MySQL

I. Overview

Revise a few definitions related to the database. **Relational DataBase Management System (RDBMS) Terminology**

- **Database** − A database is a collection of tables, with related data.
- **Table** − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column** − One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row** − A row (= tuple, entry, or record) is a group of related data, for example, the data of one subscription.
- **Redundancy** − Storing data twice, redundantly to make the system faster.
- **Primary Key** − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- **Foreign Key** − A foreign key is a linking pin between two tables.
- **Compound Key** − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index** − An index in a database resembles an index at the back of a book.
- **Referential Integrity** − Referential Integrity makes sure that a foreign key value always points to an existing row.

You can grab the general information about MySQL here, following aspect you maybe want to know:

- MySQL is released under an open-source license
- Written in C and C++.
- Provides transactional and nontransactional storage engines.
- Uses very fast B-tree disk tables (`MyISAM`) with index compression.
- Implements in-memory hash tables, which are used as temporary tables.
- Uses a standard form of the well-known SQL data language.
- supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

Install:

- Docker: https://hub.docker.com/_/mysql
- Manual: https://dev.mysql.com/doc/refman/8.0/en/installing.html

II. Storage engine

Get the knowledge about the pros and cons of various MySQL storage engines, and pay attention at the InnoDB storage engine (the default and most general-purpose storage engine, and Oracle recommends using it for tables except for specialized use case), introduction selection is enough, doesn't need to get deep at the moment.

III. Data types

Knowledge about the basic data type and how to use it effectively.

MySQL support many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, `FLOAT`, `DOUBLE`, `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY`, `TEXT`, `BLOB`, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`, `SET`, `ENUM`, and OpenGIS spatial types.

There are five data type categories and divided into two types fixed-length and variable-length types. You can read detail about data type categories and data size here.

And please pay attention to:

- Data type storage requirement so you can choose the effective data type.
- Out-of-Range and Overflow Handling for numeric
- Related function to those data type

Be aware that some special data like emoji,.. must change the charset to store and processing:

- Charset unicode
- How to support full Unicode in MySQL databases

Choosing Optimal Data Types

Smaller is usually better.

Smaller data types are usually faster, because they use less space on the disk, in memory, and in the CPU cache. They also generally require fewer CPU cycles to process.

Make sure you don't underestimate the range of values you need to store, though, because increasing the data type range in multiple places in your schema can be a painful and time-consuming operation. If you're in doubt as to which is the best data type to use, choose the smallest one that you don't think you'll exceed.

Simple is good.

Fewer CPU cycles are typically required to process operations on simpler data types. For example, integers are cheaper to compare than characters, because character sets and collations (sorting rules) make character comparisons complicated. Here are two examples: you should store dates and times in MySQL's built-in types instead of as strings, and you should use integers for IP addresses.

III. Transaction and locking

This is the most important selection, please pay attention and deeply understand, research following topic and keyword.

This section relates to InnoDB storage engine locking and transaction cause only InnoDB support transactions in MySQL.

What's transaction and purpose? A transaction is a very small unit of a program and it may contain several low-level tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

Please deeply understand:

- ACID properties
- Isolation level and read phenomena, InnoDB isolation

Locking impact on performance, so you must understand those lock type and use case: InnoDB Locking, InnoDB locks set.

The lock may lead to Deadlock, be aware, and try to avoid this situation.

MySQL transaction syntax: Transaction and locking statements

VI. Best practices

- Top 10 MySQL best practices
- MySQL Optimization
- Use Prepared statement
- Java lib: https://github.com/brettwooldridge/HikariCP

VII. References

- https://www.tutorialspoint.com/mysql/mysql-introduction.htm
- https://www.oreilly.com/library/view/high-performance-mysql/9781449332471/ch04.html
- https://www.mysqltutorial.org/mysql-table-locking/#:~:text=A%20lock%20is%20a%20flag,table%20locks%20only%20for%20itself.