MySQL数据库设计规范

1. 规范背景与目的

MySQL数据库与 Oracle、 SQL Server 等数据库相比，有其内核上的优势与劣势。我们在使用MySQL数据库的时候需要遵循一定规范，扬长避短。本规范旨在帮助或指导RD、QA、OP等技术人员做出适合线上业务的数据库设计。在数据库变更和处理流程、数据库表设计、SQL编写等方面予以规范，从而为公司业务系统稳定、健康地运行提供保障。

2. 设计规范

2.1 数据库设计

以下所有规范会按照【高危】、【强制】、【建议】三个级别进行标注，遵守优先级从高到低。

对于不满足【高危】和【强制】两个级别的设计，DBA会强制打回要求修改。

2.1.1 库名

1. 【强制】库的名称必须控制在32个字符以内，相关模块的表名与表名之间尽量体现join的关系，如user表和user_login表。
2. 【强制】库的名称格式：业务系统名称_子系统名，同一模块使用的表名尽量使用统一前缀。
3. 【强制】创建数据库时必须显式指定字符集，并且字符集只能是utf8或者utf8mb4。创建数据库SQL举例：`create database cncounter_uc default character set utf8;`。

2.1.2 表结构

1. 【强制】表和列的名称必须控制在32个字符以内，表名只能使用字母、数字和下划线且不能以数字开头，字母一律小写。
2. 【强制】创建表时必须显式指定字符集为utf8或utf8mb4。
3. 【强制】创建表时必须显式指定表存储引擎类型，如无特殊需求，一律为InnoDB。当需要使用除InnoDB/MyISAM/Memory以外的存储引擎时，必须通过DBA审核才能在生产环境中使用。因为Innodb 表支持事务、行锁、宕机恢复、MVCC等关系型数据库重要特性，为业界使用最多的MySQL存储引擎。而这是其他大多数存储引擎不具备的，因此首推InnoDB。
4. 【强制】建表必须有comment
5. 【建议】建表时关于主键：(1)强制要求必须有主键，推荐字段名为`f_id`，类型为bigint，无符号 unsigned，且为`auto_increment(2)` 。 标识表里每一行主体的字段不要设为主键，建议设为其他字段如`user_id`，`order_id`等，并建立unique key索引（可参考`cdb .teacher`表设计）。因为如果设为主键且主键值为随机插入，则会导致innodb内部page分裂和大量随机I/O，性能下降。
6. 【建议】核心表（如用户表，金钱相关的表）必须有行数据的创建时间字段`f_created_at`和最后更新时间字段`f_updated_at`，便于查问题。
7. 【建议】表中所有字段必须都是`NOT NULL`属性，业务可以根据需要定义`DEFAULT`值。因为使用NULL值会存在每一行都会占用额外存储空间、数据迁移容易出错、聚合函数计算结果偏差等问题。 但是，`CLOB/BLOB/TEXT` 类型不要设置 `NOT NULL` 和 `DEFAULT`
8. 【建议】建议对表里的`blob`、`text`等大字段，垂直拆分到其他表里，仅在需要读这些对象的时候才去select。
9. 【建议】反范式设计：把经常需要join查询的字段，在其他表里冗余一份。如`user_name`属性在`user_account`，`user_login_log`等表里冗余一份，减少join查询。
10. 【强制】对于超过1000W行的大表，DBA 不再进行`alter table`。

2.1.3 列数据类型优化

1. 【建议】表中的自增列（`auto_increment`属性），推荐使用`bigint`类型。因为无符号`int`存储范围为`0-4294967295`（大约42亿左右），溢出后会导致报错。

2. 【建议】业务中选择性很少的状态`status`、类型`type`等字段推荐使用`tinytint`或者`smallint`类型节省存储空间。
3. 【建议】业务中IP地址字段推荐使用`int`类型，不推荐用`char(15)`。因为`int`只占4字节，可以用如下函数相互转换，而`char(15)`占用至少15字节。一旦表数据行数到了1亿，那么要多用1.1G存储空间。 SQL：`select inet_aton('192.168.2.12'); select inet_ntoa(3232236044);` PHP: `ip2long('192.168.2.12'); long2ip(3530427185);`
4. 【建议】不推荐使用`enum`，`set`。 因为它们浪费空间，且枚举值写死了，变更不方便。推荐使用`tinyint`或`smallint`。
5. 【建议】不推荐使用`blob`，`text`等类型。它们都比较浪费硬盘和内存空间。在加载表数据时，会读取大字段到内存里从而浪费内存空间，影响系统性能。建议和PM、RD沟通，是否真的需要这么大字段。Innodb中当一行记录超过8098字节时，会将该记录中选取最长的一个字段将其768字节放在原始page里，该字段余下内容放在`overflow-page`里。不幸的是在`compact`行格式下，原始`page`和`overflow-page`都会加载。
6. 【建议】存储金钱的字段，建议用`DECIMAL`。
7. 【建议】文本数据尽量用`varchar`存储。因为`varchar`是变长存储，比`char`更省空间。MySQL server层规定一行所有文本最多存65535字符，超过会自动转换为`mediumtext`字段。而`text`在utf8字符集下最多存21844个字符，`mediumtext`最多存2 ^24/3个字符，`longtext`最多存2^32个字符。一般建议用`varchar`类型，字符数不要超过2700。
8. 【建议】时间类型尽量选取`timestamp`。因为`datetime`占用8字节，`timestamp`仅占用4字节，但是范围为`1970-01-01 00:00:01`到`2038-01-01 00:00:00`。更为高阶的方法，选用`int`来存储时间，使用SQL函数`unix_timestamp()`和`from_unixtime()`来进行转换。

详细存储大小参加下图：

类型（同义词）	存储长度	最小值（无符号）	最大值（无符号）
整型数字			
TINYINT	1	—128（0）	127（255）
SMALLINT	2	-32768（0）	32767（65535）
MEDIUMINT	3	-8388608（0）	8388607（16777215）
INT（INTEGER）	4	-2147483648（0）	2147483647（4294967295）
BIGINT	8	-9223372036854775808（0）	9223372036854775807（18446744073709551615）
小数支持			
FLOAT[(M[,D])]	4 or 8	-3.402823466E+38-1.175494351E-38 0 1.175494351E-38～3.402823466E+38	
DOUBLE[(M[,D])]（REAL, DOUBLE PRECISION）	8	-1.7976931348623157E+308～-2.2250738585072014E-308; 0 2.2250738585072014E-308～ 1.7976931348623157E+308	
时间类型			
DATETIME	8	1001-01-01 00:00:00	9999-12-31 23:59:59
DATE	3	1001-01-01	9999-12-31
TIME	3	00:00:00	23:59:59
YEAR	1	1001	9999
TIMESTAMP	4	1970-01-01 00:00:00	

2.1.4 索引设计

1. 【强制】InnoDB表必须主键为`f_id bigint unsigned not null auto_increment`,且主键值禁止被更新。
2. 【建议】主键的名称以“`pk_`”开头，唯一键以“`uniq_`”开头，普通索引以“`idx_`”开头，一律使用小写格式，以字段的名称或缩写作为后缀。
3. 【强制】InnoDB和MyISAM存储引擎表，索引类型必须为`BTREE`；MEMORY表可以根据需要选择`HASH`或者`BTREE`类型索引。
4. 【强制】单个索引中每个索引记录的长度不能超过64KB。
5. 【建议】单个表上的索引个数不能超过5个。
6. 【建议】在建立索引时，多考虑建立联合索引，并把区分度最高的字段放在最前面。如列`userid`的区分度可由`select count(distinct userid)`计算出来。
7. 【建议】在多表join的SQL里，保证被驱动表的连接列上有索引，这样join执行效率最高。
8. 【建议】建表或加索引时，保证表里互相不存在冗余索引。对于MySQL来说，如果表里已经存在`key(a,b)`，则`key(a)`为冗余索引，需要删除。

2.1.5 分库分表、分区表

1. 【强制】禁止使用分区表。
2. 【强制】有分表需求的，由RD和DBA商定创建、归档策略。

2.1.6 字符集

1. 【强制】数据库本身库、表、列所有字符集必须保持一致，为`utf8`或`utf8mb4`。
2. 【强制】前端程序字符集或者环境变量中的字符集，与数据库、表的字符集必须一致，统一为`utf8`。

2.1.7 程序层DAO设计建议

1. 【建议】代码层尽量不要用model，推荐使用手动拼SQL+绑定变量传入参数的方式。因为model虽然可以使用面向对象的方式操作db，但是其使用不当很容易造成生成的SQL非常复杂，且model 层自己做的强制类型转换性能较差，最终导致数据库性能下降。
2. 【建议】前端程序连接MySQL或者redis，必须要有连接超时和失败重连机制，且失败重试必须有间隔时间。
3. 【建议】前端程序报错里尽量能够提示MySQL或redis原生态的报错信息，便于排查错误。
4. 【建议】对于有连接池的前端程序，必须根据业务需要配置初始、最小、最大连接数，超时时间以及连接回收机制，否则会耗尽数据库连接资源，造成线上事故。
5. 【建议】对于log或history类型的表，随时间增长容易越来越大，因此上线前RD或者DBA必须建立表数据清理或归档方案。
6. 【建议】在应用程序设计阶段，RD必须考虑并规避数据库中主从延迟对于业务的影响。尽量避免从库短时延迟（20秒以内）对业务造成影响，建议强制一致性的读开启事务走主库，或更新后过一段时间再去读从库。
7. 【建议】多个并发业务逻辑访问同一块数据（innodb表）时，会在数据库端产生行锁甚至表锁导致并发下降，因此建议更新类SQL尽量基于主键去更新。
8. 【建议】业务逻辑之间加锁顺序尽量保持一致，否则会导致死锁。
9. 【建议】对于单表读写比大于10:1的数据行或单个列，可以将热点数据放在缓存里（如mecache或redis），加快访问速度，降低MySQL压力。

2.1.8 一个规范的建表语句示例

一个较为规范的建表语句为：

```
 CREATE TABLE `t_address` (
`f_id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '主键id',
`f_street` varchar(100) NOT NULL DEFAULT '' COMMENT '街道',
`f_town` varchar(30) NOT NULL DEFAULT '' COMMENT '镇',
`f_county` varchar(30) NOT NULL DEFAULT '' COMMENT '县',
`f_state` varchar(50) NOT NULL DEFAULT '' COMMENT '州',
`f_created_at` bigint(20) unsigned NOT NULL DEFAULT '0' COMMENT '创建时间',
`f_updated_at` bigint(20) unsigned NOT NULL DEFAULT '0' COMMENT '更新时间',
PRIMARY KEY (`f_id`),
UNIQUE KEY `uniq_state_county` (`f_state`,`f_county`),
KEY `idx_town` (`f_town`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8  COMMENT='地址表';
```

2.2 SQL编写

2.2.1 DML语句

1. 【强制】SELECT语句必须指定具体字段名称，禁止写成*。因为`select *`会将不该读的数据也从MySQL里读出来，造成网卡压力。且表字段一旦更新，但model层没有来得及更新的话，系统会报错。
2. 【强制】insert语句指定具体字段名称，不要写成`insert into t1 values(…)`，道理同上。
3. 【建议】`insert into…values(XX),(XX),(XX)…`。这里XX的值不要超过1000个。值过多虽然上线很很快，但会引起主从同步延迟。
4. 【建议】SELECT语句不要使用`UNION`，`UNION ALL`，节省数据库资源，提高性能。
5. 【建议】in值列表限制在100以内。例如`select… where userid in(….100个以内…)`，这么做是为了减少底层扫描，减轻数据库压力从而加速查询。
6. 【建议】事务里批量更新数据需要控制数量，进行必要的sleep，做到少量多次。
7. 【强制】事务涉及的表必须全部是innodb表。否则一旦失败不会全部回滚，且易造成主从库同步终端。
8. 【强制】有从库的话，建议只读SQL发往从库。
9. 【强制】除静态表或小表（100行以内），DML语句必须有where条件，且使用索引查找。
10. 【强制】生产环境禁止使用`hint`，如`sql_no_cache`，`force index`，`ignore key`，`straight join`等。因为`hint`是用来强制SQL按照某个执行计划来执行，但随着数据量变化我们无法保证自己当初的预判是正确的，因此我们要相信MySQL优化器！
11. 【强制】where条件里等号左右字段类型必须一致，否则无法利用索引。
12. 【建议】`SELECT|UPDATE|DELETE`要有WHERE子句，且WHERE子句的条件必需使用索引查找。
13. 【强制】生产数据库中强烈不推荐大表上发生全表扫描，但对于100行以下的静态表可以全表扫描。查询数据量不要超过表行数的25%，否则不会利用索引。
14. 【强制】WHERE 子句中禁止只使用全模糊的LIKE条件进行查找，必须有其他等值或范围查询条件，否则无法利用索引。
15. 【建议】索引列不要使用函数或表达式，否则无法利用索引。如`where length(name)='Admin'`或`where user_id+2=10023`。
16. 【建议】减少使用or语句，可将or语句优化为union，然后在各个where条件上建立索引。如`where a=1 or b=2`优化为`where a=1… union …where b=2, key(a),key(b)`。
17. 【建议】分页查询，当limit起点较高时，可先用过滤条件进行过滤。如`select a,b,c from t1 limit 10000,20;`优化为:`select a,b,c from t1 where id>10000 limit 20;`。

2.2.2 多表连接

1. 【强制】禁止多表连接查询。

2.2.3 事务

1. 【建议】事务中INSERT|UPDATE|DELETE语句操作的行数控制在1000以内，以及WHERE子句中IN列表的传参个数控制在100以内。
2. 【建议】批量操作数据时，需要控制事务处理间隔时间，进行必要的sleep，一般建议值5-10秒。
3. 【建议】对于有auto_increment属性字段的表的插入操作，并发需要控制在200以内。
4. 【强制】程序设计必须考虑“数据库事务隔离级别”带来的影响，包括脏读、不可重复读和幻读。线上事务隔离级别默认为READ-COMMITTED。
5. 【建议】事务里包含SQL不超过5个（支付业务除外）。因为过长的事务会导致锁数据较久，MySQL内部缓存、连接消耗过多等雪崩问题。
6. 【建议】事务里更新语句尽量基于主键或unique key，如update … where id=XX; 否则会产生间隙锁，内部扩大锁定范围，导致系统性能下降，产生死锁。
7. 【建议】尽量把一些典型外部调用移出事务，如调用webservice，访问文件存储等，从而避免事务过长。
8. 【建议】对于MySQL主从延迟严格敏感的select语句，请访问主库。

2.2.4 排序和分组

1. 【建议】减少使用order by，将排序放到程序端去做。order by、group by、distinct这些语句较为耗费CPU，数据库的CPU资源是极其宝贵的。
2. 【建议】order by、group by、distinct这些SQL尽量利用索引直接检索出排序好的数据。如where a=1 order by可以利用key(a,b)。
3. 【建议】包含了order by、group by、distinct这些查询的语句，where条件过滤出来的结果集请保持在1000行以内，否则SQL会很慢。

2.2.5 线上禁止使用的SQL语句

1. 【高危】禁用update|delete t1 … where a=XX limit XX; 这种带limit的更新语句。因为会导致主从不一致，导致数据错乱。建议加上order by PK。
2. 【高危】禁止使用关联子查询，如update t1 set … where name in(select name from user where…);效率极其低下。
3. 【强制】禁用procedure、function、trigger、views、event、外键约束。因为他们消耗数据库资源，降低数据库实例可扩展性。推荐都在程序端实现。
4. 【强制】禁用insert into …on duplicate key update…在高并发环境下，会造成主从不一致。
5. 【强制】禁止联表更新语句，如update t1,t2 where t1.id=t2.id…。

3. 熟悉数据库范式

1. 第一范式(1NF)：字段值具有原子性,不能再分(所有关系型数据库系统都满足第一范式);
   例如：姓名字段,其中姓和名是一个整体,如果区分姓和名那么必须设立两个独立字段;
2. 第二范式(2NF)：在满足第一范式下，一个表必须有主键,即每行数据都能被唯一的区分;
3. 第三范式(3NF)：在满足第二范式下，一个表中不包含其他相关表中非关键字段的信息,即数据表不能有冗余字段;

4. SQL审核规则

1. 【强制】表名符合命名规范且不能是MySQL关键字或保留字，建议添加前缀't_'来规避。
2. 【强制】列名符合命名规范且不能是MySQL关键字或保留字，建议添加前缀'f_'来规避 。
3. 【强制】设置无业务意义的无符号整型为自增主键。
4. 【强制】所有列都添加注释。
5. 【强制】所有的普通索引以 'idx_' 为前缀， 'idx_' 后面跟字段名称或字段简称，做到观其名知其意。
6. 【强制】所有的唯一索引以 'uniq_' 为前缀， 'uniq_' 后面跟字段名称或字段简称，做到观其名知其意。
7. 【强制】表添加注释，并且添加上 'ENGINE=InnoDB DEFAULT CHARSET=utf8mb4/utf8' 属性。
8. 【强制】编码类型字段需要给出编码对应关系。
9. 【强制】所有 varchar/char 字段指定为 not null 并带有默认值default， text/blob/clob 等大字段不允许有default属性,尽量避免使用text/blob/clob 。
10. 【强制】表名、字段名之间需要下划线 '_' 连接。 表名、字段名都小写 。
11. 【建议】bool值类型使用tinyint(1) 替代 。
12. 【建议】目前的默认的“自动审核”不支持 JSON 类型。
13. 【建议】如果需要使用 JSON 类型， 请选择“自动审核(支持JSON)” 或者 “人工审核”。
14. 【建议】如果默认的“自动审核”报索引长度不能超过766字节错误，请选择“自动审核(支持JSON)” 或者 “人工审核”。