S04-L01

Intro: Partitions & why to use them?

S04-L01 Intro: Partitions & why to use them?

Split one table to multiple partitions

- Group data that are logically and cohesively in the same class
- Faster data select
- More scalable way to manage data
- Easier delete of old data

```sql
CREATE TABLE `test` (`id` INT, `purchased` DATE)
  PARTITION BY RANGE( YEAR(`purchased`) )
  SUBPARTITION BY HASH( TO_DAYS(`purchased`) )
  SUBPARTITIONS 2 (
    PARTITION p0 VALUES LESS THAN (1990),
    PARTITION p1 VALUES LESS THAN (2000),
    PARTITION p2 VALUES LESS THAN MAXVALUE
  )
```

S04-L01 Partition types

RANGE partitioning: This type of partitioning assigns rows to partitions based on column values falling within a given range.

LIST partitioning: Similar to partitioning by RANGE, except that the partition is selected based on columns matching one of a set of discrete values.

HASH partitioning: A partition is selected based on the value returned by a user-defined expression that operates on column values in rows to be inserted into the table. The function may consist of any expression valid in MySQL that yields a nonnegative integer value.

KEY partitioning: This type of partitioning is similar to partitioning by HASH, except that only one or more columns to be evaluated are supplied, and the MySQL server provides its own hashing function. These columns can contain other than integer values