六、数据库MySql

1.数据库的范式

（1）范式

第一范式：列不可分，eg:【联系人】（姓名，性别，电话），一个联系人有家庭电话和公司电话，那么这种表结构设计就没有达到 1NF；

第二范式：有主键，保证完全依赖。eg:订单明细表【OrderDetail】（OrderID，ProductID，UnitPrice，Discount，Quantity，ProductName），Discount（折扣），Quantity（数量）完全依赖（取决）于主键（OderID，ProductID），而 UnitPrice，ProductName 只依赖于 ProductID，不符合2NF；

第三范式：无传递依赖(非主键列 A 依赖于非主键列 B，非主键列 B 依赖于主键的情况)，eg:订单表【Order】（OrderID，OrderDate，CustomerID，CustomerName，CustomerAddr，CustomerCity）主键是（OrderID），CustomerName，CustomerAddr，CustomerCity 直接依赖的是 CustomerID（非主键列），而不是直接依赖于主键，它是通过传递才依赖于主键，所以不符合 3NF。

（2）数据库设计范式主要用于解决一下问题：

1.强调关系数据库中的表是不可分的

2.减少数据库的数据冗余

3.关系数据库不会因为数据的更新而导致数据的不一致性。

2.索引

索引是对数据库表中一个或多个列的值进行排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B\_TREE及其变种。索引加速了数据访问，因为存储引擎不会再去扫描整张表得到需要的数据；相反，它从根节点开始，根节点保存了子节点的指针，存储引擎会根据指针快速寻找数据。

1）索引的底层实现

（1）B树和B+树(概念，插入删除)

（2）为什么说B+-tree比B 树更适合实际应用中操作系统的文件索引和数据库索引？

B+tree的磁盘读写代价更低：B+tree的内部结点并没有指向关键字具体信息的指针(红色部分)，因此其内部结点相对B 树更小。如果把所有同一内部结点的关键字存放在同一盘块中，那么盘块所能容纳的关键字数量也越多。一次性读入内存中的需要查找的关键字也就越多，相对来说IO读写次数也就降低了；

B+tree的查询效率更加稳定：由于内部结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引，所以，任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当；

数据库索引采用B+树而不是B树的主要原因：B+树只要遍历叶子节点就可以实现整棵树的遍历，而且在数据库中基于范围的查询是非常频繁的，而B树只能中序遍历所有节点，效率太低。

2） 索引的优点

大大加快数据的检索速度，这也是创建索引的最主要的原因；

加速表和表之间的连接；

在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间；

通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性；

3). 什么情况下设置了索引但无法使用？

以“%(表示任意0个或多个字符)”开头的LIKE语句，模糊匹配；

OR语句前后没有同时使用索引；

数据类型出现隐式转化（如varchar不加单引号的话可能会自动转换为int型）；

对于多列索引，必须满足 最左匹配原则 (eg：多列索引col1、col2和col3，则 索引生效的情形包括 col1或col1，col2或col1，col2，col3)。

4). 什么样的字段适合创建索引？

经常作查询选择的字段

经常作表连接的字段

经常出现在order by, group by, distinct 后面的字段

5) 创建索引时需要注意什么？

非空字段：应该指定列为NOT NULL，除非你想存储NULL。在mysql中，含有空值的列很难进行查询优化，因为它们使得索引、索引的统计信息以及比较运算更加复杂。你应该用0、一个特殊的值或者一个空串代替空值；

取值离散大的字段：（变量各个取值之间的差异程度）的列放到联合索引的前面，可以通过count()函数查看字段的差异值，返回值越大说明字段的唯一值越多字段的离散程度高；

索引字段越小越好：数据库的数据存储以页为单位一页存储的数据越多一次IO操作获取的数据越大效率越高。

6)索引的缺点

时间方面：创建索引和维护索引要耗费时间，具体地，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度；

空间方面：索引需要占物理空间。

7)索引的分类

普通索引和唯一性索引：索引列的值的唯一性

单个索引和复合索引：索引列所包含的列数

聚簇索引与非聚簇索引：聚簇索引按照数据的物理存储进行划分的。对于一堆记录来说，使用聚集索引就是对这堆记录进行堆划分，即主要描述的是物理上的存储。正是因为这种划分方法，导致聚簇索引必须是唯一的。聚集索引可以帮助把很大的范围，迅速减小范围。但是查找该记录，就要从这个小范围中Scan了；而非聚集索引是把一个很大的范围，转换成一个小的地图，然后你需要在这个小地图中找你要寻找的信息的位置，最后通过这个位置，再去找你所需要的记录

8). 主键、自增主键、主键索引与唯一索引概念区别

主键：指字段 唯一、不为空值 的列；

主键索引：指的就是主键，主键是索引的一种，是唯一索引的特殊类型。创建主键的时候，数据库默认会为主键创建一个唯一索引；

自增主键：字段类型为数字、自增、并且是主键；

唯一索引：索引列的值必须唯一，但允许有空值。主键是唯一索引，这样说没错；但反过来说，唯一索引也是主键就错误了，因为唯一索引允许空值，主键不允许有空值，所以不能说唯一索引也是主键。

9) 主键就是聚集索引吗？主键和索引有什么区别？

主键是一种特殊的唯一性索引，其可以是聚集索引，也可以是非聚集索引。在SQLServer中，主键的创建必须依赖于索引，默认创建的是聚集索引，但也可以显式指定为非聚集索引。InnoDB作为MySQL存储引擎时，默认按照主键进行聚集，如果没有定义主键，InnoDB会试着使用唯一的非空索引来代替。如果没有这种索引，InnoDB就会定义隐藏的主键然后在上面进行聚集。所以，对于聚集索引来说，你创建主键的时候，自动就创建了主键的聚集索引。

3.数据库事务

事务是一个不可分割的数据库操作序列，也是数据库并发控制的基本单位，其执行的结果必须使数据库从一种一致性状态变到另一种一致性状态。

(1). 事务的特征(ACID特性)

原子性(Atomicity)：事务所包含的一系列数据库操作要么全部成功执行，要么全部回滚；

一致性(Consistency)：事务的执行结果必须使数据库从一个一致性状态到另一个一致性状态；

隔离性(Isolation)：并发执行的事务之间不能相互影响；

持久性(Durability)：事务一旦提交，对数据库中数据的改变是永久性的。

(2). 事务并发带来的问题

脏读：一个事务读取了另一个事务未提交的数据；

不可重复读：不可重复读的重点是修改，同样条件下两次读取结果不同，也就是说，被读取的数据可以被其它事务修改；

幻读：幻读的重点在于新增或者删除，同样条件下两次读出来的记录数不一样。

(3). 隔离级别

隔离级别决定了一个session中的事务可能对另一个session中的事务的影响。ANSI标准定义了4个隔离级别，MySQL的InnoDB都支持，分别是：

READ UNCOMMITTED：最低级别的隔离，通常又称为dirty read，它允许一个事务读取另一个事务还没commit的数据，这样可能会提高性能，但是会导致脏读问题；

READ COMMITTED：在一个事务中只允许对其它事务已经commit的记录可见，该隔离级别不能避免不可重复读问题；

REPEATABLE READ：在一个事务开始后，其他事务对数据库的修改在本事务中不可见，直到本事务commit或rollback。但是，其他事务的insert/delete操作对该事务是可见的，也就是说，该隔离级别并不能避免幻读问题。在一个事务中重复select的结果一样，除非本事务中update数据库。

SERIALIZABLE：最高级别的隔离，只允许事务串行执行。

MySQL默认的隔离级别是REPEATABLE READ

(4)、mysql的事务支持

MySQL的事务支持不是绑定在MySQL服务器本身，而是与存储引擎相关：

MyISAM：不支持事务，用于只读程序提高性能；

InnoDB：支持ACID事务、行级锁、并发；

Berkeley DB：支持事务。

4.SQL优化

实践中，MySQL的优化主要涉及SQL语句及索引的优化、数据表结构的优化、系统配置的优化和硬件的优化四个方面。

1）SQL语句及索引的优化

2)、数据表结构的优化

数据库表结构的优化包括选择合适数据类型、表的范式的优化、表的垂直拆分和表的水平拆分等手段。

5.MySql的存储引擎的不同

1)InnoDB 和 MyISAM之间的区别：

(1).InnoDB支持事物，而MyISAM不支持事物

(2).InnoDB支持行级锁，而MyISAM支持表级锁

(3).InnoDB支持MVCC(多版本并发控制), 而MyISAM不支持

(4).InnoDB支持外键，而MyISAM不支持

(5).InnoDB不支持全文索引，而MyISAM支持。

MyISAM：如果数据表主要用来插入和查询记录，则MyISAM（但是不支持事务）引擎能提供较高的处理效率

Memory：如果只是临时存放数据，数据量不大，并且不需要较高的数据安全性，可以选择将数据保存在内存中的Memory引擎，MySQL中使用该引擎作为临时表，存放查询的中间结果。数据的处理速度很快但是安全性不高。

Archive：如果只有INSERT和SELECT操作，可以选择Archive，Archive支持高并发的插入操作，但是本身不是事务安全的。Archive非常适合存储归档数据，如记录日志信息可以使用Archive

InnoDB ：如果要提供提交、回滚、崩溃恢复能力的事务安全（ACID兼容）能力，并要求实现并发控制，InnoDB是一个好的选择

2)InnoDB引擎的四大特性

插入缓冲（insert buffer)

二次写(double write)

自适应哈希索引(ahi)

预读(read ahead)

6.Mysql怎么分表，以及分表后如果想按条件分页查询怎么办(如果不是按分表字段来查询的话，几乎效率低下，无解)

https://blog.csdn.net/ahjxhy2010/article/details/50540500

7.NOSQL数据库 —— Redis

Redis是一款基于内存的且支持持久化、高性能的Key-Value NoSQL 数据库，其支持丰富数据类型(string，list，set，sorted set，hash)，常被用作缓存的解决方案。Redis具有以下显著特点：

速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)；

支持丰富数据类型，支持string，list，set，sorted set，hash；

支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行；

丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除。

Redis作查询缓存需要注意考虑以下几个问题，包括防止脏读、序列化查询结果、为查询结果生成一个标识和怎么使用四个问题，具体如下：

(1). 防止脏读

对一张表的查询结果放在一个哈希结构里，当对这个表进行修改、删除或者更新时，删除该哈希结构。对这张表所有的操作方法，使用注解进行标记，

　　我们缓存了查询结果，那么一旦数据库中的数据发生变化，缓存的结果就不可用了。为了实现这一保证，可以在执行相关表的更新查询(update,delete,insert)查询前，让相关的缓存过期。这样下一次查询时程序就会重新从数据库中读取新数据缓存到redis中。那么问题来了，在执行一条insert前我怎么知道应该让哪些缓存过期呢？对于Redis，我们可以使用Hash结构，让一张表对应一个Hash，所有在这张表上的查询都保存到该Hash下。这样当表数据发生变动时，直接让Set过期即可。我们可以自定义一个注解，在数据库查询方法上通过注解的属性注明这个操作与哪些表相关，这样在执行过期操作时，就能直接从注解中得知应该让哪些Set过期了。

(2). 序列化查询结果

利用JDK自带的ObjectInputStream/ObjectOutputStream将查询结果序列化成字节序列，即需要考虑Redis的实际存储问题。

(3). 为查询结果生成一个标识

被调用的方法所在的类名，被调用的方法的方法名，该方法的参数三者共同标识一条查询结果。也就是说，如果两次查询调用的类名、方法名和参数值相同，我们就可以确定这两次查询结果一定是相同的（在数据没有变动的前提下）。因此，我们可以将这三个元素组合成一个字符串做为key，就解决了标识问题。

(4). 以 AOP 方式使用Redis

方法被调用之前，根据类名、方法名和参数值生成Key；

通过Key向Redis发起查询；

如果缓存命中，则将缓存结果反序列化作为方法调用的返回值 ，并将其直接返回；

如果缓存未命中，则继续向数据库中查询，并将查询结果序列化存入redis中，同时将查询结果返回。

例如，插入删除缓存逻辑如下：

/\*\*

\* 在方法调用前清除缓存，然后调用业务方法

\* @param jp

\* @return

\* @throws Throwable

\*/

@Around("execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.insert\*(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.update\*(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.delete\*(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.increase\*(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.decrease\*(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.complaint(..))" +

"|| execution(\* com.fh.taolijie.dao.mapper.JobPostModelMapper.set\*(..))")

public Object evictCache(ProceedingJoinPoint jp) throws Throwable {}

8.什么是存储过程？有哪些优缺点？

存储过程是事先经过编译并存储在数据库中的一段SQL语句的集合。进一步地说，存储过程是由一些T-SQL语句组成的代码块，这些T-SQL语句代码像一个方法一样实现一些功能（对单表或多表的增删改查），然后再给这个代码块取一个名字，在用到这个功能的时候调用他就行了。存储过程具有以下特点：

存储过程只在创建时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次，所以使用存储过程可提高数据库执行效率；

当SQL语句有变动时，可以只修改数据库中的存储过程而不必修改代码；

减少网络传输，在客户端调用一个存储过程当然比执行一串SQL传输的数据量要小；

通过存储过程能够使没有权限的用户在控制之下间接地存取数据库，从而确保数据的安全。

9.简单说一说drop、delete与truncate的区别

SQL中的drop、delete、truncate都表示删除，但是三者有一些差别：

Delete用来删除表的全部或者一部分数据行，执行delete之后，用户需要提交(commmit)或者回滚(rollback)来执行删除或者撤销删除， delete命令会触发这个表上所有的delete触发器；

Truncate删除表中的所有数据，这个操作不能回滚，也不会触发这个表上的触发器，TRUNCATE比delete更快，占用的空间更小；

Drop命令从数据库中删除表，所有的数据行，索引和权限也会被删除，所有的DML触发器也不会被触发，这个命令也不能回滚。

因此，在不再需要一张表的时候，用drop；在想删除部分数据行时候，用delete；在保留表而删除所有数据的时候用truncate。

10.什么叫视图？游标是什么？

视图是一种虚拟的表，通常是有一个表或者多个表的行或列的子集，具有和物理表相同的功能，可以对视图进行增，删，改，查等操作。特别地，对视图的修改不影响基本表。相比多表查询，它使得我们获取数据更容易。

游标是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

在操作mysql的时候，我们知道MySQL检索操作返回一组称为结果集的行。这组返回的行都是与 SQL语句相匹配的行（零行或多行）。使用简单的 SELECT语句，例如，没有办法得到第一行、下一行或前 10行，也不存在每次一行地处理所有行的简单方法（相对于成批地处理它们）。有时，需要在检索出来的行中前进或后退一行或多行。这就是使用游标的原因。游标（cursor）是一个存储在MySQL服务器上的数据库查询，它不是一条 SELECT语句，而是被该语句检索出来的结果集。在存储了游标之后，应用程序可以根据需要滚动或浏览其中的数据。游标主要用于交互式应用，其中用户需要滚动屏幕上的数据，并对数据进行浏览或做出更改。

11.乐观锁和悲观锁有什么区别

DBMS中的并发控制的任务是确保在多事务同时存取数据库中一数据时不被破坏事物的隔离性和统一性以及数据库的统一性。乐观并发控制和悲观并发控制是并发控制的两种方式。

悲观所，假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。

乐观锁：假定不会发生并发冲突，只是在提交时检查是否违反数据完整性。

12.分表之后想让一个id多个表是自增的，效率实现

5.MySql的主从实时备份同步的配置，以及原理(从库读主库的binlog)，读写分离 。