

Practical MySQL

Session 10

Partitioning

For Aptech Centre Use Only

Session Overview

- Define partitioning and its different types
- Describe how partitioning works
- List the benefits of partitioning
- Explain partitioning maintenance, pruning and selection
- Outline the restrictions and limitations of partitioning

For Apteck Centre Use Only

What is Partitioning?

Partitioning is the process of dividing the rows of a table into separate tables in different locations.

In MySQL, Partitioning is used to split or partition the records of a table as rows into separate tables in different locations. Even though it is partitioned, it is treated as a single table.

Partitioning function is the rule that is set to accomplish the division of table data.

Partitioning is supported by MySQL only in InnoDB and NDB storage engines.

Working of Partitioning

The user must ensure that the look-up table goes to the correct partition or group of partition.

`PARTITION BY` divides the result set into partitions and further performs queries on it.

Look-up table can be defined as array of data that maps input values to output values. It contains the data that matches an ID in view.

Uses of Partitioning

Partitioning improves scalability and optimizes the performance.

It provides a mechanism for dividing data by usage pattern.

Due to partitioning, queries that access only a fraction of the data can run faster.

It stores huge data in one table which can be kept on a single disk or in file system.

It supports to keep more control to manage the data in the database.

Types of Partitioning

Horizontal Partitioning: This partitioning involves the rows of table to be split into one or multiple tables depending on the criteria specified by the user.

Vertical Partitioning: Vertical Partitioning splits the table into multiple tables with only few columns from the original table.

Partitioning By Range: Range partitioning partitions the rows of a table on column values within the provided range.

Partitioning by List: List Partitioning is similar to Range Partitioning. Here, the partitioning is defined depending on columns.

Partitioning By Hash

In Hash partitioning, data distribution is done on the basis of predefined number of partitions.



In simple words, a table splits into that number of times as the user-defined value.

It distributes the data evenly in partitions. It can be performed by `PARTITION BY HASH` clause.

The number of partitions into which the table must be partitioned along with the column name to be hashed must be provided.

Partitioning By Key

Key partitioning is identical to HASH partitioning. It also requires a user-defined expression. MySQL provides its own hashing function for partitioning using key.

MySQL uses its internal hashing function to perform `PARTITION BY KEY` clause with other storage engines.

In this case, a key is used instead of HASH. It takes only a list of zero or more column names as parameters.

If no column is specified for partition, then the primary key is automatically considered as partitioning key.

Partitioning by Column

In this, multiple columns are used as partitioning keys. It serves the purpose of partitioned rows and determines which partitions are to be validated for matching of rows. It is divided into two parts. Range and List Columns supports following data types:

Integer Types such as INT (INTEGER), MEDIUMINT, BIGINT, TINYINT, and SMALLINT.

String Types such as CHAR, VARCHAR, BINARY, and VARBINARY

Date types such as DATE and DATETIME.

Range, List, and Sub Partitioning

Partitioning By Range specifies the partitions using ranges based on values of different columns.

```
CREATE TABLE part (Column1 INT, Column2 CHAR (5),  
Column3 INT, Column4 INT)
```

LIST Columns Partitioning involves single or multiple columns taken as partition keys.


Sub partition also called composite partitioning, splits the partitioned tables into further partitions.

Benefits of Partitioning

Partitioning optimizes the query performance by scanning only portions of tables that satisfy the condition or criteria of the query.



Huge amounts of data can be stored in one table rather than on a single hard disk or file system and manage the file organizations as well.



Partitioning makes drive data more secure from malware attacks.



Sub partition also called composite partitioning, splits the partitioned tables into further partitions.

Partition Management

Management of RANGE and LIST Partitions: It is basically adding and dropping of partitions. RANGE and LIST partitions are used to create tables. Similarly, range and list partitions can be added and dropped.

By using ALTER TABLE command: In this management, users can perform various operations on existing partitions such as adding, dropping, redefining, merging, or splitting. All these operations can be performed using ALTER TABLE statement.

Management of HASH and KEY Partitions: Partition by HASH and Partition by KEY are similar to each other in making changes in partitioning set up.

Partition Maintenance

Rebuilding Partitions: The process of partition maintenance includes multiple things such as rebuilding partitions, optimizing the partitions so on.

Optimizing Partitions: The partitioned data can be defragmented using optimizing partitions.

Analyzing Partitions: Partitions can be analyzed by using SQL Statements such as CHECK TABLE.

Repairing Partitions: Partitions which are corrupted during partitions can be repaired using REPAIR PARTITION command.

Checking of Partitions: Partitions can contain errors that require checking.

Partition Pruning

Partition pruning is basically the technique of optimizing. It mentions that partitions must not be scanned if there are no matching values.

It removes or cuts undesirable partitions.

The concept of data warehousing uses this as an essential performance feature.

The optimizer checks for useless partitions using `WHERE` and `FROM` clause, which can further be removed.

In this process, scanning certain partitions not having any matching conditions can be avoided.

Partition Selection

It is similar to the concept of partition pruning. In this process, only specific partitions are checked for matches, but it may differ in two key aspects that are as follows:

Partitions are checked which are specified by issuer of the statement. It is different from pruning as it is done automatically in pruning.

Partition selection is done for all types of queries and a number of DML statements, whereas, in pruning it is only done for queries.

Following are some SQL statements that support partition selection:

DELETE

LOAD
DATA

LOAD XML

SELECT

INSERT

UPDATE

REPLACE

Restrictions and Limitations on Partitioning

While doing partitions it is important to understand their limitations and restrictions so that the process is successfully completed. Limitations on partitioning are as follows:

8192 Partitions and sub partitions in MySQL are possible. It is applicable to all versions of MySQL.

There is a limit of 1024 partitions per table as well.

Table partitioning can be performed if a storage engine supports it.

All partitions must be from the same storage engine.

Partition table should not contain any foreign key.

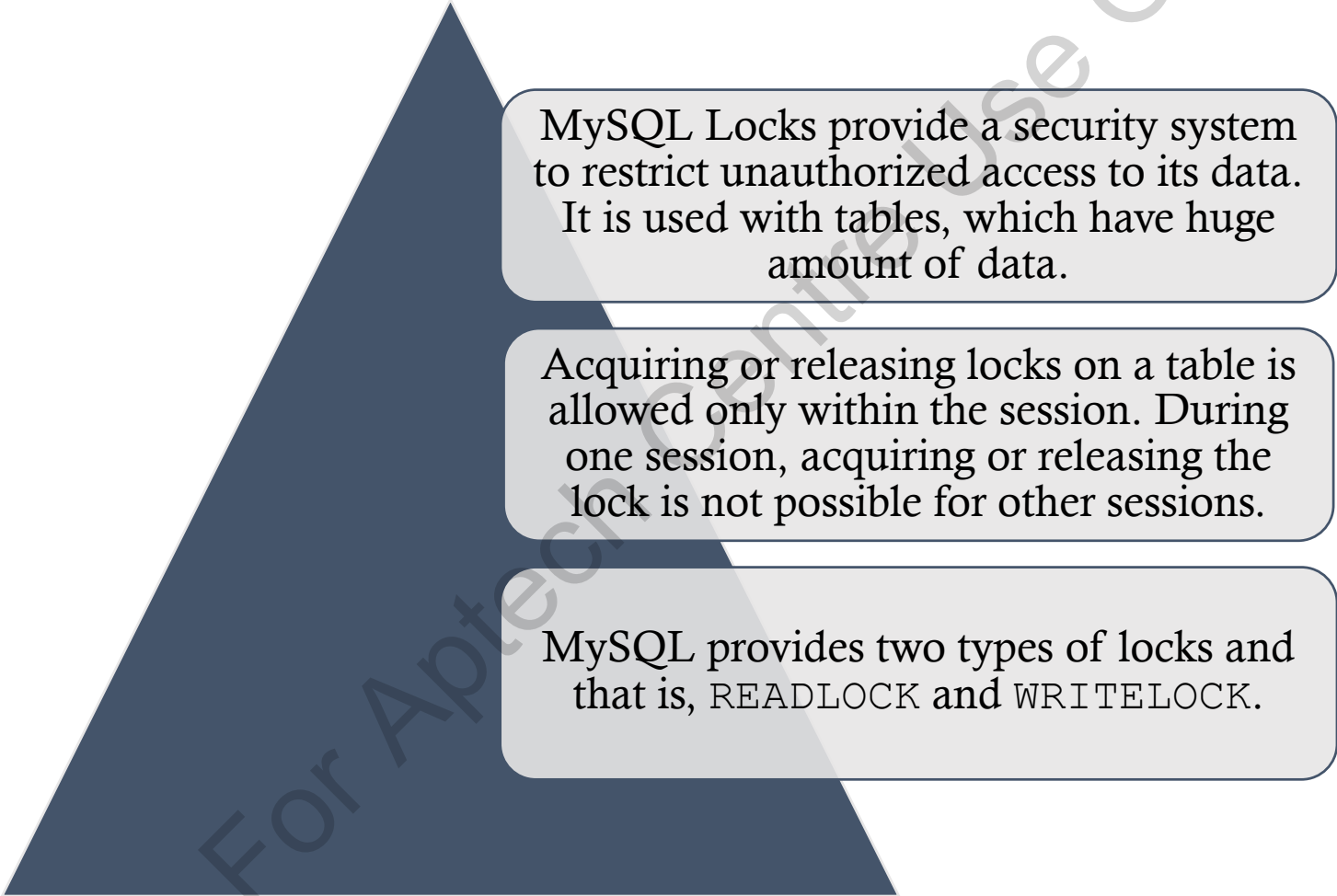
MySQL LOCK Table

Row-Level Locking: Row-level locking means that only the row that is accessed by an application will be locked. The other rows belonging to the same page are released.

Read Lock or Shared Lock: In this type of lock, any row having a read lock or shared lock can only be read by other sessions but cannot be written or have update statements performed on their locked data.

Write Lock or Exclusive lock: In this type of locking, other sessions cannot read or write the locked data.

MySQL Locks



MySQL Locks provide a security system to restrict unauthorized access to its data. It is used with tables, which have huge amount of data.

Acquiring or releasing locks on a table is allowed only within the session. During one session, acquiring or releasing the lock is not possible for other sessions.

MySQL provides two types of locks and that is, READLOCK and WRITELOCK.

READ AND WRITE LOCK

READ LOCK

- More than one session can acquire a READ LOCK on a single table.

WRITE LOCK

- A session holding the write lock on a table is allowed to read data from the table or write data into the table.

READ VS. WRITE LOCKS

- READLOCKS are shared and hence, they prevent a write lock to become acquired. WRITELOCKS are exclusive locks they allow you to have many read statement, but only one write statement is allowed.

Summary

- Partition allows users to divide portions of individual tables in a file system or partition according to the rules.
- Vertical Partitioning splits the table into multiple tables with very few columns from the original table.
- Sub partitioning splits each partition in a partition table.
- Rebuilding Partitions is a part of Partition Maintenance.
- Users can perform operations such as `CHECK TABLE`, `REPAIR TABLE`, `OPTIMIZE PARTITION`, `REBUILD PARTITION`, `REPAIR PARTITION`, and so on for partition maintenance.
- Partitions are analyzed using SQL statements such as `CHECK TABLE`.
- Locking mechanism is associated with tables that restrict unauthorized access of data inside the table.