目录

1. 表设计

1、库名、表名、字段名必须使用小写字母，“_”分割

2、库名、表名、字段名必须不超过 12 个字符

3、库名、表名、字段名见名知意,建议使用名词而不是动词,对象名称尽可能简单；过长或冗长的名称增加了优化器在做硬解析时的时间，

从而影响性能

4、必须使用 InnoDB 存储引擎

5、存储精确浮点数必须使用 DECIMAL 替代 FLOAT 和 DOUBLE;禁止使用 float、double 类型，建议使用 decimal、int、bigint 替代

6、建议使用 UNSIGNED 存储非负数值

7、建议使用 INT UNSIGNED 存储 IPV4

8、整形定义中不添加长度，比如使用 INT，而不是 INT(4)

9、使用短数据类型，比如取值范围为 0-80 时，使用 TINYINT UNSIGNED

10、不建议使用 ENUM 类型，使用 TINYINT 来代替

11、尽可能不使用 TEXT、BLOB 类型

12、VARCHAR(N)，N 表示的是字符数不是字节数，比如 VARCHAR(255)，可以最大可存储 255 个汉字，需要根据实际的宽度来选择 N

13、VARCHAR(N)，N 尽可能小，因为 MySQL 一个表中所有的 VARCHAR 字段最大长度是 65535 个字节，进行排序和创建临时表一类的

内存操作时，会使用 N 的长度申请内存

14、表字符集选择 UTF8

15、使用 VARBINARY 存储变长字符串

16、存储年使用 YEAR 类型

17、存储日期使用 DATE 类型

18、存储时间（精确到秒）建议使用 TIMESTAMP 类型

解读:

TIMESTAMP 使用 4 字节,DATETIME 使用 8 个字节,能有效节约存储空间，数据变得更紧凑，能有性能提升。

补充，在既有 mysql5.5、mysql5.6、mysql5.7 同时存在的组织中，考虑到一致性建议用 datetime

19、建议字段定义为 NOT NULL

解读:

(1) 在含有空值的列优化器很难进行查询优化，而且对表索引时不会存储 NULL 值，所以如果索引的字段可以为 NULL，索引的效率会严重下降

(2) NULL 会使得索引、索引的统计信息以及比较运算更加复杂

(3) 字符类型字段,可以默认定义’N/A’字符

(4) 数字类型字段,可以默认定义 0

(5) 日期时间类型字段,可以默认为当前时间

20、将过大字段拆分到其他表中

21、禁止使用 blob、text、binary 类型保留大文本、文件、图片，建议使用其他方式存储（TFS/SFS），MySQL 只保存指针信息

22、表结构变更需要通知 DBA 审核与执行（考虑到性能与对客户的影响）

23、表被索引列必须定义为 not null，并设置 default 值

24、表必须包含 create_datetime 和 update_datetime 字段，即表必须包含记录的创建时间和修改时间字段

解读:

(1) 有利于做归档处理，优化性能

(2) 有利于安全性与审计方便在基于 MS、MM 的复制中修复数据差异

25、单条记录大小禁止超过 8K

解读:

(1) 在基于数字、100 个字符内的短字符、日期时间的表字段中，理论字段个数控制在 10 个以内最好

(2) 依据上述，10 个字段的表数据一定会比大于 10 个字段的表数据加载或查询速度快

(3) 在 8K 内的一行记录，至少做到一页存储了两行数据，没有跨页存储，在数据库存取速度上有明显性能提升

26、日志类数据不建议存储在 MySQL 上

解读:传统关系型数据库系统擅长处理事务（ACID）、擅长处理数字与运算。日志类属于超文本(fulltext)类，字符巨大,IO 性能需求很高，建议优先考虑 Hbase 或 MongoDB

27、单表一年内数据量超过 1000w 或数据容量超过 30G 考虑分表，且需要提前考虑历史数据归档或应用自行删除历史数据方案

28、禁止使用 varchar 类型作为主键语句设计

29、表必须定义主键，默认为 ID，整型自增，如果不采用默认设计必须咨询 DBA 进行设计评估

30、ID 字段作为自增主键，禁止在非事务内作为上下文作为条件进行数据传递

31、 建表时必须将 ROW_FORMAT=COMPRESSED 选项加上

解读:

(1) 仅此选项而言，单表访问速度至少有 10%提升

(2) 结合 innodb_compression_level 参数单表访问甚至有 20%以上的性能提升

2. 索引

1、非唯一索引必须按照“ix_字段名称”进行命名

2、唯一索引必须按照“uq_字段名称”进行命名

3、索引名称必须使用小写

4、索引中的字段数建议不超过 5 个

5、单张表的索引数量控制在 5 个以内

解读:

(1) 严重影响写性能，索引越多，IO 次数越多，与索引个数成正比，

(2) 存储引擎在数据落盘做顺序处理时增加了 n+1 的 IO 次数，耗用资源更多

(3) 过多的索引，优化器在生成执行计划时增加了索引 cardinality 的选择性难度，进而导致选择不到最优的索引

(4) 增加阻塞与死锁风险

6、唯一键由 3 个以下字段组成，并且字段都是整形时，使用唯一键作为主键

7、没有唯一键或者唯一键不符合 5 中的条件时，使用自增（或者通过发号器获取）id 作为主键

8、唯一键不和主键重复

9、索引字段的顺序需要考虑字段值去重之后的个数，个数多的放在前面

10、ORDER BY、GROUP BY、DISTINCT 的字段需要添加在索引的后面

11、使用 EXPLAIN 判断 SQL 语句是否合理使用索引，尽量避免 extra 列出现：Using File Sort，Using Temporary

12、UPDATE、DELETE 语句需要根据 WHERE 条件添加索引

13、不建议使用%前缀模糊查询，例如 LIKE '%weibo'

解读: 优化器对此 SQL 会生成全表扫描的执行计划

14、对长度过长的 VARCHAR 字段建立索引时，添加 crc32 或者 MD5 Hash 字段，对 Hash 字段建立索引

15、合理创建联合索引（避免冗余），(a,b,c) 相当于 (a) 、(a,b) 、(a,b,c)

16、合理利用覆盖索引

解读:

(1) 所有索引类型中速度最快,比 primary key 快

(2) 顺序 IO

(3) 不需做回表处理，少做一个算法

17、SQL 变更需要确认索引是否需要变更并通知 DBA

18、不建议在频繁更新的字段上建立索引

解读: 如果在读写比例为 1:1 的场景中，频繁更新的字段不得不要建索引来提升读的性能，但在做数据更新时，建议使用二级索引(secondary

index)先查询出主键，再根据主键进行数据更新

19、主键(primary key)不要用 UUID 类型

解读:

(1) MySQL 优化器针对 UUID 类型在 IO 层面会走随机读

(2) 非顺序存储

(3) 浪费掉一个表只有一个 primary key 的资源

20、禁止在选择性不高的字段上建索引

3. SQL语句

1、使用 prepared statement，可以提供性能并且避免 SQL 注入

2、SQL 语句中 IN 包含的值不应过多

3、UPDATE、DELETE 语句不使用 LIMIT

4、WHERE 条件中必须使用合适的类型，避免 MySQL 进行隐式类型转化

5、SELECT 语句只获取需要的字段

6、SELECT、INSERT 语句必须显式的指明字段名称，不使用 SELECT *，不使用 INSERT INTO table()

7、使用 SELECT column_name1, column_name2 FROM table WHERE [condition]而不是 SELECT column_name1 FROM table WHERE [condition]

和 SELECT column_name2 FROM table WHERE [condition]

8、WHERE 条件中的非等值条件（IN、BETWEEN、<、<=、>、>=）会导致后面的条件使用不了索引

9、避免在 SQL 语句进行数学运算或者函数运算，容易将业务逻辑和 DB 耦合在一起

10、INSERT 语句使用 batch 提交（INSERT INTO table VALUES(),(),()……），values 的个数不应过多

11、避免使用存储过程、触发器、函数等，容易将业务逻辑和 DB 耦合在一起，并且 MySQL 的存储过程、触发器、函数中存在一定的 bug

12、避免使用 JOIN

13、使用合理的 SQL 语句减少与数据库的交互次数

14、不使用 ORDER BY RAND()，使用其他方法替换

15、建议使用合理的分页方式以提高分页的效率

16、统计表中记录数时使用 COUNT(*)，而不是 COUNT(primary_key)和 COUNT(1)

17、禁止在从库上执行后台管理和统计类型功能的 QUERY

18、禁止使用跨库查询

解读:

(1) 耦合度太高,如果两张数据表处于不同的数据库中，那么它们在开始的时候就在不同的子系统中，也就是说在设计过程中没有很强的关联关系，在一句 SQL 中进行关联，不符合分而治之的思想。

(2) 效率问题,如果两个数据库在同一个实例中，查询效率还是能接受的。但是，随着时间推移，子系统越来越庞大，需要将数据库迁移出实例，那么，再次查询的时候就会有很大的效率问题。

(3) 物理与逻辑地分库会变得非常困难

19、禁止使用子查询，建议将子查询转换成关联查询

20、禁止核心业务流程 SQL 包含计算操作、多表关联、表遍历 case when 等复杂查询，建议拆分成单表简单查询

21、禁止使用非同类型的列进行等值查询

22、尽力做到以一条 SQL 语句的思想实现目的

23、禁止在 on 关键字后用 or 关联查询

解读:

(1) 严重影响性能

(2) 改为 union all 合并集

4. 分表

1、每张表数据量建议控制在 5000w 以下

2、可以结合使用 hash、range、lookup table 进行分表

3、分表时如果使用 md5（或者类似的 hash 算法）进行分表，表名后缀使用 16 进制，比如 user_ff

4、推荐使用 CRC32 求余（或者类似的算术算法）进行分表，表名后缀使用数字，数字必须从 0 开始并等宽，比如分 100 张表，后缀从 00-99

5、使用时间分表，表名后缀必须使用特定格式，比如按日期分表 user_20110209、按月分表 user_201102

5. 其他

1、批量导入、导出数据需要 DBA 进行审查，并在执行过程中观察服务

2、批量更新数据，如 update、delete 操作，需要 DBA 进行审查，并在执行过程中观察服务

3、产品出现非数据库平台运维导致的问题和故障时，如前端被抓站，请及时通知 DBA，便于维护服务稳定

4、业务部门程序出现 bug 等影响数据库服务的问题,请及时通知 DBA，便于维护服务稳定

5、业务部门推广活动，请提前通知 DBA 进行服务和访问评估

6、如果出现业务部门人为误操作导致数据丢失，需要恢复数据，请在第一时间通知 DBA，并提供准确时间，误操作语句等重要线索

7、优先选择以最新 GA 版本的 MySQL 次一个版本投入产品线上

解读: 现在 MySQL8.0 已 GA 了，考虑到稳定性与可靠性，可以用 www.mysql.com 上的 mysql5.7.21 版本投入生产。MySQL5.7 的性能是

MySQL5.6 的两倍，是 MySQL5.5 的五倍；建议尽快升级至 5.7.21 版本

8、应用服务 web service 在连接数据库时必须加入重连机制

9、考虑到 HA 及可扩展性，MySQL 的 UNIX 通信层的 socket 套接字文件 mysql.sock 必须位于/tmp 目录下

10、考虑到数据库系统的稳定性与可靠性,MySQL 必须部署在 Linux 操作系统上，必须使用 64 位

11、主从复制中的从库（只读型）实例必须关闭 binlog

解读：如果开启 binlog，IO 性能减半，严重影响性能；要开启 binlog，可以增加一倍 IO 性能为前提条件

12、数据库服务器操作系统 root 或 administrator 和 MySQL root 必须归属 DBA，从安全性与稳定性来说，知道的人越少越好

13、生产环境中应用帐号连接数据库的权限只需赋予 select、update、delete、insert、execute 权限

14、在 Linux 操作系统中且版本 MySQL5.7，在配置复制时必须标准化以下参数：

slave_parallel_type	= LOGICAL_CLOCK
slave_parallel_workers	= 16
master_info_repository	= TABLE
relay_log_info_repository	= TABLE
relay_log_recovery	= 1
slave_compressed_protocol	= 1
binlog_cache_size	= 1M
max_allowed_packet	= 1024M

15、在 Linux 操作系统且版本 MySQL5.7 的实例中必须启动 truncate undo log 功能，开启参数如下：

innodb_undo_directory	= /mysql/log
innodb_undo_logs	= 128
innodb_undo_tablespaces	= 10
innodb_undo_log_truncate	= 1
innodb_max_undo_log_size	= 1024M
innodb_purge_rseg_truncate_frequency	= 128

解读: 如果不开启 undo 日志切断功能，共享表空间文件 ibdata1 会变得无穷大且不能压缩、不能删除、不能释放，严重影响数据库性能和浪

费存储空间

16、在 Linux 操作系统且版本 MySQL5.7 的实例中，出于性能上的考虑必须标准化以下参数：

innodb_flush_method	= O_DIRECT
innodb_support_xa	= 1
innodb_max_dirty_pages_pct	= 75
innodb_autoinc_lock_mode	= 2
innodb_log_buffer_size	= 64M
innodb_file_per_table	= 1
innodb_log_file_size	= 2048M
innodb_log_files_in_group	= 2
innodb_autoextend_increment	= 1000
innodb_flush_log_at_trx_commit	= 0

innodb_online_alter_log_max_size = 1024M

innodb_change_buffer_max_size = 50

event_scheduler = 1

sync_binlog = 0

thread_stack = 1024000

skip-external-locking

skip-name-resolve

解读：如不标准以上参数，会严重影响数据库性能

17、在物理服务器中，必须进入 BIOS 关闭 NUMA

6. 存储与网络

1、必须使用本地存储

解读:

(1) MySQL 的体系结构是基于 share-nothing 的架构，不适合使用共享存储

(2) 及其容易导致 CPU System Time 升高，造成 CPU 负载过高，严重影响性能

(3) 导致 IO wait 过高，严重影响性能

(4) 读写速度共享存储没有本地存储快

(5) 上述问题是在实际公司中发生过的案例总结

2、必须使用 SSD

解读:

(1) SSD 读写速度比传统 SAS 机械硬盘超快是业界公论的

(2) 操作系统也必须安装在基于 RAID 的 SSD 存储上

3、必须做 RAID 10

4、LINUX 操作系统中的 IO 调度算法必须是基于 deadline 或 noop

解读: 在基于 SSD 的存储上，deadline 或 noop IO 调度算法性能相比默认值 cfq 能提升 25%存储性能，此值是官方公布的准确数据。实际测试中高于 25%

5、所有 MySQL 实例的 IP 地址（包含 VIP）必须位于同一网段的 LAN 内，必须广播而非路由

6、考虑到整个生态系统的负荷和数据库系统的网络流量与网速瓶颈（尤其数据库复制），建议整个物理机房 LAN 采用光钎网络交换

7、业界标准对于内存配置 128G，CPU 32 核，建议生产环境中内存最低配 128G，CPU32 核