MySQL 开发规范

目录

一、基础类.....3
二、命名规范.....3
三、表设计.....4
三、字段设计.....4
四、索引策略.....5
五、SQL 语句.....5
六、脚本规范.....6

一、基础类

1、必须使用 InnonDB 存储引擎。

解读：支持事务、行级锁、并发性能更好。

2、新库（表）默认使用 utf8mb4 字符集

解读：emoji 表情以及部分不常见汉字在 utf8 下会表现为乱码。

3、数据库表、表字段必须加入中文注释。

4、避免使用存储过程、视图、触发器、Event。

解读：业务逻辑放到服务层具备更好的扩展性。

5、禁止在数据库中存储文件或者照片。

解读：大文件和照片存储在文件系统，数据库里存 URI。

二、命名规范

1、只能使用英文字母（小写）、数字、下划线，单词之间必须以下划线作为分隔符。

2、命名中不允许出现 MySQL 数据库中的关键字和保留字。MySQL5.7 关键字和保留字查看：https://dev.mysql.com/doc/refman/5.7/en/keywords.html。

3、数据库对象的命名要能做到见名知意，建议不超过不超过 32 个字符。

4、不同数据库对象的命名以对应的类型缩写作为前缀(表名使用模块明为前缀）。

表名：{模块名}_xxx
临时表：tmp_xxx
主键索引：pk_{表名缩写}_xxx
唯一索引：uniq_{表名缩写}_xxx
非唯一索引名：idx_{表名缩写}_xxx

三、表设计

1、表列数目不宜过多，建议不超过 50 个。

2、表必须有主键。

解读：

主键字段必须不能为空值，唯一且不会改变;

主键值要求递增（如自增主键），而不是随机的（如 UUID)。主键递增，可以提高插入性能，减少索引分页和碎片的产生;

主键尽量选择较短的数据类型，Innodb 引擎的普通索引会保存主键的值,较短的数据类

型可以有效的减少索引的磁盘空间，提高索引的缓存效率。

3、表必须有 create_time、update_time 二个字段，采用数据库时间更新。

4、禁止使用外键，如果有外键完整性约束，需要应用程序控制。

解读：外键会导致表与表之间耦合，update 与 delete 操作都会涉及相关联的表，高并发情况下容易引起数据库性能问题。

5、表只能有一个自增列，具有业务含义的字段不能设置为自增列，自增列命名为 ID。

6、禁止使用 MySQL 分区表。

解读：InnoDB 存储引擎分区表不适用于 OLTP 应用。

7、删除表记录时建议采用逻辑删除的方式。

三、字段设计

1、字段的类型不建议使用集合、枚举、位图类型。

解读：增加新的枚举值要做 DDL 操作。ENUM 的内部实际存储就是整数，需要该类型时可使用 TINYINT 替代。

2、字段类型不建议使用 TEXT、BLOB 类型。

解读：会浪费更多的磁盘和内存空间，非必要的大量大字段查询会淘汰掉热数据，导致内存命中率急剧降低，影响数据库性能。若业务场景确需使用，需将相关字段放入子表中。

3、业务上不允许为空的字段需定义为 NOT NULL 并且提供默认值（自增列和主键列不提供默认值）。

4、如果是自增字段，使用无符号类型。

解读：无符号类型能达到的上限更大。

5、字段须定义合适的数据类型，且尽量选择较短的数据类型。

示例：不使用小数存储货币；使用 varchar(20)存储手机号；用好数值类型 tinyint(1Byte)、smallint(2Byte)、mediumint(3Byte)；使用 int 而不是 char(15)存储 ip。

6、不同表中具有相同业务含义的字段，需使用一致的数据类型。

解读：表连接时不能有效使用连接列上的索引。

四、索引策略

1、字段是否需要索引与表的数据量无关，字段的区分度高，并且会用于过滤数据，就应该建立索引。

2、合理使用复合索引，建立复合索引时，把等值查询的列放前面，范围查询的列放后面。如果需要排序，综合考虑列的顺序。

解读：复合索引中，非等值查询之后的字段，不能有效缩减扫描数据行数。

3、单个索引(复合索引)的列数不要超过 5 个。

解读：被索引的字段越大（越多），索引 B+数越高，会影响读取效率。

4、禁止在更新十分频繁、区分度不高的字段上建立索引。

解读：更新会变更 B+树，更新频繁的字段建立索引会大大降低数据库性能；比如“性别”这种区分度不高的字段，建立索引是没有什么意义的，不能有效过滤数据。

5、如果字段值是唯一的，创建唯一索引。

6、不在索引列做列运算（如 select ... from tab where year(col)>2020;）。

解读：不能有效使用索引。

7、不是针对每一条 SQL，我们都创建一个最优的索引，而是综合考虑表的读取模式，设计出合适的索引。

五、SQL 语句

1、拒绝 3B(big)，大 SQL，大事务，大批量。

2、不要使用 SELECT *或读取不需要的列。

解读：会增加 CPU、IO 消耗，不能有效的利用覆盖索引，使用 SELECT *容易在增加或者删除字段后出现程序 BUG。

3、禁止使用 INSERT INTO t_xxx VALUES(xxx)，必须显示指定插入的列属性。

解读：容易在增加或者删除字段后出现程序 BUG，显示指定方便确认语句是否正确。

4、禁止在 WHERE 条件的列上使用函数或者表达式。

解读：对列进行运算不能有效使用该列上的索引。

5、注意列上的隐式类型转换，隐式类型转换会让索引失效。

6、不建议使用负向查询（NOT、!=、<>、!<、!>、NOT IN、NOT LIKE）和以%开头的模糊查询（特别是在索引字段上）。

7、建议 OR 改写为 IN 或 UNION（UNION ALL）。

8、禁止不带条件的全表更新和删除。

9、禁止两表间没有连接条件的 JOIN，JOIN 列必须创建索引。

10、Where 子句中 in 的值禁止一次性输入超过 500 个。

六、脚本规范

1、对同一个表的多次 alter 操作必须合并为一个 SQL 一次操作。

2、通过数据库脚本进行 delete，update，drop，truncate 之前必须提供备份和回退脚本，并且核对数据是否需要备份和备份是否成功。

3、非维护窗口禁止使用 insert/delete/update　into/from　table_a　select * from table_b where 操作，此类语句会锁 TABLE_B。

4、大批量操作必须分段提交，每次提交不得超过 1 万条。

5、DDL 脚本和 DML 脚本须分开提供。

6、提供的脚本必须经过测试，确保能正确执行。