MySQL基本用法总结

1. 存储过程和函数的基本用法
2. 存储过程的优缺点

一般SQL语句在执行的时候需要要先编译，然后执行，而存储过程和函数是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数来调用执行它。

一个存储过程是一个可编程的函数，它在数据库中创建并保存。它可以由SQL语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。数据库中的存储过程可以看做是对编程中面向对象方法的模拟。它允许控制数据的访问方式。

其大致有如下优点：

1. 可以使用流程控制语句完成复杂的判断和较复杂的运算，封装数据逻辑和业务规则
2. 重用性强，减少工作量，DBA可以灵活地升级、维护存储过程和函数，对应用程序源代码毫无影响
3. 存储过程和函数都是经过预编译的，执行速度快
4. 存储过程能减少网络流量和数据库连接，网络中传输的只是调用语句和参数
5. 可在一定程度上保证数据安全，通过存储过程的权限限制，能实现对数据的访问权限的限制，防止注入攻击

缺点：

1. 编写和调试麻烦
2. 性能优势不明显。大多数SQL编译的时间并不长，数据库本身是做数据存储的，不适合进行复杂的业务逻辑操作，承担业务压力会占用大量的系统资源
3. 可移植性差。每种数据库的内部编程语法都不太相同，一种数据库的存储过程迁移到另外一种数据库可能无法运行
4. 可维护性差，大量采用存储过程进行业务逻辑的开发时，由于不支持面向对象的设计，故无法采用面向对象的方式将业务逻辑进行封装
5. 基本语法和用法

存储过程：

CREATE PROCEDURE sp\_name([proc\_parameter])

[characteristic...] routine\_body

函数：

CREATE FUNCTION sp\_name([func\_parameter]) RETURN type

[characteristic...] routine\_body

注：

proc\_parameter：[IN|OUT|INOUT] param\_name type

func\_parameter：param\_name type

Type：任何有效的MySQL数据类型

Characteristic：指定存储过程的特性，有多个可选值。LANGUAGE SQL、[NOT] DETERMINISTIC、 { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }、 SQL SECURITY { DEFINER | INVOKER }、COMMENT 'string'

routine\_body：储过程体，BEGIN…END标志存储过程体的开始和结束

3、综上所述，通常在以下情况下，我们可以考虑使用存储过程来实现：

1. 当一个事务涉及到多个SQL语句时或者涉及到对多个表的操作时
2. 当在一个事务的完成需要很复杂的商业逻辑时，比如，对多个数据的操作，对多个状态的判断更改等
3. 比较复杂的统计和汇总

适当的使用存储过程，能够提高我们SQL查询的性能，以便于提高我们的工作效率。但是过多的使用存储过程也会降低系统的性能。故当系统没有性能问题时不建议用存储过程。

1. 触发器的基本用法
2. 触发器的优缺点

触发器是个特殊的存储过程，它的执行不是由程序调用，也不是手工启动，而是由事件来触发，触发器经常用于加强数据的完整性约束和业务规则等。 触发器可以从 TRIGGERS数据字典中查到。

优点：

1. 可以禁止或回滚违反引用完整性的更改，从而取消所尝试的数据修改
2. 自动更新和同步数据，减少代码量

缺点：

1. 触发器排错困难，后期维护不方便
2. 不能做DDL
3. 同存储过程一样，移植性比较差
4. 触发器的基本语法

CREATE TRIGGER trigger\_name trigger\_time trigger\_event

ON table\_name FOR EACH ROW trigger\_stmt

注：

trigger\_time：BEFORE或AFTER，触发器的执行时间

trigger\_event： 触发器的触发事件

INSERT：将新行插入表时激活触发程序，例如，通过INSERT、LOAD DATA和REPLACE

语句

UPDATE：更改某一行时激活触发程序，例如，通过UPDATE语句

DELETE：从表中删除某一行时激活触发程序，例如，通过DELETE和REPLACE语句

trigger\_stmt：触发程序激活时执行的语句，可使用BEGIN ... END复合语句结构

1. 触发器使用场景

一般而言，当我们增删改一条数据，并且需要对多表做数据同步的时候，就可以考虑使用触发器，其他情况下建议不要用触发器。

1. 索引的基本用法
2. 索引的优缺点

索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。数据库索引好比是一本书前面的目录，能加快数据库的查询速度。MySQL索引分为聚簇索引和非聚簇索引两种，聚簇索引是按照数据存放的物理位置为顺序的，而非聚簇索引是随机存放的；聚簇索引能提高多行检索的速度，而非聚簇索引对于单行的检索很快。每个表只能有一个聚簇索引，因为一个表中的记录只能以一种物理顺序存放，缺省情况下建立的索引是非聚簇索引。MySQL的存储引擎到目前为止都不支持用户主动创建聚簇索引，唯一的聚簇索引就是InnoDB存储引擎的主键。

MySQL中的索引有：主键索引(PRIMARY)、唯一索引(UNIQUE)、普通索引(NORMAL)、全文索引(FULLTEXT)、组合索引，空间索引，前缀索引等。

按索引结构的不同，又可以分为HASH索引、BTREE索引、RTREE索引、FULLTEXT索引。

按索引的顺序和数据的物理存储顺序的关系，又可以分为聚簇索引和非聚簇索引。聚簇索引的顺序就是数据的物理存储顺序，而对非聚簇索引的索引顺序与数据物理排列顺序无关。

根据HASH值建立索引，索引效率非常高，索引的检索可以一次定位；BTREE索引就是一种将索引值按一定的算法，存入一个树形的数据结构中；RTREE索引表示空间索引。

另外需要注意的是，在MySQL中，主键和外键约束也是通过索引来实现的，数据库会自动给主键和外键建立索引。

优点：提高了查询速度，大大减少了服务器需要扫描的数据量；可以帮助服务器避免排序或减少使用临时表排序；索引可以使随机I/O变为顺序I/O

缺点：降低更新表的速度；可能占用较多的磁盘空间；对于插入、删除、更新操作也会增加处理上的开销；增加优化器在选择索引时的计算代价；索引会产生相应的碎片，产生维护开销

1. 索引的语法

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index\_name

[index\_type]

ON tbl\_name (index\_col\_name,...)

index\_col\_name:

col\_name [(length)] [ASC | DESC]

index\_type:

USING {BTREE | HASH}

1. 索引的使用场景
2. 索引应该建在选择性高的字段上（键值唯一的记录数/总记录条数），唯一索引的选择性最高
3. 组合索引中选择性越高的字段排在最前面
4. 表的主键、外键必须有索引
5. 经常出现在Where子句和连接子句中的字段，特别是大表的字段，应该建立索引
6. 使用短索引，特别是对字符串索引，尽量指定前缀长度
7. 根据最左前缀的原则，尽量建立复合索引
8. 频繁进行数据操作的表，不要建立太多的索引
9. 删除无用的索引，避免对执行计划造成负面影响

哪些情况下数据库会使用索引：

1. 对一个键码使用>、 >=、 =、<、<=、IS NULL、IN、BETWEEN、LIKE(不以通配符开始)、GROUP BY、ORDER BY
2. 复合索引的最左前缀
3. 列前缀可以使用索引
4. where子句中，部分字段精确匹配，其他字段范围匹配

哪些情况下存在索引但是数据库无法使用：

1. 以%开头的LIKE查询不能利用到BTREE索引
2. 数据类型出现隐式转换不会使用索引
3. 复合索引下，查询条件不满足最左原则
4. 数据列中的数据区分度小，查询分析器分析使用索引比全表扫描更慢
5. OR条件中，OR前的条件中有索引，而后面的没有索引
6. 搜索一个索引而在另一个索引上做ORDER BY
7. 请求表上的数据行超出表总记录数30%，变成全表扫描
8. 谓词上的索引列上存在NULL值
9. 谓词上的索引列条件使用函数
10. 谓词上的索引列条件上使用了<>，NOT IN操作符
11. where子句中，范围匹配后面的其他字段无法使用索引
12. 内存表（HEAP表）使用HASH索引时，使用范围检索或者ORDER BY
13. 表关联字段类型不一样