

Database Systems Course

Report 1

by Yuxiang Gao

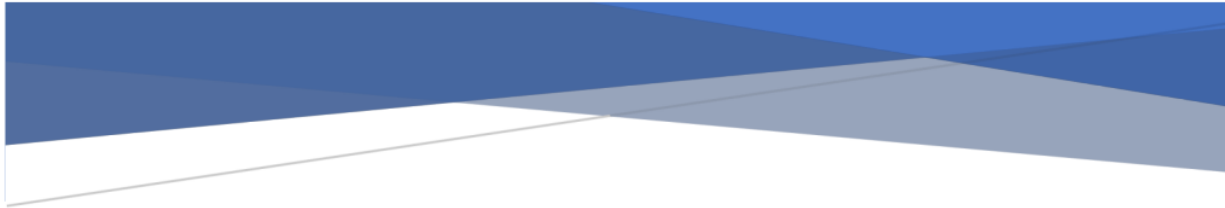
Submission date: 05-Jan-2023 10:41AM (UTC+0000)

Submission ID: 193669730

File name: 39380_Yuxiang_Gao_Database_Systems_Course_Report_1_474126_1604575108.docx (127.21K)

Word count: 4134

Character count: 20653



DATABASE SYSTEMS

Course Report 1

Gao Yuxiang 2200441
ACCA7002 DATABASE SYSTEMS

Content

Summary	2
1.0 Current status of research on various security frameworks or models.....	3
1.1 ITIL The industry standard for process-centric IT management	3
1.2 ISO 27000- the international standard for process-centric management ..	3
1.3 BS7799 - Security-Centric IT Management Standard	3
1.4 COBIT - IT Management Standards for IT Auditing Error! Bookmark not defined.	
2.0 ISO27001-based information security system construction practice.....	5
2.1 Information security risk assessment practice..... Error! Bookmark not defined.	
2.2 Information security project planning process..... Error! Bookmark not defined.	
2.3 Design and application of information security system effectiveness measurement method	Error! Bookmark not defined.
2.4 System operation and implementation effect evaluation..Error! Bookmark not defined.	
3.0 Conclusion.....	10
Reference	20

Summary

This report focuses on how to manage ALU data securely and effectively while building libraries and tables that meet the actual business needs as well.

1.0 ALC Solutions

1.1 Performance Tuning

Tuning 1: We use the trigger to write the staff entry staff information, when the VIP customer information is entered, the trigger will automatically enter the VIP customer into the staff service table, and the program will automatically specify the staff service corresponding to the VIP customer.

Tuning 2: The second use of the trigger is when the VIP customer joins the corresponding class, each class will correspond to a room, and when a new class is generated, the trigger will generate a room for each new class. When the class expires, the trigger will spontaneously delete the room information, to better provide the next new class. If the number of rooms is less than the number of classes, the class insertion statement will be rolled back and the corresponding log will be prompted.

1.2 Trading

The transaction is the entry of customer information, i.e. the operation of data entry to the VIP table.

Determining which vip the customer has purchased by the type of vip_info.

The VIP customer information is entered through the clerk, and the table employee performs the data insertion.

Immediately afterward, the customer information is paired with the class table, and the customer id is inserted into the class table as an extra.

Finally, the staff will bind the corresponding room number to the class.

1.3 Evaluation of design and implementation sites

We created five tables to cope with the whole system of ALU.

The VIP customer table contains vip customer id, VIP customer name, yip customer join time, vip customer mobile number, and VIP customer update time.

The VIP customer category table contains, the VIP customer category, and VIP customer id.

The class table contains class id, class category, room number, and staff number.

The staff table contains, staff id, staff name, staff job, staff service vip client id.

The room table contains the room number, start date, borrow date, start time, and end time.

2.0 Discuss the importance of rollback

Undo log is one of the more important transaction logs in MySQL. As the name implies, undo log is a kind of log used to undo the rollback. Before the transaction is committed, MySQL will record the data before the update to the undo log file, when the transaction rolls back or when the database crashes, you can use undo log to roll back.

In MySQL, the undo log serves two main purposes.

1、 Provide rollback operation [undo log achieves atomicity of transaction]

If for some reason the transaction is rolled back, then MySQL will perform a rollback operation and use the undo log to restore the data to the state before the transaction started.

For example, we execute the following delete statement: delete from user where id = 1; delete from user where id = 1; delete from user where id = 1(Thomas et al. 2009)

```
delete from user where id = 1;
```

Then the undo log will record a corresponding insert statement [reverse operation statement] to ensure that when the transaction is rolled back, the data will be restored.

Another example is to execute an update statement: update user set name

```
update user set name = "Li Si" where id = 1; --- modify the previous name = Zhang San
```

At this point undo log will record an opposite update statement, as follows.

```
update user set name = "Zhang San" where id = 1;
```

If there is an exception to this modification, you can use the undo log to achieve a rollback operation to ensure the consistency of the transaction.

2、Provide multiple version control (MVCC) [undo log to achieve multiple version concurrency control (MVCC)]

MVCC, which is multiple version control. In the MySQL database InnoDB storage engine, undo log is used to achieve multiple version concurrency control (MVCC). When a row read is locked by another transaction, it can analyze from the undo log what the previous version of data in that row record is, so that the user can read the data before the current transaction operation [Snapshot Read].

The following explains what a snapshot read is, and the corresponding one is --- current read.

Snapshot Read.

The data read by SQL is the snapshot version [visible version], that is, the historical version, without locking, and the ordinary SELECT is the snapshot read.

The data read by SQL is the latest version. The lock mechanism is used to ensure that the read data cannot be modified by other transactions UPDATE, DELETE, INSERT, SELECT ... LOCK IN SHARE MODE, SELECT ... FOR UPDATE are all current reads.

The storage of undo log is implemented by the InnoDB storage engine, and the data is stored in InnoDB data files. In the InnoDB storage engine, undo log is stored by segmentation. The rollback segment is called the rollback segment, and each rollback segment has 1024 undo log segments. Before MySQL 5.5, only 1 rollback segment was supported, that is, only 1024 undo operations could be logged. After

MySQL 5.5, it can support 128 rollback segments from resg slot0 - resg slot127, each resg slot, that is, each rollback segment, consists of 1024 undo segments, that is, a total of $128 * 1024$ undo operations can be recorded.

The undo log not only stores the records before the data update, but also records the RowID, transaction ID, and rollback pointer. The transaction ID is incremented each time, and the rollback pointer is NULL for the first time if it is an insert statement, and the rollback pointer of the undo log after the second update will point to the undo log just now, and so on, forming a rollback chain of the undo log, which is convenient to find the historical version of the record.

Before updating the data, MySQL will generate the undo log in advance, and when the transaction is committed, the undo log will not be deleted immediately because the rollback operation may be needed later, and the data will be read from the cache when the rollback operation is executed. undo log deletion is processed by recycling through the background purge thread.

1、Transaction A performs an update operation, at this time, the transaction has not been committed, the data will be backed up to the corresponding undo buffer, then the undo buffer is persisted to the undo log file in the disk, at this time, the undo log saves the operation log before it is committed, then the operation data, that is, the data of the Teacher table is persisted to the InnoDB data file. to InnoDB's data file IBD.

2. At this time, transaction B performs query operations and reads directly from the undo buffer cache, at which time transaction A has not yet committed the transaction, and if it wants to rollback the transaction, it does not read the disk, but first reads directly from the undo buffer cache.

A simplified process for implementing atomic and persistent transactions with undo log.

Suppose there are two data A and B with values 1,2 respectively.

A. Transaction starts

B. Record A=1 in undo log

C. modify A=3

D. Record B=2 in the undo log

E. Modify B=4

F. Write undo log to disk -----undo log persistence

G. Write data to disk ----- data persistence

H. Transaction commit ----- commit transaction

The reason why atomicity and persistence are guaranteed at the same time is because of the following features.

The undo log is recorded before updating the data.

To ensure persistence, the data must be written to disk before the transaction is committed, and as soon as the transaction is successfully committed, the data must have been persisted to disk.

The undo log must be persisted to disk before the data. If a system crash occurs between G and H, the undo log is complete and can be used to roll back.

If a system crash occurs between A - F, the data on disk remains in the same state as it was before the transaction started because the data is not persisted to disk.

Disadvantage: Writing data and undoing logs to disk before each transaction commits causes a lot of disk IO, so performance is poor. If you can cache the data for some time, you can reduce IO to improve performance, but then you lose the persistence of the transaction.

The undo log belongs to the logical log, the redo is the physical log, and the so-called logical log is undone log is to record an operation process, will not physically delete undo log, sql execution delete or update operation will record a undo log.

`innodb_undo_directory`

Specifies the directory where the undo log is stored, the default value is `./`.

`innodb_undo_logs`

After MySQL 5.6, you can customize the number of rollback segments with this parameter, the default value is 128.

`innodb_undo_tablespaces`

Specifies how many tablespaces the undo log is distributed to. The default value is 0, which means all of them are written in one file. It is not recommended to change it to a non-zero value, we can just use the default value.

3.0 Conclusion

3.1 Commit

Commit, for oracle, is a commit action, that is, you submit your changes to the Oracle database to the database, tell the database, you have determined to modify these things, and then the database saw your commit, the content of the modified, in the commit inside it, involves a concept of oracle database, that is, transaction, the characteristics of the transaction, there are four ACIDATOMIC: atomicity, that is, this is the smallest unit, there is a no smaller unit than this either commit. The transaction, the characteristics of it, there are four ACIDATOMIC: atomicity, that is, this is the smallest unit, there is no smaller unit than this either commit. Tuning 2: The second use of the trigger is when the VIP customer joins the corresponding class, each class will correspond to a room, and when a new class is generated, the trigger will generate a room for each new class. When the class expires, the trigger will spontaneously delete the room information, to better provide the next new class. If the number of rooms is less than the number of classes, the class insertion statement will be rolled back and the corresponding log will be prompted.

3.2 Function

Not only does MySQL implement these explanations, but the names are identical. Perhaps this is not surprising, since these are the basic numeric functions of the database product and programming language.

Since these functions are identical in both DBMSs, it makes no sense to retell how they are used. Instead, we will move on to explore other useful numeric functions in MySQL.

3.3 AVG

You probably already know what the average or arithmetic mean is. You may even know that it is calculated by adding up all the values in a dataset and dividing that result by the number of data points in that dataset. So if we have five numbers, say 4, 5, 6, 5, 3, we would calculate their average as follows

$$(4 + 5 + 6 + 5 + 3) / 5 = 4.6$$

This is a simple way to handle a small number of values, but what happens when you have 10,000 rows? The answer of course is to use MySQL's built-in AVG function (which, by the way, is named the same in SQL Server). All we need to do is to provide it with a numeric expression and it will return its average value. Here is its simple syntax.

`AVG(expression)`

Most of the time, you will have entered a column name whose average value you want to calculate. For example, here is a query that gives us the average length of all movies in the Sakila sample database.

GROUP BY breaks down the values by `category_id` so that the calculated average is based on each movie type, i.e. "Action", "Drama", etc.

Bypassing the results of the AVG function to ROUND, we can omit some extra decimal places.

In this article, we used Navicat 16 for MySQL as our database client to introduce some useful numeric functions in MySQL, including AVG, MIN, and MAX.

3.4 Trigger

A trigger is an event that triggers an operation. These events include INSERT statements, UPDATE statements and DELETE statements. When the database system executes these events, it will activate the trigger to perform the corresponding operation. I. Create a trigger

The trigger is triggered by events such as insert, update and delete to trigger a specific operation. When the trigger conditions of the trigger are met, the database system will execute the program statements defined in the trigger. Doing so ensures consistency between certain operations. For example, when a student's information is added to the student table, the total number of students must be changed at the same time. You can create a trigger here to execute an operation to calculate the total number of students every time a student record is added. This will ensure that the total number of students is the same as the number of records each time a student record is added. A trigger may have only one execution statement or multiple execution statements.

1、 Create a trigger with only one execution statement

In MySQL, the basic form of creating a trigger with only one execution statement is as follows.

create trigger name before | after trigger event on table name for each row
Execute statement (Kulkarni et al. 1999)

where the trigger name parameter refers to the name of the trigger to be created; the before and after parameters specify the trigger execution time, "before" means execute the triggering statement before the trigger event, after means execute the triggering statement after the trigger event; the "trigger event" parameter refers to the

trigger conditions, including insert, update and delete; "table name" parameter refers to the name of the table that triggers the event operation; for each row means that any operation on a row satisfying the trigger event will trigger the trigger. The "execute statement" parameter refers to the procedure to be executed after the trigger is triggered.

The following creates a trigger dept_trig1 triggered by the insert.

The result shows that the trigger dept_trig1 has been created successfully. When an insert operation is executed into the department table, the database system inserts the current time into the trigger_time table before the insert statement is executed. The following insert a row into the department table and then check whether the trigger_time table executes the insert operation.

The execution result shows that the current system time is inserted in the trigger_time when the insert operation is executed to the department table. From this example, we can see that the insert successfully triggered the trigger.

2. Creating a Trigger with Multiple Execution Statements

In MySQL, there may be more than one execution statement triggered by a trigger. The basic form of creating a trigger with multiple execution statements is as follows.

```
create trigger name before | after trigger event on table name for each row begin  
to Execute statement list end
```

The "list of executable statements" parameter directly between begin and end indicates the contents of multiple executable statements to be executed. The different execution statements are separated by semicolons.

Note: In general, MySQL defaults to ";" as the end execution statement. You need to use ";" in the process of creating triggers. To solve this problem, you can use the delimiter statement. Such as "delimiter&&", you can change the ending symbol to "&&". When the trigger is created, you can use the command "delimiter;" to turn the end symbol into ";".

The following code creates a trigger dept_trig2 with multiple execution statements triggered by delete.

The execution result shows that the trigger is created successfully. When the delete operation is executed in the department table, two rows will be inserted in the trigger_time table.

The execution result shows that two rows are inserted in the trigger_time when the delete operation is executed to the department table. From this example, you can see that the trigger can execute multiple execution statements at the same time.

Note: In MySQL, only one trigger can be created for a table with the same trigger condition at the same trigger time. For example, in the department table, there can be only one trigger with trigger event insert and trigger time after. However, it is possible to define a trigger with a trigger event BEFORE. If the insert statement is executed in that table, then this trigger will be executed automatically.

Second, view the trigger

To view triggers means to view the definition, status, and syntax of the existing triggers in the database. The method to view triggers includes show triggers statements and querying the triggers table under the information_schema database.

1、show triggers statement to view trigger information

In MySQL, you can execute the show triggers statement to view the basic information of triggers. The basic form is as follows.

```
show triggers;
```

Copy after login

The result of the following show triggers statement is as follows.

The result shows the basic information of all triggers. Since there are only two triggers in the database for the time being, only the basic information about these two triggers is displayed.

Tips: show triggers statement cannot query the specified triggers, the statement can only query the information of all triggers. If there are many triggers in the database system, a lot of information will be displayed. This is not convenient to find the information about the trigger you need. Therefore, when there are few triggers, you can choose to show the triggers statement.

2、Check the trigger information in the triggers table

In MySQL, the definitions of all triggers are in the triggers table under the information_schema database. Query the triggers table to see the details of all triggers in the database. The query statement is as follows.

```
select * from information_schema.triggers;
```

Copy after login

where "*" means query all the columns; "information_schema.triggers" means the triggers table under the information_schema database. .

The following is a select statement to query the information in the triggers table.

The result shows the details of all the triggers. Also, this method can query the details of a specified trigger. The basic form of the statement is as follows.

```
select * from information_schema.triggers where trigger_name='trigger_name';
```

where the "trigger_name" parameter refers to the name of the trigger to be viewed, which needs to be enclosed in single quotes.

The following is the information of trigger dept_trig1 using a select statement.

The result shows the details of the trigger dept_trig1. This way can query the specified trigger, which is more convenient and flexible to use.

Tip: The information of all triggers is stored in the triggers table under the information_schema database, and you can use the select statement to query the information of triggers from the triggers table. If there are more triggers in the database, then there will be more records in the triggers table. When using the select statement, it is best to query through the trigger_name field.

III. Use of Triggers

In MySQL, the order of trigger execution is before the trigger, table operations (insert, update and delete), and after the trigger. The following is a sample to demonstrate the order of execution of these three.

The following two triggers, before insert and after insert, are created on the department table. When inserting data into the department table, observe the trigger order of these two triggers. The code to create the triggers is as follows.

```
//create before insert trigger create trigger before_insert before insert on  
department for each row insert into trigger_test values(null, "before_insert"); // create
```

after insert trigger create trigger after_insert after insert on department for each row
insert into trigger_test values(null, "after_insert"); (Yang and Zheng 2010)

After the triggers are created, insert a row into the department table. The code is executed as follows.

The execution result shows that the record was inserted successfully. Now you can view the records in the trigger_test table. Code execution is as follows: IV. Delete Trigger

The drop trigger statement is used in MySQL to delete a trigger that already exists in the database. Its basic form is as follows.

```
drop trigger Trigger Name;
```

where the "trigger name" parameter refers to the name of the trigger to be deleted. If only the trigger name is specified, the database system will look for the trigger under the current database. If it is found, the deletion will be executed. If you specify a database, the database system will go to the specified database to find the trigger. For example, job.worker_trig means the trigger worker_trig under the job database.

Note: If a trigger is no longer needed, be sure to delete it. If the trigger is not deleted, the execution statement in the trigger will be executed every time the trigger event is executed. Execution statements perform certain operations on the data in the database, which can cause changes to the data. Therefore, it is important to delete the trigger that is not needed.(Krogh and Okuno 2017)

The following is the drop trigger statement executed to delete the trigger dept_trig1.

The result shows that the deletion was successful. To determine if the trigger is deleted, you can query the information of 'dept_trig1' with the select statement. the select statement is executed as follows.

The execution result shows that the record does not exist. This indicates that the trigger dept_trig1 has been deleted successfully.

3.5 Performance Tuning

Stored procedure A set of SQL statements that perform a specific function, stored in the database, and called again without compilation after the first compilation (more efficient).(Zhu et al. 2019)

3.6 Affairs

A transaction is a single unit of one or more sql statements, in which each sql statement is interdependent. The entire individual unit exists as an indivisible whole, similar to an atom in physics (an indivisible minimum unit).

In layman's terms, a transaction is a whole in which everything is either executed successfully or unsuccessfully. There cannot be a situation where part of the execution is successful and part of it is unsuccessful.

3.7 Index

The read-to-write ratio of a website is typically 10:1, so queries will always be the bulk of the database.

If a site has only tens of thousands of data, then there is no need for indexing at all. But once it reaches a million or more, without indexes, every query will cause seconds of waiting! This is something that users should not tolerate.

To speed up the search speed, the database will generally have an index function, that is, similar to the index of the book number in the library.

The index in the database is a special file that stores the reference to all records in a database table.

And the implementation of indexes in databases is much more complicated than building indexes for books. (Generally do not need to know in depth)

The purpose of creating indexes in databases is: so that you do not have to traverse every piece of information to find the results. And to achieve this effect, it is obvious that trees in data structures are specialized to do this kind of thing.

Reference

- KROGH, J. W. AND M. OKUNO *Pro MySQL NDB Cluster*. Edition ed.: Springer, 2017. ISBN 1484229827.
- KULKARNI, K., N. MATTOS AND R. COCHRANE. Active database features in SQL3. In *Active Rules in Database Systems*. Springer, 1999, p. 197-219.
- THOMAS, S., L. WILLIAMS AND T. XIE On automated prepared statement generation to remove SQL injection vulnerabilities. *Information and Software Technology*, 2009, 51(3), 589-598.
- YANG, G. AND Y. ZHENG. The design and realization of workflow engine based on the relational data model. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*. IEEE, 2010, vol. 8, p. V8-408-V408-411.
- ZHU, B., R. MENG, Z. ZHANG AND W. CHAN. ConRS: A Requests Scheduling Framework for Increasing Concurrency Degree of Server Programs. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2019, vol. 1, p. 169-178.

Database Systems Course Report 1

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

40/100

Instructor

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills When writing about Performance Tuning, Triggers etc, do so with reference to your database and support with citations to the literature. Your discussion talks about MySQL and Oracle, again keep to the database engine used in the assignment. Engagement with a very limited range of sources. Also there are no citations in the main body of the text.	10	3
2. Knowledge and Understanding Skills Knowledge is accurate and reasonably detailed. You have a Entity Relationship Diagram that is missing expected entities for Qualifications and Room Type. Also	20	9

some of the entities such as VIP (Member) have a number of missing attributes. Where are you storing the payment details for membership?		
3. Cognitive and Intellectual Skills	20	8
From your analysis the case study problem scenario (ERD) you have proposed a relatively coherent database schema. With VIP_ID in the Employee table I am not convinced that the database has been normalised to 3NF. You have demonstrated that you can deal with complex issues but not fully systematically or creatively.		
4. Practical Application Skills	30	12
Mainly consistent application of theory to practice. You have taken the ERD and produced a database structure with Primary and Foreign Keys. However, the database is missing key tables because of the incomplete ERD. I cannot see where you have created and used indexes, procedures and functions etc.. The database has been tested with some SQL data manipulations statements.		

5. Transferable Skills for Life and Professional Practice	20	8
Mostly coherent, organised work, in a suitable structure and is for the most part clearly expressed. When discussing Performance Tuning, Triggers etc, do so with reference to your database and support with citations to the literature.		

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)

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

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21
