****

JSPM’S

**BHIVRABAI SAWANT INSTITUTE OF TECHNOLOGY& RESEARCH, WAGHOLI**

**DEPARTMENT OF COMPUTER ENGINEERING**

LAB MANUAL

Programming Lab-I

T.E (Comp) Ac. Yr 2014-15

**Prepared By,**

**Prof. Kumbhar V.S.**

**Prof.Mrs.Jadhav M.**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of experiments** | **Page No.** |
|  | **Assignments Group A (Mandatory)** |  |
| 1 | DBMS using connections(Client-Data server, two tier) Oracle/MySQL (ODBC/JDBC), SQL  prompt to create data base tables insert, update data values, delete table, use table, select  queries with/without where clause. |  |
| 2 | DBMS using connections(Client-application server-Data sever, three tier) Oracle/MySQL  (ODBC/JDBC), SQL Joints, prompt.(three tier) |  |
| 3 | Design and Develop SQL DDL statements which demonstrate the use of SQL objects  such as Table, View , Index using Client-Data sever(two tier) |  |
| 4 | Write a program in Python/C++ to read display the i-node information for a given text file,  image file. |  |
| 5 | Write an IPC program using pipe. Process A accepts a character string and Process B  inverses the string. Pipe is used to establish communication between A and B processes  using Python or C++. |  |
| 6 | Use Python for Socket Programming to connect two or more PCs to share a text file. |  |
|  | **Assignments Group B (Any Six Assignments, All assignments to be covered in a Batch)** |  |
| 1 | Design at least 10 SQL queries for suitable database application using SQL DML statements:  Insert, Select, Update, Delete Clauses using distinct, count, aggregation on Client-Data  sever(three tier) |  |
| 2 | Implement database with suitable example using MongoDB and implement all basic operations and administration commands using two tier architecture. |  |
| 3 | Use MongoDB to process semi structured and unstructured data collections such as Rfid,  Images, blogs use python/Java MongoDB interface. |  |
| 4 | Write an python/Java application using MongoDB to maintain the blog for composing the  blog consists of text columns, images and videos also calculate the hit or users visited by  Drawing 2D graphs. |  |
| **Sr. No.** | **Name of experiments**  **Assignments Group B (Continued)** | **Page No.** |
| 5 | Write a program in Python/C++ to test that computer is booted with Legacy Boot ROM-  BIOS or UEFI. |  |
| 6 | Write a program in C++ to create a RAMDRIVE and associate an acyclic directory structure  to it. Use this RAMDRIVE to store input, out files to run a calculator program. |  |
| 7 | Write a program in C++ to develop a tool using GRUB2 or GRUBx64.rfi select and display  a boot partition. (use appropriate overloading) |  |
| 8 | Write a Python/Java/C+ program to verify the operating system name and version of Mobile  Devices. |  |
| 9 | Write a program using MongoDB to compose a web news-letter consisting of videos,  images, text use python MongoDB interface. |  |
| 10 | Create a iso boot image using open source tools. |  |
| 11 | Write a python program for creating virtual file system on Linux environment. |  |
| 12 | Write a program in C++ to make USB Device Bootable by installing required system files |  |
| 13 | Write a program in python for USB Device File Management. Check usefulness of  command e2fsck for different file systems mounted on computer. |  |
| 14 | Aggregation and indexing with suitable example using Cassendra and RdfID based  employees attendance system. |  |
| 15 | Aggregation and indexing with suitable example using MongoDB. |  |
| 16 | Map reduce operation with suitable example using MongoDB. |  |
| 17 | Indexing and querying with MongoDB using suitable example. |  |
| 18 | Connectivity with MongoDB using any Java application. |  |
| 19 | Using MongoDB create a database of employee performance, employee attendance on the  workstation. Perform statistical analysis for the results of the products produced by  employees rated as passed ok, damaged products ( 5 samples per batch size 1000) and the  portion covered in the training and absentee of the employees during training. Use  programming language R. (or R-Python/R-Java) or equivalent assignment using R  Programming Language for BiGDATA computing. |  |
|  | **Assignment Group C: Advance Technology Assignments (Any One, all three to be**  **covered in a Batch)** |  |
| **Sr. No.** | **Name of experiments** | **Page No.** |
| 1 | BIG DATA applications using Hadoop |  |
| 2 | BIG DATA applications using Blogs |  |
| 3 | Big Data Predictive Machine Learning |  |
| 4 | Create and test functioning of Windows-8 ReFS (Resilient File System) |  |

**GROUP A**

Assignment No: - 1 (Group A)

**DML with JDBC connectivity**

**Title:-**DML queries with Client Server Architecture using java

**Problem definition:-**DBMS using connections (Client-Data sever, two tier) Oracle/MySQL (ODBC/JDBC), SQL prompt to create data base tables and perform following operations:-

1. Insert
2. update data values
3. delete table
4. Select queries with/without where clause.
5. demonstrate use of stored procedure / function (create procedure at the data side and make use of it on the client side)

**Learning Objectives:-**

1) Implementation of the problem statement using Object oriented programming and DBMS features.

**Learning Outcomes:-**

1) Use of appropriate connectivity tool for implementation.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System, Fedora19, 20

3. Eclipse, JDK 7.1

**Mathematical Model:-**

Let S be the solution perspective of the class DML queries such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state and e be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD-Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class DML queries:

s=initial state of the created schemas ()

s={init()}- sets the default values for all five variables to respective values given in assignment.

e= and e be the end state i.e. post DML operations on schemas

Input i=(I1,I2)

I1={DSN} where DSN(Data source name) is the connectivity created between Java and Mysql with help of JDBC driver.

I2={name of database, name of table , connection object, recordset object, command object},

name of database≠ NULL & database must exist in the Mysql server side

name of table ≠NULL & table must exist in the mentioned database on server side.

Connection object ≠ NULL

Recordsetobject≠NULL

Command object≠NULL

O={Data inserted,data deleted,data updated,data found}

f={Insert,Update,Delete,Select}

Insert={value of col 1,value of col 2…..value of col n} n≠Ø. n=no. of columns.

Update={value of col 1 , value of col 2 ….where value of col m=a} m≠Ø& a≠ Ø

Delete={col 1 , col 2…col n where value col m = a}.m≠Ø& a≠ Ø

Select={ col 1 , col 2…col n where value col m = a } m≠Ø& a≠ Ø

Success- desired output is generated in and shown in the java frame.

Success set contains following:-

|  |  |
| --- | --- |
| **Input** | **Output** |
| Data values for table | Data should be inserted in table |
| Searching a specific record which exists in the system | Record must be retrieved from table & displayed |
| Searching for a record which does not exist in system | “Record not found!!” , message must be prompted |
| Updating a record | Record must get updated with new values |
| Deletion of a record | Record must get deleted from the table |

Failure- desired output is not generated and error is shown in a message box. The failure cases are as follows:-

|  |  |
| --- | --- |
| **Input** | **Output** |
| Alphabetical data for numeric field | Error should be displayed |
| Repeating the primary key value to insert data | Error must be displayed |
| Entering a value for foreign key which is absent in parent table | Error must be displayed |
| Deletion of a record | Record does not get deleted from the table |

**State Transition Diagram** –

2

1

4

3

5

6

7

***States:-***

s – Initial state of schema. (Refer s of mathematical model)

e- end state i.e. after data is successfully inserted, updated or deleted in/from table.(Refer e of mathematical model)

s1- DSN successfully created

s2- connection established

s3- query executed

***Events:-***

1. DSN JDBC parameters provided (Refer set I1 of mathematical model)
2. DSN JDBC parameters not available
3. Valid Connection string provided ( Refer set I2 of mathematical model)
4. Invalid connection string provided
5. Valid query submitted (Refer set f of mathematical model)
6. Query executed successfully (Refer Success set of mathematical model)
7. Invalid query submitted (Refer Failure set of mathematical model)

**Test Cases:-** Refer Success and failure cases in mathematical model

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Data values for table | Data should be inserted in table |  |
| TC\_02 | Alphabetical data for numeric field | Error should be displayed |  |
| TC\_03 | Repeating the primary key value to insert data | Error must be displayed |  |
| TC\_04 | Entering a value for foreign key which is absent in parent table | Error must be displayed |  |
| TC\_05 | Searching a specific record which exists in the system | Record must be retrieved from table & displayed |  |
| TC\_06 | Searching for a record which does not exist in system | “Record not found!!” , message must be prompted |  |
| TC\_07 | Updating a record | Record must get updated with new values |  |
| TC\_08 | Deletion of a record | Record must get deleted from the table |  |

**Theory:-**

*JDBC*- It is a Java-based data access technology from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. JDBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.

If a database operation fails, JDBC raises a [SQLException](http://docs.oracle.com/javase/7/docs/api/java/sql/SQLException.html). There is typically very little one can do to recover from such an error, apart from logging it with as much detail as possible.

**Syntax of DML commands:-**

1.Insert command

INSERT [INTO] *tbl\_name* values(val1,val2,...)

2. Update Command

UPDATE *table\_references*

SET *col\_name1*={*expr1*|DEFAULT} [, *col\_name2*={*expr2*|DEFAULT}] ...

[WHERE *where\_condition*]

3. Delete Command

DELETE FROM *tbl\_name*

[WHERE *where\_condition*]

4. Select Command

Select [Column names] from table\_name [where] [conditions]

5. PL/SQL Stored Procedure and Function

**Procedure:** A procedure is a subprogram that performs a specific action or task. A procedure has two parts.

* The procedure specification: The procedure specification specifies the procedure name and the parameters it accepts. It is not necessary to create a procedure that accepts parameters.
* The procedure body: The procedure body contains the declarative section without DECLARE keyword, the executable section and an exception section.

1. **Creating a Procedure: -** To create a procedure from the database.

Syntax:-

Create [or replace] PROCEDURE procedure\_name

[(argument1 [IN / OUT / IN OUT] type),

(argument2 [IN / OUT / IN OUT] type),

….]

IS/AS

Procedure\_body

Where

Procedure\_name: – is the name of the procedure to be created

Argument:- is the name of the procedure parameter

Type:- Is the data type of the associated parameter

Procedure\_body:-Is a PL/SQL block that makes up the code of the procedure.

IN: **-** This is default mode. The value of the actual parameter is passed into the procedure. Inside the procedure the formal parameter is considered read only.

OUT: -Any value the actual parameter has when the procedure is called ignored. Inside the procedure, the formal parameters are considered as write only.

IN OUT: -This mode is combination of IN and OUT

1. **Deleting Procedure:-** To remove a procedure from the database.

Syntax:-

Drop procedure <procedure\_name>;

**Function:-**

A function is a subprogram, which is used to compute values. It is similar to a procedure; function also takes arguments and can be in different modes. Function also can be stored in the database. It is a PL/SQL block consisting of declarative, executable and exception section.

Difference between procedure and function is that the procedure call is a PL/SQL statement by itself, while a function call is called as a part of an expression. A function can return more than one value using OUT parameter. A function can be called using positional or named notation.

1. **Creating a Function: -** To create the subprogram from the database.

Syntax:-

Create [or replace] FUNCTION function\_name

[(argument1 [IN / OUT / IN OUT] type),

(argument2 [IN / OUT / IN OUT] type), ….]

Return return\_type IS / AS

Function\_body

Where

Function\_name: – is the name of the function to be created

Argument: - is the name of the function parameter

Type: - Is the data type of the associated parameter

Function\_body:-Is a PL/SQL block containing code for the function.

IN:-This is default mode. The value of the actual parameter is passed into the procedure. Inside the procedure the formal parameter is considered read only.

OUT:-Any value the actual parameter has when the procedure is called ignored. Inside the procedure, the formal parameters are considered as write only.

INOUT:-this mode is combination of IN and OUT

1. **Deleting a Function: -** To remove the subprogram from the database.

Syntax:-

Drop function<function\_name>;

**Conclusion: -** The Client Server architecture in two-tier is used to implement the DML commands using Java as the Client end and mysql as the back (server) end. The GUI provided in java, makes it easier for the user to perform the insert, update and delete operations. If the schema changes in the server side the user just has to modify the query string in the java code, the connectivity need not be changed. Stored procedure and functions are written for programming using the data retrieved from the tables. Stored procedure can return multiple values , whereas a function can return only one value.

**References:-**

‘SQL, PL/SQL the programming language of oracle’, by Ivan Bayross (third edition).

Assignment No: - 2 (Group A)

**DML with JDBC connectivity (Three Tier)**

**Title:-**DML queries with Client Server Architecture using java

**Problem definition:-**DBMS using connections(Client-Data sever, two tier) Oracle/MySQL (ODBC/JDBC), SQL prompt to create data base tables insert, update data values, delete table, use table, select queries with/without where clause.

**Learning Objectives:-**

1) Implementation of the problem statement using Object oriented programming and DBMS features.

**Learning Outcomes:-**

1) Use of appropriate connectivity tool for implementation.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System, Fedora19, 20

3. Eclipse for JEE, JDK 7.1

4.Apache Tomcat.

**Mathematical Model:-**

Let S be the solution perspective of the class DML queries such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state and e be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD-Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class DML queries:

s=initial state of the created schemas ()

s={init()}- sets the default values for all five variables to respective values given in assignment.

e= and e be the end state i.e. post DML operations on schemas

Input i=(I1,I2)

I1={DSN} where DSN(Data source name) is the connectivity created between Java and Mysql with help of JDBC driver.

I2={name of database, name of table , connection object, recordset object, command object},

name of database≠ NULL & database must exist in the Mysql server side

name of table ≠NULL & table must exist in the mentioned database on server side.

Connection object ≠ NULL

Recordsetobject≠NULL

Command object≠NULL

O={Data inserted,data deleted,data updated,data found}

f={Insert,Update,Delete,Select}

Insert={value of col 1,value of col 2…..value of col n} n≠Ø. n=no. of columns.

Update={value of col 1 , value of col 2 ….where value of col m=a} m≠Ø& a≠ Ø

Delete={col 1 , col 2…col n where value col m = a}.m≠Ø& a≠ Ø

Select={ col 1 , col 2…col n where value col m = a } m≠Ø& a≠ Ø

Success- desired output is generated in and shown in the java frame.

Failure- desired output is not generated and error is shown in a message box.

**State Transition Diagram** -

2

1

4

3

5

6

7

***States:-***

s – Initial state of schema.

e- end state i.e. after data is successfully inserted, updated or deleted in/from table.

s1- DSN successfully created

s2- connection established

s3- query executed

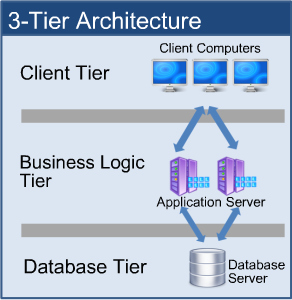
***Events:-***

1. DSN JDBC parameters provided
2. DSN JDBC parameters not available
3. Valid Connection string provided
4. Invalid connection string provided
5. Valid query submitted
6. Query executed successfully
7. Invalid query submitted

**Test Cases:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Data values for table | Data should be inserted in table |  |
| TC\_02 | Alphabetical data for numeric field | Error should be displayed |  |
| TC\_03 | Repeating the primary key value to insert data | Error must be displayed |  |
| TC\_04 | Entering a value for foreign key which is absent in parent table | Error must be displayed |  |
| TC\_05 | Searching a specific record which exists in the system | Record must be retrieved from table & displayed |  |
| TC\_06 | Searching for a record which does not exist in system | “Record not found!!” , message must be prompted |  |
| TC\_07 | Updating a record | Record must get updated with new values |  |
| TC\_08 | Deletion of a record | Record must get deleted from the table |  |

**Theory:-**



**Figure: Three Tier Architecture [1]**

In this architecture all three tiers are separated onto different computers. The UI runs on the client (what the user is working with). The BL is running on a separate server, called the business logic tier, middle tier, or service tier. Finally the DB is running on its own database server. By introducing the middle tier, the client is only handling presentation logic (UI). This means that only little communication is needed between the client and the middle tier (BL) making the client “thin” or “thinner”. An example of a thin client is an Internet browser that allows you to see and provide information fast and almost with no delay. Three tier is more scalable and improves performance. Security is also the best in the three-tier architecture because the middle tier protects the database tier. There is one major drawback to the N-tier architecture and that is that the additional tiers increase the complexity and cost of the installation.

*JDBC*- It is a Java-based data access technology from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. JDBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.

If a database operation fails, JDBC raises a [SQL Exception](http://docs.oracle.com/javase/7/docs/api/java/sql/SQLException.html). There is typically very little one can do to recover from such an error, apart from logging it with as much detail as possible.

**Joins:** The ability of relational ‘join’ operator is an important feature of relational systems. A join makes it possible to select data from more than table by means of a single statement. This joining of tables may be done in a many ways.

**Types of JOIN:**

* **Inner Join:** Also known as equi join. Used for comparing two columns with the operator =. Join is used for retrieving data from more than one table. Tables have relationship between them and pertaining information regarding to same database but with various attributes.

**Syntax:**

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>, <tablename2> WHERE <tablename1> **.** <columnname> = <tablename2> **.** <columnname> AND <condition> ORDER BY <columnname1>**;**

* **Outer Join**

Outer joins are similar to inner joins, but give a little bit more flexibility when selecting data from related tables. This type of joins can be used in situations where it is desired, to select all rows from the table on left (or right, or both) regardless of whether the other table has values in common & (usually) enter NULL where data is missing.

Syntax:

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>**,** <tablename2> WHERE <tablename1>.<columnname> = <tablename2>.<columnname> AND <condition> ORDER BY <columnname1>;

* **Right outer Join**

List the employee details with contact details (if any using right outer join. Since the RIGHT JOIN returns all the rows from the second table even if there are no matches in the first table.

Syntax:

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>, <tablename2> WHERE <tablename1>.<columnname> = <tablename2>.<columnname> AND <condition> ORDER BY <columnname1>;

* **Cross Join**

A cross join returns what known as a Cartesian product. This means that the join combines every row from the left table with every row in the right table. As can be imagined, sometimes this join produces a mess, but under the right circumstances, it can be very useful. This type of join can be used in situation where it is desired, to select all possible combinations of rows & columns from both tables. The kind of join is usually not preferred as it may run for a very long time & produce a huge result set that may not be useful.

Syntax:

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>, <tablename2>;

* **Self Join**

In some situation, it is necessary to join to itself, as though joining 2 separate tables.

This is referred to as self join

Syntax:

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>, <tablename2> WHERE <tablename1>.<columnname> = <tablename1>.<columnname> ;

**Set Operators**

Sometimes it is useful to combine query results from two or more queries into a single result.SQL supports three set operators which have the pattern:

Syntax:

<query 1> <set operator> <query 2>

The set operators are:

* Union [all] returns a table consisting of all rows either appearing in the result of <query1> or in the result of <query 2>. Duplicates are automatically eliminated unless the clause all is used.
* Intersect returns all rows that appear in both results <query 1> and <query 2>.
* Minus returns those rows that appear in the result of <query 1> but not in the result of <query 2>.
* All employee numbers and names from both tables:
* Employees who are listed in both EMP and DEPT table:
* Employees who are only listed in EMP:

**Conclusion: -** The Client Server architecture in three-tier is used to implement the DML commands using Java as the Client end and MYSQL as the back (server) end. The GUI provided in java, makes it easier for the user to perform the insert, update and delete operations. If the schema changes in the server side the user just has to modify the query string in the java code, the connectivity need not be changed. DML query output we have sent on web page by using web server TOMCAT/APACHE.

**References:-**

‘SQL, PL/SQL the Programming Language of Oracle’, by Ivan Bayross (third edition).

Figure [1]:-Link Referred: - http://blog.simcrest.com/what-is-3-tier-architecture-and-why-do-you-need-it/

JDBC Connectivity Guidelines:-The Complete Reference JAVA 7th Edition by Herbert Schildt

Assignment No: - 3 (Group A)

**DDL Commands using MYSQL**

**Title:-**DDL queries with Client Server Architecture using MYSQL.

**Problem definition: -** Design and Develop SQL DDL statements which demonstrate the use of SQL objects Such as Table, View, Index using Client-Data server (two tier).

**Learning Objectives:-**

Implementation of the problem statement using DBMS features.

**Learning Outcomes:-**

Use of appropriate Data Definition Language Commands to work with SQL objects.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 / Open Source Operating System.

3. MySQL Server and Client

**Mathematical Model:-**

Let S be the solution perspective of the class DML queries such that

**S={s, e, i, o, f, DD, NDD, success, failure}**

**s**=initial state and **e** be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

DD here is created table with inserted values.

NDD- Non deterministic data of the system S to be solved.

NDD is schema or structure of SQL objects defined by user with the help of DDL commands.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class DDL queries:

s=initial state of the created schemas ()

s= {init ()} - sets the default values for all five variables to respective values given in assignment.

e= and e be the end state i.e. post DML operations on schemas

Input i= (I1)

I1= {name of database, name of table, name of view, name of index},

I2={name of column, name of data type}- in Alter table command

name of database≠ NULL & database must exist in the MySQL server side

name of table ≠NULL & table must exist in the mentioned database on server side.

O= {Table Created, Table Renamed, Table Dropped, Table Altered, Table Truncated, View created, Index Created }

f= {Create, Rename, Alter, Drop, Truncate, Describe}

Create= {Table name(att.1 datatype(size),att.2 datatype(size),…..att.**n** datatype(size) } n≠Ø. n=no. of columns.

Rename= {Table name a} a≠ Ø

Drop= {Table name a} a≠ Ø

Describe= {Table name a} a≠ Ø

Truncate= {Table name a} a≠ Ø

Alter= {Table name **a** ADD Newcolumnname datatype(size)} a≠ Ø

Alter= {Table name **a** Modify already created columnname Newdatatype(Newsize)} a≠ Ø

Success- desired output is generated in and shown in the java frame.

Failure- desired output is not generated and error is shown in a message box.

**State Transition Diagram** -

**e**

1

3

2

4

5

7

8

10

9

6

11

***States:-***

s – Initial state of schema.

e- end state i.e. after data is successfully created table, renamed table, truncated table, dropped table,

described table and altered table with ADD and Modify constraint, view and index successfully created.

s1- successfully created table

s2- successfully described table

s3- successfully renamed table

s4- successfully altered table

s5- successfully truncated table

s6- successfully dropped table

***Events:-***

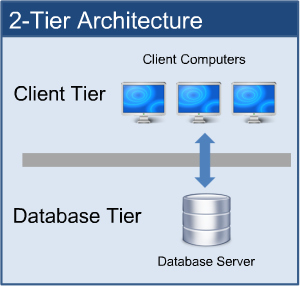
1. Table name provided for creating with attribute name, data type and size.
2. Table name more than 30 characters provided.
3. Valid describe command provided.
4. Invalid name of table provided
5. Valid query to alter table provided.
6. Table already have data with another data type then modification of data type is not possible..
7. Valid query to truncate table provided.
8. Valid query to drop table provided.
9. Query executed successfully
10. Invalid query submitted

**Test cases:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Table creation | Table should be created with attributes |  |
| TC\_02 | Alphabetical data for numeric field | Error should be displayed |  |
| TC\_03 | Repeating the primary key value to insert data | Error must be displayed |  |
| TC\_04 | Entering a value for foreign key which is absent in parent table | Error must be displayed |  |
| TC\_05 | Table name given with space or exisiting name | Adequate Error must be displayed |  |
| TC\_06 | While altering if data with another datatype already present in table. | Error must be displayed regarding datatype. |  |

**Theory:-**

**Two Tier Architecture:** This architecture is also called Client-Server architecture because of the two components: The client that runs the application and the server that handles the database back-end. The client handles the UI and the BL and the server handles the DB. When the client starts, it establishes a connection to the server and communicates as needed with the server while running the client. The client computer usually can’t see the database directly and can only access the data by starting the client. This means that the data on the server is much more secure. Now users are unable to change or delete data unless they have specific user rights to do so.



**Figure:-Two Tier Architecture [1]**

**Data Definition Language**

DDL (Data Definition Language) is one of the MySQL Language which deals with queries related to definition of data. Commands are used to create, delete, or change the objects of a database.

DDL deals with structure or schema of a table. Database administrator generally work with DDL commands.

DDL Commands: - For Database, Table, View, Index, Procedure and Functions (MySQL Objects)

* **Create** - Used to create a table.
* **Describe** - Used to view the structure of the table.
* **Alter** -Used to add a new column, modify the existing column definition and to include or drop integrity constraint.
* **Rename-** Used to Rename Database/Table.
* **Truncate** - If there is no further use of records stored in a table and the structure has to be retained, then the records alone can be deleted. Table structure remains as it is after truncate command.
* **Drop** – used to delete the table structure.

**Table Creation**

**Rules for creating table:-**

● Reserved words cannot be used.

● Underscore, numerals, letters are allowed but not blank space in table name.

● Maximum length for the table name is 30 characters.

● 2 different tables should not have same name.

● Specify a unique column name (without space in name).

● Specify proper data type along with width.

● We can include “not null” condition when needed. By default it is ‘null’. We have to give all constraints of keys while creation of table. Best option is to provide while table creation.

**Syntax**

**Create database databasename;**

**create table <table name>{attributename-1 datatype constraints if any,attributename-2 datatype constraints if any,…….attributename-n datatype constraints if any,};**

If we have to create a table based on attributes of already created table then use:-

**create table <table name> as (select(attribute-list) from <existing table name>);**

**ALTER TABLE – syntax**

**alter table <table name> ADD(attributename datatype (size))**

**alter table <table name> MODIFY(attributename newdatatype (newsize))**

To drop column from table using alter:-

*alter table drop column column name*;

**RENAME table**

To change the name of a table, view, sequence, or synonym, execute the rename statement.

**Syntax:**

*rename old name table to new table name;*

**DROP TABLE**

1. All data and structure along with indexes on the table get deleted.

2. Drop table commands effect cannot br rollbacked.

**Syntax:**

*drop table <table name>;*

**TRUNCATE: -** The truncate table statement

● Removes all rows from a table

● Release the storage space used by that table

● We cannot rollback row removal when using truncate.

**Syntax:**

**truncate table <table name>;**

**DESCRIBING TABLE**

*Describe <tablename>;*

**VIEW:**

A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data in the tables. A view is a “virtual table” or a “stored query” which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table.

Views gives Additional level of table security, Hides data complexity, Simplifies the usage by combining multiple tables into a single table.

**Creating and dropping view:**

**Syntax:**

*Create view <view name> [column alias names] as <query> [with <options> conditions]; Drop view <view name>;*

**Example**: Create view compstaff as select \* from collegeinfo where staffdept =’comp’;

Drop view compstaff;

**INDEX:**

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table, the more this costs. If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. If a table has 1,000 rows, this is at least 100 times faster than reading sequentially. If you need to access most of the rows, it is faster to read sequentially, because this minimizes disk seeks.

**MySQL uses indexes for these operations:**

* To find the rows matching a WHERE clause quickly.
* To eliminate rows from consideration. If there is a choice between multiple indexes, MySQL normally uses the index that finds the smallest number of rows.
* To retrieve rows from other tables when performing joins. MySQL can use indexes on columns more efficiently if they are declared as the same type and size.

CREATE [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX *index\_name*

[*index\_type*]

ON *tbl\_name* (*index\_col\_name*,...)

[*index\_option*] ...

*index\_col\_name*:

*col\_name* [(*length*)] [ASC | DESC]

*index\_type*:

USING {BTREE | HASH | RTREE}

*index\_option*:

KEY\_BLOCK\_SIZE [=] *value*

| *index\_type*

| WITH PARSER *parser\_name*

**Example of Index:**

The statement shown here creates an index using the first 10 characters of the name column:

CREATE INDEX p1 ON parts (partid);

**Conclusion:** The Client Server architecture in two-tier is used to implement the DDL commands using MYSQL Server and Client. We can successfully create SQL objects through these commands which can be very basic concept in Creation of any database needed for project. To speed up execution we successfully created indexes on table and also created views for storing results of some table.

**References:-**

Figure [1]:- Referred from : - http://blog.simcrest.com/what-is-2-tier-architecture-and-why-do-you-need-it/

**GROUP B**

Assignment No: - 1 (Group B)

**DML queries with aggregation**

**Title:-**DML queries with Client Server Architecture using java

**Problem definition:-** Design at least 10 SQL queries for suitable database application using following SQL DML statements:-

* + - 1. Insert
      2. Select
      3. Update
      4. Delete
      5. Clauses using distinct, count, aggregation

on Client-Data server(three tier).

**Learning Objectives:-**

1) Implementation of the problem statement using Mysql.

**Learning Outcomes:-**

1) Use of aggregation functions and clauses.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System/Fedora 19, 20

3. Mysql for 64-bit machines

4. JSP using APACHE Tomcat

**Mathematical Model:-**

Let S be the solution perspective of the class DML queries such that

S={s, e, i, o,f, DD, NDD, success, failure}

s=initial state and e be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD-Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class DML queries:

s=initial state of the created schemas ()

s= {init ()} - sets the default values for all five variables to respective values given in assignment.

e= and e be the end state i.e. post DML operations on schemas

Input i= (I1, I2)

I1={DSN} where DSN(Data source name) is the connectivity created between Java and Mysql with help of JDBC driver.

I2={name of database, name of table , connection object, recordset object, command object}, name of database≠ NULL & database must exist in the Mysql server side

name of table ≠NULL & table must exist in the mentioned database on server side.

Connection object ≠ NULL

Recordsetobject≠NULL

Command object≠NULL

O={Data inserted,data deleted,data updated,data found}

f={Insert,Update,Delete,Select}

Insert={value of col 1,value of col 2…..value of col n} n≠Ø. n=no. of columns.

Update={value of col 1 , value of col 2 ….where value of col m=a} m≠Ø& a≠ Ø

Delete={col 1 , col 2…col n where value col m = a}.m≠Ø& a≠ Ø

Select={ col 1 , col 2…col n where value col m = a } m≠Ø& a≠ Ø

Figure shows mapping of Input to output

Record Inserted

Table

Connection

Command

Recordset

Record Updated

Record Deleted

Record Found

Success- desired output is generated in and shown in the java frame.

Failure- desired output is not generated and error is shown in a message box.

**Test Cases:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Data values for table (Success) | Data should be inserted in table |  |
| TC\_02 | Alphabetical data for numeric field (Failure) | Error should be displayed |  |
| TC\_03 | Repeating the primary key value to insert data (Failure) | Error must be displayed |  |
| TC\_04 | Entering a value for foreign key which is absent in parent table (Failure) | Error must be displayed |  |
| TC\_05 | Searching a specific record which exists in the system(Success) | Record must be retrieved from table & displayed |  |
| TC\_06 | Searching for a record which does not exist in system(Failure) | “Record not found!!” , message must be prompted |  |
| TC\_07 | Updating a record (Success) | Record must get updated with new values |  |
| TC\_08 | Deletion of a record (Failure) | Record must get deleted from the table |  |
| TC\_09 | Search for a data with distinct clause and group by(Success) | Correct distinct values must be displayed |  |
| TC\_10 | Search for an aggregated value of a specific column in the tables  (Success) | Required value must be displayed. |  |

**Theory: - Aggregate Functions:**

Aggregate functions are statistical functions such as count, min, max etc. They are used to compute a single value from a set of attribute values of a column: The SUM and AVG must be a collection of numbers, but the other operators can operate on collections of nonnumeric data types such as string.

* AVG:-It is used to find out the average value.

Syntax: - avg (column\_name)

* SUM:-It is used to find out the total or sum.

Syntax: - sum (column\_name)

* MIN:-It is used to find out the minimum value.

Syntax: - min (column\_name)

* MAX:-It is used to find out the average value. COUNT:-It is used to find out the total number of record or rows present in the relation or table.

Syntax: - max (column\_name)

* Count: It is used to eliminate the duplicate and null values in the specified column.

Syntax: - count (Distinct column\_name)

**Distinct:-** In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values. The DISTINCT keyword can be used to return only distinct (different) values.

SELECT DISTINCT column\_name,column\_name  
FROM table\_name;

**Ordering operation:** using ORDER BY: The order by cause is used to arrange the records in ascending or descending order

Syntax:-

Select<expr> from <table\_name>

[where condition(s)]

[order by{column,expr}[asc|desc]];

* Asc:-orders the rows in ascending order.by default order are asc.
* Desc:-orders the rows in descending order.

**Conclusion: -** Aggregation pipelining is a feature which allows user to apply various filters on a set of documents to obtain the result of operations on group of data. Aggregation helps in grouping the relevant data to obtain statistical information like min, max, count etc.

**References:-**

‘SQL, PL/SQL the Programming Language of Oracle’, by Ivan Bayross (third edition).

**ASSIGNMENT NO.2 (Group B)**

**MongoDB Basic and Administration Commands**

**Title: -** Implement database with suitable example using Mongo DB and implement all basic operations and administration commands using two tier architecture.

**Problem definition: -** Design and Develop Mongo DB statements which demonstrate the use of Mongo DB Basic operations and administration commands using Client-Data server (two tier).

**Learning Objectives:-**

1) Implementation of the problem statement using DBMS features for unstructured data.

**Learning Outcomes:-**

1. Use of appropriate Mongo DB commands basic as well as administration by creating database in Mongo DB.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 / Open Source Operating System.

3. Mongo DB Server and Client.

**Mathematical Model:-**

Let S be the solution perspective of the class DML queries such that

**S={s, e, i, o, f, DD, NDD, success, failure}**

**s**=initial state and **e** be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Mongo DB basic queries:

s=initial state of the created collection ()

s= {init ()} - sets the default values for all variables to respective values given in assignment.

e= and e be the end state

Input i= (I1, I2)

I1= {name of collection, name of database using},

I2= {name of fields in document}-

name of collection≠ NULL & collection must exist in the Mysql server side

name of field ≠NULL & field must exist in the mentioned database on server side.

O= {Collection Created, Collection inserted, collection read, Collection updated, Collection deleted}

f= {db.Createcollection, db.find, db.insert ,db.update,db.remove}

Create= {Collecton name(key1:value1,key2:value2,….key n:value n) } n≠Ø. n=no. of fields.

insert= {fields a} a≠ Ø

Update values= {value name a} a≠ Ø

(Remove)Delete= {fieldname a} a≠ Ø

Success- desired output is generated in and shown in the java frame.

Failure- desired output is not generated and error is shown in a message box.

**State diagram:-**

**e**

3

2

4

5

7

1

6

9

10

8

***States:-***

s – Initial state of unstructured DB.

e- end state i.e. after data is successfully created

s1- successfully created collection

s2- successfully inserted documents in collection with key and value.

s3- successfully read data from collection

s4- successfully update collection.

s5- successfully remove fields from documents of collection.

***Events:-***

1. Collection name provided for creating with key as attribute name, and value.
2. Collection name with space provided
3. Insert document command provided.
4. Invalid name of collection provided then consider another collection.
5. Valid command to read documents provided.
6. Collection name not valid then not possible to display contents.
7. Valid command to update collection provided.
8. Valid command to remove collection provided.
9. Query executed successfully
10. Invalid query submitted

**Theory:**

Mongo DB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

**Database**

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**Collection**

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields.

Typically, all documents in a collection are of similar or related purpose.

**Document**

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Below given table shows the relationship of RDBMS terminology with MongoDB

**RDBMS MongoDB**

|  |  |
| --- | --- |
| Database | Database |
| Table | Collection |
| Tuple | Row Document |
| Column | Field |
| Table Join | Embedded Document |
| Primary Key | Primary Key (Default key \_id provided by mongodb itself) |
| Database Server and Client |  |
| Mysqld/Oracle | Mongod |
| Mysql/sqlplus | mongo |

To start mongod process (server side) through bin directory of mongo installed folder of mongo set up.

bin> mongod --dbpath path

To start mongod process (client side)

Now we are in mongo shell and Commands to work on mongo shell:-

To know current database. >db

To work in any database >use employee

To get help parameters related with database. >db.help

To get stastical information related with database. >db.stats()

Sample document in Mongo DB

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2011,1,20,2,15),

like: 0

},

{

user:'user2',

message: 'My second comments',

dateCreated: new Date(2011,1,25,7,45),

like: 5

}

]

}

**Advantages of MongoDB over RDBMS**

**1.** Schema less: MongoDB is document database in which one collection holds different different documents. Number of fields, content and size of the document can be differ from one document to another.

2. Structure of a single object is clear

3. No complex joins

4. Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language. that's nearly as powerful as SQL

5. Tuning

6. Ease of scale-out: MongoDB is easy to scale

7. Conversion / mapping of application objects to database objects not needed

8. Uses internal memory for storing the (windowed) working set, enabling faster access of data

We should use MongoDB

* Document Oriented Storage : Data is stored in the form of JSON style documents
* Index on any attribute
* Replication & High Availability
* Auto-Sharding
* Rich Queries
* Fast In-Place Updates
* Professional Support By MongoDB

**Administration Operations in MongoDB**

The most basic tool for getting statistics about a running MongoDB server is the

serverStatus command

1. db.runCommand({"serverStatus" : 1})

serverStatus provides a detailed look at what is going on inside a MongoDB server.

Information such as the current server version, uptime (in seconds), and current number

of connections is easily available.

2. starting from a shell connected to a server without security turned on:

> use admin

switched to db admin

> db.addUser("root", "abcd");

{

"user" : "root",

3. to know current user information:-

>show user;

4. to remove users along with all its collections you have to go through admin:

>use admin

switched to db admin

>db.system.users.remove({"user" : "test\_user"});

5.fsync and Lock

MongoDBâ€™s fsync command allows us to copy the data directory of a running MongoDB server without risking any corruption.

The fsync command will force the MongoDB server to flush all pending writes to disk.

It will also, optionally, hold a lock preventing any further writes to the database until the server is unlocked. This write lock is what allows the fsync command to be useful

for backups.

> use admin

switched to db admin

> db.runCommand({"fsync" : 1, "lock" : 1});

{

"info" : "now locked against writes, use db.$cmd.sys.unlock.findOne() to unlock",

"ok" : 1

}

6. UNLOCK:- After performing the backup, we need to unlock the database again:

Simultaneously if connection terminal is open you can see changes over their while ruuning this command.

> db.$cmd.sys.unlock.findOne();

{ "ok" : 1, "info" : "unlock requested" }

> db.currentOp();

{ "inprog" : [ ] }

7.For seeing information about a whole collection, there is a stats function:

> db.boards.stats() // db.collection\_name.stats()

{

"ns" : "brains.boards",

"count" : 12,

"size" : 32292,

"avgObjSize" : 2691,

"storageSize" : 270336,

"numExtents" : 3,

"nindexes" : 2,

"lastExtentSize" : 212992,

"paddingFactor" : 1.0099999999999825,

"flags" : 1,

"totalIndexSize" : 16352,

"indexSizes" : {

"\_id\_" : 8176,

"username\_1\_slug\_1" : 8176

},

"ok" : 1

}

8.> db.stats(); //check databse status

{

"db" : "testdata",

"collections" : 3,

"objects" : 5,

"avgObjSize" : 60.8,

"dataSize" : 304,

"storageSize" : 24576,

"numExtents" : 3,

"indexes" : 1,

"indexSize" : 8176,

"fileSize" : 67108864,

"nsSizeMB" : 16,

"dataFileVersion" : {

"major" : 4,

"minor" : 5

},

"extentFreeList" : {

"num" : 0,

"totalSize" : 0

},

"ok" : 1

}

9. Rename collection permanentaly

db.sourceColl.renameCollection("newName")

>> db.bank.renameCollection("bankdata");

{ "ok" : 1 }

**Conclusion:** The Client Server architecture in two-tier is used to implement the MongoDB commands using MongoDB Server and Client. We can successfully create collection and performed all basic commands called **CRUD** operations in mongo. We can perform administration commands using admin database in mongo shell

**References:-**

MongoDB Commands:- http://docs.mongodb.org/manual/tutorial/

**ASSIGNMENT NO.3 (Group B)**

MongoDB Python interface

**Title: -** Prosessing MongoDB data collections as Rfd,images ,blogs with python interface .

**Problem definition: -** Use MongoDB to process semi structured and unstructured data collections such as Rfid, images, blogs use python/Java MongoDB interface.

**Learning Objectives:-**

1) Implementation of the problem statement using DBMS features for unstructured data.(MongoDB) and establish interface with python.

**Learning Outcomes:-**

1. Use of appropriate Mongo DB interface with python and processing collection with rfid, images and blogs in MongoDB.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 / Open Source Operating System.

3. Mongo DB 2.2.3

4. Python driver

5.JDK 1.7

6.MongoDB-Java-Driver 2.10.1

7.Eclipse

**Mathematical Model:-**

Let S be the solution perspective of the class processing MongoDB queries such that

**S={s, e, i, o, f, DD, NDD, success, failure}**

**s**=initial state and **e** be the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Mongo DB basic queries:

s=initial state of the created collection ()

s= {init ()} - sets the default values for all variables to respective values given in assignment.

e= and e be the end state

Input i= (I1, I2,I3)

I1= {name of collections with Rfid},

I2={name of collections with images}

I2={name of collections with blogs}

name of collection≠ NULL & collection must exist in the MongoDB server side

name of field ≠NULL & field must exist in the mentioned database.

O= {Collection Created, interface created, Collection processed, collection reads}

f= {db.Createcollection, db.find, db.insert ,functions for interface}

Create= {Collecton name(key1:value1,key2:value2,….key n:value n) } n≠Ø. n=no. of fields.

insert= {fields a} a≠ Ø

Success- desired output is generated in and shown in the java frame.

Failure- desired output is not generated and error is shown in a message box.

**Theory:-**

**Mongodb Server**

exam@ccompl0768:~/mongodb-linux/bin$ ./mongod --dbpath /home/exam/mongodb-linux/data/db

**Mongodb client**

exam@ccompl0768:~/mongodb-linux/bin$ ./mongo

MongoDB shell version: 2.6.1

connecting to: test

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

<http://docs.mongodb.org/>

Questions? Try the support group

<http://groups.google.com/group/mongodb-user>

> db

test

> show dbs

admin (empty)

local 0.078GB

> use pnr

switched to db pnr

Python Driver

PyMongo is the recommended way to work with MongoDB from Python.

**Installing with easy\_install**

To use easy\_install from [setuptools](http://pypi.python.org/pypi/setuptools) do:

$ easy\_install pymongo

To upgrade do:

$ easy\_install -U pymongo

Alternative for ubauntu:

sudo apt-get install python-setuptools

**Making a Connection with MongoClient**

The first step when working with PyMongo is to create a [MongoClient](http://api.mongodb.org/python/current/api/pymongo/mongo_client.html#pymongo.mongo_client.MongoClient) to the running mongod instance. Doing so is easy:

>>> from pymongo import MongoClient

>>> conn=MongoClient()

The above code will connect on the default host and port. We can also specify the host and port explicitly, as follows:

>>> conn = MongoClient('localhost', 27017)

>>> conn

MongoClient('localhost', 27017)

>>> MongoClient('localhost', 27017)

MongoClient('localhost', 27017)

>>> db=conn.pnr

>>> db

Database(MongoClient('localhost', 27017), u'pnr')

To read data

>>> cur=db.blog.find()

>>> cur.next()

{u'Message': u'Hello', u'\_id': ObjectId('53a02df25ce53643797b0393'), u'Name': u'Poonam'}

Another example:

>>> from pymongo import MongoClient

>>> conn=MongoClient()

>>> conn

MongoClient('localhost', 27017)

>>> db=conn.pnr

>>> db

Database(MongoClient('localhost', 27017), u'pnr')

>>> cur=db.teachers.find()

>>> cur.next()

{u'dept': u'E and TC', u'\_id': ObjectId('53915e971b21a2e6b204b448'), u'name': u' Rajesh '}

>>> cur.next()

{u'\_id': ObjectId('53915ec81b21a2e6b204b449'), u'name': u'Rama'}

>>> cur.next()

{u'\_id': ObjectId('53915f821b21a2e6b204b44a'), u'college': u'SKN', u'name': u'Advait'}

>>> for d in cur: print d

...

{u'Phone': 50.0, u'dept': u'IT', u'\_id': ObjectId('539161e41b21a2e6b204b44b'), u'name': u'Manasi', u'address': {u'city': u'Pune'}}

{u'dept': u'IT', u'\_id': ObjectId('539162681b21a2e6b204b44c'), u'name': u'Akshara', u'address': {u'city': u'Pune', u'state': u'Maharashtra'}}

{u'dept': u'IT', u'\_id': ObjectId('539163351b21a2e6b204b44d'), u'name': u'Raghav', u'contacts': [111111.0, 393939.0]}

{u'Phone': u'10', u'dept': u'IT', u'\_id': ObjectId('53917d2367cb2952b36b2f0a'), u'name': u'Manasi', u'address': {u'city': u'Pune'}}

{u'Phone': 11.0, u'dept': u'IT', u'\_id': ObjectId('53917d5c67cb2952b36b2f0b'), u'name': u'Manasi', u'address': {u'city': u'Pune'}}

{u'dept': u'IT', u'\_id': ObjectId('5391806f99843eb6f32d24f6'), u'name': u'Deepa', u'contacts': [66.0, 99.0]}

>>>

**Data from other server**

>>> conn=MongoClient()

>>> conn=MongoClient('172.20.55.67')

>>> db = conn.pnr

>>> db.articles.find().next()

{u'dept': u'comp', u'\_id': ObjectId('53929fcc3bd3086d7ce30b44'), u'name': u'aparna'}

**Inserting Values through python:**

content on Mongodb before inserting:

> db.blog.find()

{ "\_id" : ObjectId("53a1405425bd680ef3ed971b"), "Name" : "Poonam", "Message" : "Hello" }

{ "\_id" : ObjectId("53a1407125bd680ef3ed971c"), "Name" : "Ninad", "Message" : "Hello everybody" }

**Inserting a Document**

To insert a document into a collection we can use the [insert()](http://api.mongodb.org/python/current/api/pymongo/collection.html#pymongo.collection.Collection.insert) method:

Code to insert data in Mongodb with python:

blog1={"Name":"Advait Railkar","Message":"First blog of Advait"}

>>> blog=db.blog

>>> blog\_id=blog.insert(blog1)

content on Mongodb afther inserting:

> db.blog.find()

listing of all of the collections in our database:

>>> db.collection\_names()

[u'blog', u'system.indexes', u'posts']

>>>

blog3=db.blog

>>> blog3={"Name":"Rajesh Parlkar","Message":"First blog of Rajesh"}

>>> blog\_id=blog.insert(blog3)

>>>

> db.blog.find()

{ "\_id" : ObjectId("53a1405425bd680ef3ed971b"), "Name" : "Rani", "Message" : "Hello" }

{ "\_id" : ObjectId("53a1407125bd680ef3ed971c"), "Name" : "Ninad", "Message" : "Hello everybody" }

{ "\_id" : ObjectId("53a143a51f85d008cbe58c12"), "Message" : "First blog of Advait", "Name" : "Advait parlkar" }

{ "\_id" : ObjectId("53a1487f1f85d008cbe58c13"), "Message" : "First blog of Rajesh", "Name" : "Rajesh parlkar" }

>

>>> blog4={"Name":"Rajesh Parlakr","Message":"third blog of Rajesh"}

>>> blog.insert(blog4)

ObjectId('53a1490f1f85d008cbe58c15')

>>>

> db.blog.find()

{ "\_id" : ObjectId("53a1405425bd680ef3ed971b"), "Name" : "Rani", "Message" : "Hello" }

{ "\_id" : ObjectId("53a1407125bd680ef3ed971c"), "Name" : "Ninad", "Message" : "Hello everybody" }

{ "\_id" : ObjectId("53a143a51f85d008cbe58c12"), "Message" : "First blog of Advait", "Name" : "Advait Railkar" }

{ "\_id" : ObjectId("53a1487f1f85d008cbe58c13"), "Message" : "First blog of Rajesh", "Name" : "Rajesh Parlkar" }

{ "\_id" : ObjectId("53a148e01f85d008cbe58c14"), "Message" : "second blog of Rajesh", "Name" : "” Rajesh Parlkar" }

{ "\_id" : ObjectId("53a1490f1f85d008cbe58c15"), "Message" : "third blog of Rajesh ", "Name" : " Rajesh Parlakar" }

**Conclusion:** Interface between MongoDB and Python can be established to display contents of MongoDB collections through java code. Collections of MongoDB must based on Rfid, images and blogs.

**References:-**

MongoDB Commands: - http://docs.mongodb.org/manual/tutorial/

**Assignment No. 15**

Aggregation and indexing in MongoDB

**Title:-**Aggregation and indexing with suitable example using MongoDB

**Problem definition:-** Design at least 10 NoSQL queries that demonstrate use of aggregation and indexing in MongoDB.

**Learning Objectives:-**

1. Importance of indexing in faster retrieval.
2. Understanding difference between different types of indexes.
3. Use of aggregation in MongoDB.

**Learning Outcomes :-**

1. Use of aggregation functions and clauses.
2. Creation of indexes on appropriate keys.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System/any other Open-source OS

3. MongoDB for 64-bit OS

**Mathematical Model:-**

Let set U be the database

Let A be the set of Documents in a collection.

Let B be the set of key-value pairs on which we create indexes.

Then BA ,

If C denotes aggregated data of collection A , then CA such that

U

BC = ϕ or BC ≠ ϕ

**Test Cases:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Query with avg function & group by clause | Avg of values specified with group by clause |  |
| TC\_02 | Query with sum function & group by clause | sum of values specified with group by clause |  |
| TC\_03 | Query with min function & group by clause | Minimum of values specified with group by clause |  |
| TC\_04 | Query with max function & group by clause | maximum of values specified with group by clause |  |
| TC\_05 | Creating compound index on any 2 keys of documents | Index successfully created |  |
| TC\_06 | Creating text index on any 2 keys of documents | Index successfully created |  |
| TC\_07 | Creating multikey index on any 2 keys of documents | Index successfully created |  |

**Theory:-**

1. Aggregation:-Aggregationsare operations that process data records and return computed results. MongoDB provides a rich set of aggregation operations that examine and perform calculations on the data sets. Running data aggregation on the [mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod" \o "mongod) instance simplifies application code and limits resource requirements. Like queries, aggregation operations in MongoDB use [collections](http://docs.mongodb.org/manual/reference/glossary/#term-collection) of documents as an input and return results in the form of one or more documents.
2. Indexing:- Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must scan every document in a collection to select those documents that match the query statement. These collection scansare inefficient because they require [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) to process a larger volume of data than an index for each operation. Indexes are special data structures that store a small portion of the collection’s data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. Fundamentally, indexes in MongoDB are similar to indexes in other database systems. MongoDB defines indexes at the [collection](http://docs.mongodb.org/manual/reference/glossary/#term-collection) level and supports indexes on any field or sub-field of the documents in a MongoDB collection.

**Conclusion: -** Aggregation pipelining is a feature which allows user to apply various filters on a set of documents to obtainthe result of operations on group of data. Aggregation helps in grouping the relevant data to obtain statistical information like min, max, count etc. Indexes help in faster retrieval of data from collections. Various types of Indexes created in MongoDB help in better performance of the system.

**References:-**

MongoDB 2.6 Manual

**Assignment No. 16**

Map reduce operation in MongoDB

**Title:-**Map reduce operation with suitable example in MongoDB.

**Problem definition:-** Implement Map reduce operation for a collection in MongoDB.

**Learning Objectives:-**

Understanding of Map reduce operation.

**Learning Outcomes :-**

Use of Map reduce operation in aggregation.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System/ Any other open-source OS

3. MongoDB for 64-bit OS

**Mathematical Model:-**

Let set U be the database

Let A be documents in a collection.

Let B be the set of documents retrieved after we apply the query.

Let C1 and C2 be the set of key-value pairs retrieved after map operation.

Let D be the set of key-value pairs obtained after reduce operation.

Then ,

If C denotes aggregated data of collection A , then CA such that

U

D can be denoted as follows:-

D= C1C2

D is the final result obtained after mapping and reduction.

**Theory:-**

Map-reduce is a data processing paradigm for condensing large volumes of data into useful *aggregated* results. For map-reduce operations, MongoDB provides the [mapReduce](http://docs.mongodb.org/manual/reference/command/mapReduce/#dbcmd.mapReduce) database command.

{

stud\_id: “A101”

marks1: 56

div:”A”

}

{

stud\_id: “A101”

marks2:70

div : “A”

}

{

stud\_id: “B109”

marks1: 54

div: “B”

}

{

stud\_id: “A021”

marks1:48

div: “A”

}

{

stud\_id: “A101”

marks1: 56

div:”A”

}

{

stud\_id: “A101”

marks2:70

div : “A”

}

{

stud\_id: “A021”

marks1:48

div: “A”

}

**query**

**map**

{ “A101” , 56}

{ “A101” , 70 }

{ “A021” , 48}

{

\_id: “A101”

Value: 126

}

{

\_id: “A021”

Value: 48

}

Students

Figure: Map-Reduce operation

Figure shows Student collection, the query filters the documents based on the div key, the map operation emits the id of student and the marks scored, the reduce operation computes the sum of the marks obtained in reduction state.

In this map-reduce operation, MongoDB applies the map phase to each input document (i.e. the documents in the collection that match the query condition). The map function emits key-value pairs. For those keys that have multiple values, MongoDB applies the reduce phase, which collects and condenses the aggregated data. MongoDB then stores the results in a collection. Optionally, the output of the reduce function may pass through a finalize function to further condense or process the results of the aggregation.

All map-reduce functions in MongoDB are JavaScript and run within the [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) process. Map-reduce operations take the documents of a single [collection](http://docs.mongodb.org/manual/reference/glossary/#term-collection) as the input and can perform any arbitrary sorting and limiting before beginning the map stage. [mapReduce](http://docs.mongodb.org/manual/reference/command/mapReduce/#dbcmd.mapReduce) can return the results of a map-reduce operation as a document, or may write the results to collections.

**Conclusion:-** Map-reduce operation can be used to obtain aggregated result of key-value pairs in a document . The useful or required data can be filtered and mapped to obtain a final statistical value, which can be used for analysis.

**References:-**

MongoDB 2.6 Manual

**Assignment No. 17**

Indexing and querying in MongoDB

**Title:-**Indexing and querying with suitable example using MongoDB

**Problem definition:-** Design at least 10 NoSQL queries that demonstrate use of indexing and querying in MongoDB.

**Learning Objectives:-**

1. Importance of indexing in faster retrieval.
2. Understanding difference between different types of indexes.
3. Querying the collection based on indexes.

**Learning Outcomes:-**

Studying the improvement in performance of a query in terms of created indexes.

**Software and Hardware Requirement**

1. 64 bit machine

2. Windows 8 Operating System/ any other open-source OS

3. MongoDB for 64-bit OS

**Mathematical Model:-**

Let set U be the database

Let A1 be the set of Documents in a collection which has no indexes(Fig 1).

Let A2 be the set of Documents in a collection which has an index (fig 2).

Let B be the set of key-value pairs on which we create an index (fig 2).

Then BA , such that if a set of queries q1 is executed on A1 then it takes time t1.

If q1 is executed on A2 it takes time t2 such that t2<t1 q1

U

U

**Test Cases:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case no.** | **Input** | **Expected Output** | **Actual Output** |
| TC\_01 | Querying a collection **without any index** for a specific key-value pair (fig 1 of mathematical model) | Result of query in time t1 |  |
| TC\_02 | Querying a collection Querying a collection **without any index** for two key-value pair. (fig 1 of mathematical model) | Result of query in time t2 |  |
| TC\_03 | Querying a collection for multiple keys in a document.(fig 1 of mathematical model) | Result of query in time t3 |  |
| TC\_04 | Creating compound index on any 2 keys of documents . (fig 2 of mathematical model) | Index successfully created |  |
| TC\_05 | Creating text index on any 2 keys of documents. (fig 2 of mathematical model) | Index successfully created |  |
| TC\_06 | Creating multikey index on any 2 keys of documents. (fig 2 of mathematical model) | Index successfully created |  |
| TC\_07 | Querying a collection based on the index created in TC\_01. (fig 2 of mathematical model) | Time required for query execution(t7) should reduce as compared to TC\_01 i.e. t7<t1 |  |
| TC\_08 | Querying a collection based on the index created in TC\_02. (fig 2 of mathematical model) | Time required for query execution(t8) should reduce as compared to TC\_02 i.e. t8<t2 |  |
| TC\_09 | Querying a collection based on the index created in TC\_03. (fig 2 of mathematical model) | Time required for query execution(t3) should reduce as compared to TC\_03 i.e. t9<t3 |  |

**Theory:-**

Indexing:- Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must scan every document in a collection to select those documents that match the query statement. These collection scansare inefficient because they require [mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod" \o "mongod) to process a larger volume of data than an index for each operation. Indexes are special data structures that store a small portion of the collection’s data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. Fundamentally, indexes in MongoDB are similar to indexes in other database systems.

MongoDB defines indexes at the [collection](http://docs.mongodb.org/manual/reference/glossary/#term-collection) level and supports indexes on any field or sub-field of the documents in a MongoDB collection. Types of indexes in mongodb, are as follows:-

1. Single field index:- A single field index only includes data from a single field of the documents in a collection. MongoDB supports single field indexes on fields at the top level of a document and on fields in sub-documents.
2. Compound Index:- A compound index includes more than one field of the documents in a collection.
3. Multikey index:- A multikey index references an array and records a match if a query includes any value in the array.
4. Geospatial index:-Geospatial indexes support location-based searches on data that is stored as either GeoJSON objects or legacy coordinate pairs.
5. Text Indexes:- Text indexes supports search of string content in documents.
6. Hashed Indexes:-Hashed indexes maintain entries with hashes of the values of the indexed field.

**Conclusion:-** Indexes help in faster retrieval of data. When a query is executed on a database with an appropriate index on appropriate keys, the query takes less time to execute as compared to a query executed on a collection with no index.

**References:-**

MongoDB 2.6 Manual