techniques and their deployment in a multi-user database environment. Overall a good report, a good layout and coherent.		
---	--	--

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11
PAGE 12
PAGE 13

