Mysql

>>存储引擎

1. 存储引擎就是如何存储数据，如何为存储数据建立索引和如何更新删除查询数据的技术表现。
2. 存储引擎：MyISAM，InnoDB，Memory，Archive
   1. MyISAM存储引擎：高速存储和检索，以及全文搜索能力，不支持事务。
   2. InnoDB存储引擎：支持ACID兼容的事务功能，具有事务提交，回滚，和崩溃修复能力，提供行级锁，外键。缺点：读写效率较差，占用数据空间较大
   3. Memory存储引擎：将表中的数据存储在内存中，如果数据库重启或者发生崩溃表中的数据就会消失，不支持事务。速度非常快，基于hash索引

>>级联

1. 交叉查询：
   1. select \* from tb\_a,tb\_b;返回的结果是两个表中的数据条数的乘积
2. 内连接：
   1. 在交叉连接的基础上添加where子句，从而实现内连接
   2. 也可以使用inner join tb\_b on condition
3. 外连接：
   1. left join:查询出左边所有的数据，而只查出右边符合条件的数据
   2. right join:查询出右边所有的数据，而只查出左边符号条件的数据，如果右边有，左边无，则为null
4. 联合查询：union

**>>查询**

select [ all | distinct ] field from tb\_name [where condition] [group by field] [having condition] [order by field] [asc | desc] [limit [startIndex,]num]

1. Having子句只能与group by 子句一块使用
2. Having 和group by 的区别
   1. Having针对结果组，where针对每一列
   2. Having可以和聚合函数一块使用，而where 不能
   3. Having只过滤分组后的数据，where在分组前进行过滤
3. 交叉连接
   1. select \* from tb\_1,tb\_2
4. 内连接
   1. select \* from tb1 inner join tb2 on condition
   2. select \* from tb1,tb2 where condition
5. 外链接
   1. select \* from tb1 left join tb2 on condition;
   2. select \* from tb1 right join tb1 on condition;
6. 自连接
   1. select \* from tb1 inner join tb1 on condition where
7. 在连接操作中使用on和使用where的区别
   1. on是针对临时表而言，不论结果如何都会返回左边表的数据
   2. 而where是针对最终表而言，如果不符号条件，左右表中的数据均删去
8. 联合查询:必须保证字段数目相同，字段也应该相同，否则无意义。
   1. select \* from tb1 union select \* from tb2;
9. 子查询：
   1. select \* from tb1 where filed = (select field from tb2 where condition);
10. 正则表达式查询：
    1. field regexp ‘’

>>索引

**>>系统函数**

1. max() min()
2. avg()
3. sum()
4. count()
5. std() stddev()标准差
6. variance()方差
7. group\_concat():将一列值拼接在一起
8. abs()
9. mod(x,y)
10. sqrt() pow(x,y) PI() greatest() least() lg() log() exp()
11. floor()向下取整 ceiling()向上取整
12. rand()生成一个0-1之间的随机数 rand(3)：生成一个0-1之间固定的随机数
13. sin() tan() cos() cot() asin() atan() acos()
14. round()四舍五入，取离得最近的整数，round(num,num2)保留num2位小数
15. truncate(num,num2):将数截断为num2位小数
16. sign()判断一个数的正负：整数1，负数-1，零0
17. concat();字符串拼接 concat\_ws(separator,s1,s2..):加分隔符的字符串拼接
18. replace(str,from\_str,to\_str):替换字符str中所有f\_str为to\_str
19. insert(str,pos,len,newstr):用字符串替换原字符串中从pos开始长为len的字符串
20. substring（str,pos,len）:截取字符中从第pos个字符开始

长为len的字符串

1. reverse(str):字符串反转
2. repeat(str,count):重复生成指定的次数的字符串
3. strcmp(str1,str2):比较两个字符串的大小，返回值-1，1，0；
4. ucase(str) lcase(str)大小写转换

**>>事务：只有InnoDB支持**

1. ACID特性：
   1. 原子性：事务是一个不可分割的单位，要么都成功，要么都不成功
   2. 一致性：事务将数据库从一种状态变成另外一只状态，数据库的完整性约束没有破坏。
   3. 隔离性：当前事务提交前其他事务不可见。每个读写事务与其他事务的操作对象相互分离。
   4. 持久性：事务一旦提交成功，其结果就是永久的。
2. 事务的隔离等级
   1. READ UNCOMMITED:脏读
   2. READ COMMITED: 不可重复读，同一事务中可能会有多次提交数据，导致两次查询的结果不一致。
   3. REPEATABLE READ: Mysql默认的事务隔离级别
   4. SERIALIZABLE:强制事务排序，使其不能相互冲突，从而解决幻读，原理是在每个读的数据行上加上共享锁。会导致超时和锁竞争。
3. 数据库的锁
   1. 共享锁：s锁，即一个事务获取了一个数数据行的贡献锁，其他事务也能获得该行对应的共享锁，也就是读取数据行的锁
   2. 排它锁：x锁，一个事务获得了 排它锁，即写锁，可以写入和读取，但是其他事务不能写，只能读。

>>索引

1. InnoDB,MyISAM支持BTREE索引，MEMORY支持HASH索引和BTREE索引，默认为前者。
   1. 普通索引：

index(field)

create index indexName on tbName(field(length))

alter table tbName add index indexName (field(length))

drop index indexName on tbName

* 1. 唯一索引：

unix index indexName (field)

create unix index indexName on tbName(field(length))

* 1. 全文索引：

fulltext index indexName(field)

* 1. 单列索引：

* 1. 多列索引：

index indexName(field1,field2..)

create index indexName on tbName(field1,field2...)

* 1. 空间索引：

1. 设计原则：
   1. 选择唯一性索引
   2. 为经常需要排序，分组和联合操作的字段简历索引
   3. 为经常作为查询条件的字段简历索引
   4. 限制索引数目
   5. 尽量使用引用数据量少的索引
   6. 尽量使用前缀最为索引
   7. 删除不在使用或者很少使用的索引
2. 优缺点:
   1. 提高查询速度，节省分组排序时间
   2. 创建和维护索引是需要耗费时间的，占用物理空间，数据改动的时候需要动态维护索引