# SQL优化（系列）-SELECT语句优化

**一、常见的 SELECT 查询语句优化技巧**

     SQL 中的 SELECT 查询语句，优化是非常重要的，可以显著提高查询性能和效率。以下是一些常见的 SELECT 查询语句优化技巧：

**1.1、选择合适的列：**

只选择需要的列，避免使用 SELECT \* 来获取所有列的数据。

避免查询大字段（如 TEXT、BLOB 类型），尽量只查询需要的字段。

**1.2、使用索引：**

    确保涉及的列上有适当的索引，可以加快查询速度。

    尽可能避免全表扫描，使用索引来定位数据。

    避免在索引列上使用函数或运算符，会导致索引失效。

**1.3、使用 JOIN 优化：**

    尽量避免使用多表连接，可以使用子查询或临时表等方式替代。

    考虑使用 INNER JOIN、LEFT JOIN 等不同类型的 JOIN，选择适合的 JOIN 类型。

    对 JOIN 的列建立索引，提高 JOIN 操作的效率。

**1.4、使用 WHERE 子句：**

    在查询中使用 WHERE 子句限制返回的行数，减少不必要的数据传输。

    使用合适的条件查询，避免过多的 AND、OR 操作符。

**1.5、分页查询：**

    对于大数据集查询，使用 LIMIT 和 OFFSET 进行分页，避免一次性返回大量数据。

**1.6、统计信息：**

    使用 COUNT()、SUM()、AVG() 等统计函数时，尽可能减少对大数据集的操作。

**1.7、避免子查询：**

    尽量避免嵌套子查询，可以尝试使用 JOIN 或临时表优化子查询。

**1.8、缓存数据：**

    对于不经常变化的数据，可以考虑使用缓存来提高查询速度。

**1.9、定期优化表：**

    定期分析表结构，优化索引，清理无用数据，可以提高查询效率。

**二、优化实践**

**2.1、避免出现select \***

select \* 操作在任何类型数据库中都不是一个好的SQL编写习惯。

    使用select \* 取出全部列，会让优化器无法完成索引覆盖扫描这类优化，会影响优化器对执行计划的选择，也会增加网络带宽消耗，更会带来额外的I/O,内存和CPU消耗。***建议提出业务实际需要的列数，将指定列名以取代select \*。***

**2.2、避免出现不确定结果的函数**

    特定针对主从复制这类业务场景。由于原理上从库复制的是主库执行的语句，使用如now()、rand()、sysdate()、current\_user()等不确定结果的函数很容易导致主库与从库相应的数据不一致。另外不确定值的函数,产生的SQL语句无法利用query cache。

**2.3、多表关联查询时，小表在前，大表在后。**

    在MySQL中，执行 from 后的表关联查询是从左往右执行的（Oracle相反），第一张表会涉及到全表扫描，所以将小表放在前面，先扫小表，扫描快效率较高，在扫描后面的大表，或许只扫描大表的前100行就符合返回条件并return了。

**例如：**

    表1有50条数据，表2有30亿条数据；如果全表扫描表2，你品，那就先去吃个饭再说吧是吧。

**2.4、使用表的别名**

    当在SQL语句中连接多个表时，请使用表的别名并把别名前缀于每个列名上。这样就可以减少解析的时间并减少哪些友列名歧义引起的语法错误。

**2.5、用where字句替换HAVING字句**

    避免使用HAVING字句，因为HAVING只会在检索出所有记录之后才对结果集进行过滤，而where则是在聚合前刷选记录，如果能通过where字句限制记录的数目，那就能减少这方面的开销。HAVING中的条件一般用于聚合函数的过滤，除此之外，应该将条件写在where字句中。

***where和having的区别：where后面不能使用组函数***

**2.6、调整Where字句中的连接顺序**

    MySQL采用从左往右，自上而下的顺序解析where子句。根据这个原理，应将过滤数据多的条件往前放，最快速度缩小结果集。