**Database Design and Performance Test**

**Table of Contents**

**1. Introduction**

**The invention of the computer has brought tremendous changes to human life and work. If you don't use computers in management, it's impossible. Invoice database allows a business to easily identify and refer to individual transactions with clients. While visiting a web page, if it does not load quickly, customers will leave. If an application’s working hours are too long, you will simply close it. If the application makes users struggle to meet their needs, they will go elsewhere. We usually don't think about performance issues until we need them. I find that premature optimization is not worthwhile, but that does not mean it is unimportant. If you can see any issues at the very beginning of the workflow, then applying fixes will be cheaper and easier. This paper focuses on the database design and performance test and optimization of the database.**

**2. Related Work**

**Many people have related work: 1. The B tree or T tree are applied to the database by Zhiqiang Hu, improving the index performance, how to pick these two algorithms to optimize the query, insert and delete performance of database indexes in a timely manner. 2. A tool is proposed by Rinta Kridalukmana that is used to have a test of the logical model for an achieved database as opposed to the one that creates its original requirements. An XML document is used by the proposed tool to represent the request and a JDBC approach is used to carry the logical model of the implemented database. 3. T.J. Moore describe a test process modeling tool used in the process of both the design and manufacturing to identify the most affordable processes to perform testing with no sacrifice in test coverage or productivity, including eliminating experimental process steps and optimizing test proccess.**

**3.** **Design of Database Relational Model**

**Before building the database, we must first design the database relational model and make it meet the requirements of the third normal form. The field customerID is the primary key in CUSTOMER and also exists in INVOICE as a foreign key, where it connects each invoice to its customer.**

**3.1 First Normal Form Design**

**First, we have to consider whether the structure of Table 1 meets the requirements of the 1NF. First Normal Form (1NF) is a characteristic of relationships in relational databases.** **In this database, each column of data is indivisible, there are no multiple values in the same column, and there are no duplicate attributes, and each row contains information about only one instance.**

**3.2 Second Normal Form Design**

**In this database, each table can be uniquely distinguishable. Each table has a primary key and the other elements correspond to the primary key one by one, that is, the other elements depend on the primary key. But if I want to update the data in one column then the data in other columns have to change with it. Suppose you want to delete a value in a column, then other columns on that row will also be deleted, which will cause a database abnormaly. Data duplication and redundancy: the same customer bought the product m times, the information on the customer table is duplicated m-1 times.**

**3.3 Third Normal Form Design**

**In this database, all data elements are independent of each other, and there is no other functional relationship. And there is no dependency of some entities on other non-primary key data entities. In conclusion, Our database meets the three normal form designs.**

**4、Database and Research Object**

**First, create a dataset named ‘invoiceappdb’ which has 6 tables in the database: Customer, PaymentTable, PaymentTable, Users, InvoiceTable, Invoiceitem, ProductTable.**

**Customer (CustomerID，Name, Address, PanNo, EmailID, IncludeTaxAmount, PlaceOfSupply CustomerName, CompanyName, GetRegisterNo, MobilePhone, Opening Balance, CustomerForm)**

**PaymentTable (CustomerID, AmountReceived, PaymentDate, PaymentID, PaymentMode, TransactionID, FinancialYear)**

**Users (UserID, EmailID, Name, Password, PhoneNumber, Roles)**

**InvoiceTable (InvoiceID, InvoiceDate, Subtotal, TaxAmount, Address, CustomerID, PanNo, PlaceOfSupply, InvoiceNumber, InvoiceDate, RegisterNo, VehicleNo, CreatedTime, LastModifiedTime)**

**Invoiceitem (InvoiceID, ProductID, Quantity, TaxAmount, ItemTotal, ItemPrice, GstTaxRate, ItemOrder, LineitemID)**

**ProductTable (ProductID, Unit, Price, GstPercentage, ProductType, ItemDesc, StockKeepingUnit, Status)**

**5. Security of Database**

**User table: For example, I am the owner of this supermarket, After you purchase my products, I will give you an invoice (bill), only I can bill you, so I am the owner of this invoice, I am the person who writes the bill. However, only those who meet the conditions in the user table can make a bill, if anyone can make a bill, the supermarket will be chaotic. Therefore, we specify in the user table: you must meet the criteria entered in the database, UserID, EmailID, Name, Password, PhoneNumber, Role you can open this invoice bill.**

**6. Database Function**

**The company's customer and order and user management system includes N modules, including Invoice module, Payment module, Product module, and Users module.**

**3.1 Invoice**

**The main goal of an invoice is to generate a sale record for the business and its customers. An invoice provides written verification of the purchase contract between your business and your customers. Invoices define the rules for your payment requirements and allow you to get paid for your services faster.**

**3.2 Payment**

**The payment table records each payment made by a customer, with information such as the amount and the products being paid for (when applicable) and the date customers purchased and their financial information.**

**3.3 Product**

**Specific information about items is stored by the product table. We propose to make it available to our customers. It defines a product database as a gathering of product-related data, deposited and managed digitally. Product table also include status of the product. The status indicates whether the product has been recently changed, or if it has been deployed on your store, or if has been retired from your catalog.**

**3.4 User**

**A list of users who have the right to get into the MariaDB server and their overall priviledge are stored by the USER table, The table itself is queryable and whilst it can be updated directly. To use GRANT and CREATE USER to add users and permissions is preferrable.**

1. **Security and integrity demands.**
2. **Privacy begins with the view mechanism, where different parties can only access their own priviledged view. Then through the user authorization mechanism, the user level is identified through user login, and user permissions are assigned according to this level to achieve a higher level of data security.**
3. **Integrity requirements are used to ensure that the value of the primary attribute of each processing object is unique and generally cannot be empty; the value of the reference attribute of each processing object must come from the referenced attribute. User-defined integrity (in accordance with actual requirements) can be used to ensure that the data meets the requirements of higher specifications. The detailed integrity requirements can be seen in the logical design stage of the system.**
4. **Database integrity (Database Integrity) indicates the logical concordance, rectitude, validity and compatibility of the contents of the DB. Database integrity is ensured by a variety of integrity restrictions. Database integrity constraints can be implemented by DBMS or applications, and DBMS-based integrity constraints are stored in the database as part of the model. The database integrity achieved by DBMS is designed in accordance with the database design steps, and the database integrity achieved by application software is incorporated into the application software design.**

**8. Relational Data Model**

**(1) E R diagram**

Diagram, schematic

Description automatically generated

**(2) Relationship Model Establishment**

**The relational model is transformed from the ER diagram. In fact, it is to clearly show the entity, the attribute of the entity, and the connection between the entity. This transformation generally follows the following rules. An entity type is transformed into a relational model. The attribute of the entity is the attribute of the relationship, and the code of the entity is the code of the relationship.**

**This database system includes multiple relational modes such as Invoice, payment, product, user, etc.**

**Customer (CustomerID，Name, Address, PanNo, EmailID, IncludeTaxAmount, PlaceOfSupply CustomerName, CompanyName, GetRegisterNo, MobilePhone, Opening Balance, CustomerForm)**

**PaymentTable (CustomerID, AmountReceived, PaymentDate, PaymentID, PaymentMode, TransactionID, FinancialYear)**

**Users (UserID, EmailID, Name, Password, PhoneNumber, Roles)**

**InvoiceTable (InvoiceID, InvoiceDate, Subtotal, TaxAmount, Address, CustomerID, PanNo, PlaceOfSupply, InvoiceNumber, InvoiceDate, RegisterNo, VehicleNo, CreatedTime, LastModifiedTime)**

**Invoiceitem (InvoiceID, ProductID, Quantity, TaxAmount, ItemTotal, ItemPrice, GstTaxRate, ItemOrder, LineitemID)**

**ProductTable (ProductID, Unit, Price, GstPercentage, ProductType, ItemDesc, StockKeepingUnit, Status)**

**The ‘InvoiceID’ is the foreign key for the Invoiceitem table and invoiceID will be used by the ‘invoice item’ table to connect or reference to the invoice table. Because Invoice ID is the primary key for the invoice table and the foreign key for the invoice item table.**

1. **The relationship between our invoice table and invoice item table is one to many, so the invoice table and invoice item table cannot be merged into one table.**
2. **The relationship between customer and payment is one to many, a customer can have more than one payment, the same person can enter the supermarket twice to buy goods and thus get two payment bills. We have only one user in our dataset, single user, so we don't need to connect it with other tables. customer cannot generate invoice . In the invoice, we have the cutomer ID, which is used to connect the customer.**

**3）Why is that there is an invoice table and there is still an invoiceItem table? For example, a customer in the customer table goes to the supermarket to buy something, he buys 5 things, these 5 things are put into the Product Table, so from the figure you can see that the relationship between the customer table and the product table is one to many relationship. But when the owner (user table) generate the invoice, he will give you a single invoice (there are 5 products on this bill), User gives you a single invoice, and there are 5 items in this single invoice, so the relationship between our invoice table and invoice item table is one to many, so invoice table and invoice item table cannot be merged into one table.**

**4）The relationship between customer and payment is one to many, a customer can have more than one payment, the same person can enter the supermarket twice or three times to buy goods and thus get 2 or 3 payment invoices(bills). The database has only one user in our dataset, single user, so don't need to connect it with other tables. customer cannot generate invoice . in the invoice, there is the cutomer ID, which is used to connect the customer.**

**9. Database Performance Test**

**Jmeter Interpretation of test results：**

**First, create 10 threads, which means 10 users will query the database at the same time. And then set 30 seconds as the ramp-up period, which means it will take 30 seconds for the database to finish the request. The JMeter will take 30 seconds to get all 10 threads up and running. Each thread will start 3(30/10) seconds after the previous thread was begun. Loop Count is 1 means: 1 iteration for each user in the group using Loop Count. In this picture, JMeter will take 30 seconds to get all the 10 threads up and running. Create a pool whose variable name is ‘ Test\_pool ‘. The max number of connections is 10. Set the Database URL as jdbc:mysql (format) localhost:3307, and find it on mysql workbench. Invoiceappdb is thedatabase name. Set the JDBC Driver class as com.mysql.jdbc.Driver, which is a unique driver for Mysql Username is root by default, I find it in mysql workbench (connect to database part). And then fill the Pool name, which I just seted into the blanks. And then select the query type as select statement. To write a query, SELECT \* from customertable ORDER BY customername ASC to search all the customer name in the customertable and order them by ascending order.**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**1. In this result, the thread 2 has loading time 11, which means it takes 11 milliseconds for thread 2 to run the query. Thread 2 will use 11 milliseconds to return the result.2. Connect time: In How long ( how much time) does the database connected to the JmeterLatency means the 2nd thread has to wait 4 seconds for the 1st thread to finish. 3. Response code(200) is a digital representation of response message (OK)**Graphical user interface, application

Description automatically generated

**In the summary report, we have 10 samples, Average execution time of the query of these 10 users is 19 milliseconds.Mininum execution time of the query of these 10 users takes 5 milliseconds.Maximum execution time of the query of these 10 users takes 100 milliseconds.Standard deviation of the execution time of the query is 27.33 mimlliseconds.Error is 0Throughput: number of requests / (total time) = 10 requests/ 27 seconds \*60 = 22.22 requests/ min**

**10, Database Performance Optimization.**

**Database optimization is the strategy of reducing database system response time**.

**Here are three comparison:**

1. **Comparison between key column and non-key column. (1) First, since product it is the primary key for product table, I want to select all from the producttable, sorted ascendingly by the productid, which is the key column. The duration of the query is 0.016 seconds. (2) I select all from the producttable, sorted descendingly by the unit and price, (it orders by unit, but if some rows have the same unit, it orders them by price), which are the non-key column. The duration of the query is 0.047. The second query is slower than the first query. Conclusion: Odering by the non-key column is slower than ordering by the key column. Because key column stored the data in the sorted form. However, non-key column stored the data randomly.**



1. **Comparison between contrain column and non-constrain column. (1) I select all from the product table where product id is greater than 5 (product id is a key column in this table), and then sorted ascendingly by its productid. This takes 0.016 seconds; (2) First, I created a ‘product’ table, which has no key column, and copy all the records from ‘producttable’ to ‘product’ table. And then I select all from the ‘product’ table, where product id is greater than 5 (product id is a non-key column), and then sorted descendingly by the unit and price. This takes 0.031 seconds. Conclusion: The contrain column is faster than the non-contrain column.**



1. **Comparision between nested query and inner join. (1) I use nested query, also called subquery. First I select all from producttype, which is a column name from producttable, then I calculated the sum of the prices of all products which has the same type, the column name is ‘totalprice’. Because I want to know how much I have sold for this product type. ‘Where’ means the new nested table ‘t’ will use the same product type as the producttable. And then I use group by to find all the products who are the same type. And then I order the producttype by ascending order. This query takes 0.046 seconds. (2) First, I select producttype and all data (from the sumed prices table) from producttable, and then I use inner join to select records that have matching values in both producttable and sumprice table, which represented as ‘t’. And then I do the same query with the first one since I want to compare this one and the first one. I select producttype and total price from the producttable, and then I group the products which has the same type together. ‘agg’ is the alias of the product table. And then I created index on the product table, and then sort the producttype by ascending order. This query takes 0.032 seconds. The second query is optimized because it takes less time than the first query. Inner join is faster than the nested query in this example.**



**11.**

**Advantages of functional testing.**

**1. It does not depend on the interface, if the service starts normally and the parameters are passed clearly, test cases can be added and tests can be executed. 2. Test scripts do not require programming, familiar with http requests and business processes, you can write test cases based on input objects in the page. 3. Test scripts are easy to maintain, you can copy the test scripts and save a part separately. 4. you can skip the page limit and add illegal data to the backend program, so you can test the robustness of the backend program. 5. using badboy to record test scripts, you can quickly form test scripts. 6. Jmeter assertions can verify whether there are values in the code that need to be obtained. 7. using parameterization and the function function provided by Jmeter, you can quickly complete the test data addition and modification, etc.**

**Disadvantages of functional testing.**

**1. the use of Jmeter can not verify the JS program, and can not verify the page, so you need to verify manually. 2. Jmeter's assertion function is not very powerful 3. even if the jmeter script is executed smoothly, it is still impossible to determine whether the program is executed correctly, and sometimes it is necessary to enter the program to check, or check the response data of Jmeter. 4.Jmeter scripts need to be saved as local files for maintenance, and each script file can only save one test case, which is not conducive to the maintenance of scripts.**

**11 Future Enhancements and Perspective**

**13. Reference**

1. T. J. Moore, "A test process optimization and cost modeling tool," Proceedings., International Test Conference, 1994, pp. 103-110, doi: 10.1109/TEST.1994.527941.

2. M. Liu and R. Shan, "Design and implementation of the Relationlog deductive database system," Proceedings Ninth International Workshop on Database and Expert Systems Applications (Cat. No.98EX130), 1998, pp. 856-863, doi: 10.1109/DEXA.1998.707505.

3. L. Wei and W. Zhang, "A Method for Rough Relational Database Transformed into Relational Database," 2009 IITA International Conference on Services Science, Management and Engineering, 2009, pp. 50-52, doi: 10.1109/SSME.2009.79.

4. W. Yin Mok, "A Feasible Schema Design Strategy for Amazon DynamoDB: A Nested Normal Form Approach," 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2020, pp. 903-907, doi: 10.1109/IEEM45057.2020.9309967.

5. Xiuling He, Yang Yang, Zengzhao Chen, Ying Yu and Cailin Dong, "Form analysis and understanding based on knowledge," 2008 7th World Congress on Intelligent Control and Automation, 2008, pp. 9286-9291, doi: 10.1109/WCICA.2008.4594401.

6. K. Norvag, "A performance evaluation of log-only temporal object database systems," Proceedings. 12th International Conference on Scientific and Statistica Database Management, 2000, pp. 256-258, doi: 10.1109/SSDM.2000.869795.

7. S. Y. W. Su, S. Ranka and Xiang He, "Performance analysis of parallel query processing algorithms for object-oriented databases," in IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 6, pp. 979-996, Nov.-Dec. 2000, doi: 10.1109/69.895805.

8. D. Taniar, "High Performance Database Processing," 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, 2012, pp. 5-6, doi: 10.1109/AINA.2012.140.