MySQL 数据库开发常见问题及优化

mysql 数据库是被广泛应用的关系型数据库，其体积小、支持多处理器、开源并免费的特性使其在 Internet 中小型网站中的使用率尤其高。在使用 mysql 的过程中不规范的 SQL 编写、非最优的策略选择都可能导致系统性能甚至功能上的缺陷。

一、库表设计

1.1 引擎选择

在 mysql 5.1 中，引入了新的插件式存储引擎体系结构，允许将存储引擎加载到正在运新的 mysql 服务器中。使用 mysql 插件式存储引擎体系结构，允许数据库专业人员或者设计库表的软件开发人员为特定的应用需求选择专门的存储引擎，完全不需要管理任何特殊的应用编码要求，也无需考虑所有的底层实施细节。因此，尽管不同的存储引擎具有不同的能力，应用程序是与之分离的。此外，使用者可以在服务器、数据库和表格三个层级中存储引擎，提供了极大的灵活性。

mysql 常用的存储引擎包括 **MYISAM**、**Innodb** 和 **Memory**，其中各自的特点如下：

- **MYISAM**：全表锁，拥有较高的执行速度，一个写请求请阻塞另外相同表格的所有

读写请求，并发性能差，占用空间相对较小，mysql 5.5 及以下仅 MYISAM 支持全文索引，不支持事务。

- **Innodb**：行级锁（SQL 都走索引查询），并发能力相对强，占用空间是 MYISAM 的 2.5 倍，不支持全文索引（5.6 开始支持），支持事务
- **Memory**：全表锁，存储在内存当中，速度快，但会占用和数据量成正比的内存空间且数据在 mysql 重启时会丢失。

基于以上特性 ,建议绝大部份都设置为 innodb 引擎 ,特殊的业务再考虑选用 MYISAM 或 Memory ，如全文索引支持或极高的执行效率等。

1.2 分表方法

在数据库表使用过程中，为了减小数据库服务器的负担、缩短查询时间，常常会考虑做分表设计。分表分两种，一种是纵向分表（将本来可以在同一个表的内容，人为划分存储在为多个不同结构的表）和横向分表（把大的表结构，横向切割为同样结构的不同表）。

其中，纵向分表常见的方式有根据活跃度分表、根据重要性分表等。其主要解决问题如下：

1. 表与表之间资源争用问题；
2. 锁争用机率小；
3. 实现核心与非核心的分级存储，如 UDB 登陆库拆分成一级二级三级库
4. 解决了数据库同步压力问题。

横向分表是指根据某些特定的规则来划分大数据量表，如根据时间分表。其主要解决问题如下：

1. 单表过大造成的性能问题；
2. 单表过大造成的单服务器空间问题。

1.3 索引问题

索引是对数据库表中一个或多个列的值进行排序的结构，建立索引有助于更快地获取信息。mysql 有四种不同的索引类型：

- 主键索引（PRIMARY）
- 唯一索引（UNIQUE）
- 普通索引（INDEX）
- 全文索引（FULLTEXT，MYISAM 及 mysql5.6 以上的 innodb）

建立索引的目的是加快对表中记录的查找或排序，索引也并非越多越好，因为创建索引是要付出代价的：

- 一是增加了数据库的存储空间；
- 二是在插入和修改数据时要花费较多的时间维护索引。

在设计表或索引时，常出现以下几个问题：

1. 少建索引或不建索引。这个问题最突出，建议建表时 DBA 可以一起协助把关。

2. 索引滥用。滥用索引将导致写请求变慢，拖慢整体数据库的响应速度（5.5 以下的 mysql 只能用到一个索引)。
3. 从不考虑联合索引。实际上联合索引的效率往往要比单列索引的效率更高。
4. 非最优列选择。低选择性的字段不适合建单列索引，如 status 类型的字段。

二、慢 SQL 问题

2.1 导致慢 SQL 的原因

在遇到慢 SQL 情况时，不能简单的把原因归结为 SQL 编写问题(虽然这是最常见的因素)，实际上导致慢 SQL 有很多因素，甚至包括硬件和 mysql 本身的 bug。根据出现的概率从大到小，罗列如下：

1. SQL 编写问题
2. 锁
3. 业务室里相互干扰对 IO/CPU 资源争用
4. 服务器硬件
5. MySQL BUG

2.2 由 SQL 编写导致的慢 SQL 优化

针对 SQL 编写导致的慢 SQL，优化起来还是相对比较方便的。正如上一节提到的正确的使用索引能加快查询速度，那么我们在编写 SQL 时就需要注意与索引相关的规则：

- 字段类型转换导致不用索引，如字符串类型的不用引号，数字类型的用引号等，这有可能会用不到索引导致全表扫描；
- mysql 不支持函数转换，所以字段前面不能加函数，否则这将用不到索引；
- 不要在字段前面加减运算；
- 字符串比较长的可以考虑索引一部份减少索引文件大小，提高写入效率；
- like % 在前面用不到索引；
- 根据联合索引的第二个及以后的字段单独查询用不到索引；
- 不要使用 select *；
- 排序请尽量使用升序；
- or 的查询尽量用 union 代替 （Innodb)；
- 复合索引高选择性的字段排在前面；
- order by / group by 字段包括在索引当中减少排序，效率会更高。

除了上述索引使用规则外，SQL 编写时还需要特别注意一下几点：

- 尽量规避大事务的 SQL，大事务的 SQL 会影响数据库的并发性能及主从同步；
- 分页语句 limit 的问题；
- 删除表所有记录请用 truncate，不要用 delete；
- 不让 mysql 干多余的事情，如计算；

- 输写 SQL 带字段，以防止后面表变更带来的问题，性能也是比较优的（ 涉及到数据字典解析，请自行查询资料)；
- 在 Innodb 上用 select count(*)，因为 Innodb 会存储统计信息；
- 慎用 Oder by rand()。

三、分析诊断工具

在日常开发工作中，我们可以做一些工作达到预防慢 SQL 问题，比如在上线前预先用诊断工具对 SQL 进行分析。常用的工具有：

- mysqldumpslow
- mysql profile
- mysql explain

四、误操作、程序 bug 时怎么办

实际上误操作和程序 bug 导致数据误删或者混乱的问题并非少见，但是刚入行的开发工作者会比较紧张。一个成熟的企业往往会有完善的数据管理规范和较丰富的数据恢复方案（初创公司除外），会进行数据备份和数据容灾。当你发现误操作或程序 bug 导致线上数据被误删或误改动时，一定不能慌乱，应及时与 DBA 联系，第一时间进行数据恢复（严重时直接停止服务），尽可能减少影响和损失。对于重要数据（如资金）的操作，在开发时一定要反复进行测试，确保没有问题后再上线。