MySQL - Repair Tables

MySQL Repair Table Statement

There can be scenarios where tables in databases can become corrupted due to various reasons such as hardware failures, software bugs, or unexpected server crashes. When this situation happens, we cannot be able to access or manipulate the data in those tables because of data inconsistencies or errors.

In such situations, to repair those currupted tables, we use the MySQL REPAIR TABLE statement. This statement works for only certain engines such as MyISAM, etc.

Syntax :-

```
REPAIR [NO_WRITE_TO_BINLOG | LOCAL]
TABLE tbl_name [, tbl_name] ...
[QUICK] [EXTENDED] [USE_FRM]
```

```
CREATE TABLE CUSTOMERS (
   ID INT AUTO_INCREMENT,
   NAME VARCHAR(20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY DECIMAL (18, 2),
   PRIMARY KEY (ID)
);

INSERT INTO CUSTOMERS
(ID,NAME,AGE,ADDRESS,SALARY) VALUES
(1, 'Ramesh', 32, 'Ahmedabad', 2000.00 ),
(2, 'Khilan', 25, 'Delhi', 1500.00 ),
(3, 'Kaushik', 23, 'Kota', 2000.00 ),
(4, 'Chaitali', 25, 'Mumbai', 6500.00 ),
(5, 'Hardik', 27, 'Bhopal', 8500.00 ),
(6, 'Komal', 22, 'Hyderabad', 4500.00 ),
(7, 'Muffy', 24, 'Indore', 10000.00 );
```

Assume the above created table is corrupted and we are using the REPAIR TABLE statement to repair it.

```
REPAIR TABLE CUSTOMERS;
```

```
7 rows in set (0.00 sec)

mysql> REPAIR TABLE CUSTOMERS;
+-----------------+--------+----------+-------------------------------------------------------------+
| Table           | Op     | Msg_type | Msg_text                                                    |
+-----------------+--------+----------+-------------------------------------------------------------+
| repair.customers | repair | note     | The storage engine for the table doesn't support repair |
+-----------------+--------+----------+-------------------------------------------------------------+
1 row in set (0.01 sec)
```

To repair the table, We need to change the table's engine to **MyISAM** because it supports the REPAIR TABLE statement.

```
ALTER TABLE CUSTOMERS ENGINE = MyISAM;
```

```
REPAIR TABLE CUSTOMERS;
```

```
mysql> REPAIR TABLE CUSTOMERS;
+------------------+--------+----------+----------+
| Table            | Op     | Msg_type | Msg_text |
+------------------+--------+----------+----------+
| repair.customers | repair | status   | OK       |
+------------------+--------+----------+----------+
1 row in set (0.01 sec)
```

Repairing multiple tables

In MySQL, we can also repair multiple tables and get the results using the REPAIR TABLE Statement. To do this, we just need to list the names of the tables we want to repair, separating them with commas.

Example

Let us create three different tables with the names **Test1**, **Test2**, and **Test3** using the following CREATE TABLE statements –

```
CREATE TABLE Test1(ID INT, Name VARCHAR(255));
CREATE TABLE Test2(ID INT, Name VARCHAR(255));
CREATE TABLE Test3(ID INT, Name VARCHAR(255));
```

Assume the above three tables are corrupted. Change the engine of these tables to MyISAM to repair them with REPAIR TABLE statement –

```
ALTER TABLE Test1 ENGINE = MyISAM;
ALTER TABLE Test2 ENGINE = MyISAM;
ALTER TABLE Test3 ENGINE = MyISAM;
```

```
REPAIR TABLE Test1, Test2, Test3;
```

Repair Table Options

We have various optional clauses to use with REPAIR TABLE such as QUICK, EXTENDED, and, USE_FRM clause. Let us discuss them one by one with suitable examples.

QUICK Clause

The QUICK clause is the is the default and it is most commonly used with REPAIR TABLE. If you specify the QUICK clause, MySQL will repair the table without re-creating it. −

Example

In the following example, we are using the QUICK clause with the REPAIR TABLE statement to repair the CUSTOMERS table

```
REPAIR TABLE CUSTOMERS QUICK;
```

```
mysql> REPAIR TABLE CUSTOMERS;
+------------------+--------+----------+----------+
| Table            | Op     | Msg_type | Msg_text |
+------------------+--------+----------+----------+
| repair.customers | repair | status   | OK       |
+------------------+--------+----------+----------+
1 row in set (0.01 sec)
```

EXTENDED Clause

If we specify the ***EXTENDED clause***, MySQL not only repairs the table but also rebuilds the index and optimizes the table structure.

Note: The EXTENDED clause is a more time-consuming compared to QUICK clause.

Example

In the following example, we are using the EXTENDED clause with the REPAIR TABLE statement to repair the CUSTOMERS table.

```
REPAIR TABLE CUSTOMERS EXTENDED;
```

USE_FRM clause

We can use the ***USE_FRM clause***, in case the MYI index file is missing. If you provide this clause the .NYI file will be recreated using information from the data dictionary −

Example

Here, we are using the USE_FRM clause with the REPAIR TABLE statement to repair the CUSTOMERS table.

```
REPAIR TABLE CUSTOMERS USE_FRM;
```

```
mysql> REPAIR TABLE CUSTOMERS USE_FRM;
+------------------+--------+----------+----------------------------------+
| Table            | Op     | Msg_type | Msg_text                         |
+------------------+--------+----------+----------------------------------+
| repair.CUSTOMERS | repair | warning  | Number of rows changed from 0 to 7 |
| repair.customers | repair | status   | OK                               |
+------------------+--------+----------+----------------------------------+
2 rows in set (0.02 sec)
```

Repairing table Using a Client Program

Besides repairing a table in a MySQL database with a MySQL query, we can also use a client program to perform the REPAIR TABLE operation.

Syntax

To repair a table in a MySQL database through a Python program we need to execute the **Repair Table** statement using the **execute()** function of the MySQL **Connector/Python** as –

sql="REPAIR TABLE table_name";

cursorObj.execute(sql);

```
import mysql.connector
#establishing the connection
connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='password',
    database='tut'
)
table_name = 'tutorials_tbl_temp'
#Creating a cursor object
cursorObj = connection.cursor()
repair_table_query = f"REPAIR TABLE {table_name}"
cursorObj.execute(repair_table_query)
print(f"Table '{table_name}' is repaired successfully.")
# Fetch and consume any remaining results from the cursor
# ensuring that there are no unread results before closing the cursor.
for _ in cursorObj:
    pass
cursorObj.close()
connection.close()
```

```
import mysql.connector
#establishing the connection
connection = mysql.connector.connect(
   host='localhost',
   user='root',
   password='password',
   database='tut'
)
table_name = 'tutorials_tbl_temp'
#Creating a cursor object
cursorObj = connection.cursor()
repair_table_query = f"REPAIR TABLE {table_name}"
cursorObj.execute(repair_table_query)
print(f"Table '{table_name}' is repaired successfully.")
# Fetch and consume any remaining results from the cursor
# ensuring that there are no unread results before closing the cursor.
for _ in cursorObj:
   pass
cursorObj.close()
connection.close()
```