MySQL Transaction

Summary: in this tutorial, you will learn about **MySQL transaction** and how to use MySQL COMMIT statement and MySQL ROLLBACK statement to manage transactions in MySQL.

Introducing to MySQL Transaction

To understand what a transaction in MySQL is, let's take a look at an example of adding a new sales order in our sample database. The steps of adding a sales order are as described as follows:

- Query the latest sales order number from the `orders` table, and use the next sales order number as the new sales order number.
- Insert a new sales order into the `orders` table for a given customer.
- Insert new sales order items into the `orderdetails` table.
- Get data from both table `orders` and `orderdetails` tables to confirm the changes

Now imagine what would happen to your data if one or more steps above fail because of database failure such as table lock security? If the step of adding order items into `orderdetails` table failed, you would have an empty sales order in your system without knowing it. Your data may not be integrity and the effort you have to spend to fix it is tremendous.

How do you solve this problem? That's why the transaction processing comes to the rescue. MySQL transaction enables you to execute a set of MySQL operations to ensure that the database never contains the result of partial operations. In a set of operations, if one of them fails, the rollback occurs to restore the database. If no error occurred, the entire set of statements is committed to the database.

Using MySQL Transaction

Let's review the most important MySQL transaction statements before we are using them for t ding sales order in the example above.

```
CREATE / ALTER / DROP DATABASE
CREATE /ALTER / DROP / RENAME / TRUNCATE TABLE
CREATE / DROP INDEX
CREATE / DROP EVENT
CREATE / DROP FUNCTION
CREATE / DROP PROCEDURE
…
```

To write the changes into the database within a transaction, you use the `COMMIT` statement. It is important to note that MySQL automatically commits the changes to the database by default.

To force MySQL not to commit changes automatically, you use the following statement:

```
SET autocommit = 0;
```

MySQL transaction example

In order to use MySQL transaction, you first have to break your MySQL statements into logical portions and determine when data should be committed or rolled back.

Let’s take a look an example of using MySQL transaction to add new sales order into our sample database above and add the transaction processing steps:

- Start a transaction using `START TRANSACTION` statement.
- Get latest sales order number from the `orders` table, and use the next sales order number as the new sales order number.
- Insert a new sales order into the `orders` table for a given customer.
- Insert new sales order items into the `orderdetails` table.
- Commit changes using `COMMIT` statement.
- Get data from both table `orders` and `orderdetails` tables to confirm the changes.

```
-- get latest order number
select @orderNumber := max(orderNUmber)
from orders;
-- set new order number
set @orderNumber = @orderNumber  + 1;

-- insert a new order for customer 145
insert into orders(orderNumber,
                   orderDate,
                   requiredDate,
                   shippedDate,
                   status,
                   customerNumber)
values(@orderNumber,
       now(),
       date_add(now(), INTERVAL 5 DAY),
       date_add(now(), INTERVAL 2 DAY),
       'In Process',
        145);
-- insert 2 order line items
insert into orderdetails(orderNumber,
                         productCode,
                         quantityOrdered,
                         priceEach,
                         orderLineNumber)
values(@orderNumber,'S18_1749', 30, '136', 1),
      (@orderNumber,'S18_2248', 50, '55.09', 2);
-- commit changes
commit;

-- get the new inserted order
select * from orders a
inner join orderdetails b on a.ordernumber = b.ordernumber
where a.ordernumber = @ordernumber;
```

In this tutorial, you've learned how to use MySQL transaction statements that include `START TRANSACTION` `COMMI`, and `ROLLBACK` to manage transactions in MySQL to protect data integrity.

MySQL JDBC Transaction

MySQL Table Locking

Tweet in 0

« Previous Tutorial: MySQL REPLACE

Next Tutorial: MySQL Table Locking »

Search this website …

MYSQL QUICK START

What Is MySQL?

Install MySQL Database Server

Download MySQL Sample Database

Load Sample Database

MYSQL DATA MANIPULATION

MySQL SELECT

MySQL DISTINCT

MySQL WHERE

MySQL AND

MySQL OR

HOME BASIC MYSQL ADVANCED MYSQL INTERFACES TIPS TRYIT

Perl MySQL Tutorial

MySQL JDBC Tutorial

OTHER TUTORIALS

MySQL Administration

MySQL Full-Text Search

MySQL Cheat Sheet

MySQL Books and Video Training

MySQL Hosting

MySQL Resources

MySQL MINUS

MySQL Join Made Easy

How To Delete Duplicate Rows in MySQL

How To Find Duplicate Values in MySQL

MySQL LAST_DAY Function

MySQL EXTRACT Function

MySQL SYSDATE Function

MySQL CURDATE Function

ABOUT MYSQL TUTORIAL WEBSITE

MySQLTutorial.org is a website dedicated to MySQL database. We regularly publish useful MySQL tutorials to help web developers and database administrators learn MySQL faster and more effectively.

All MySQL tutorials are practical and easy-to-follow, with SQL script and screenshots available. More About Us

SITE LINKS

About Us

Contact Us

Request a Tutorial

Privacy Policy

HOME BASIC MYSQL ADVANCED MYSQL INTERFACES TIPS TRYIT