1. **在千万级的数据库查询中，如何提高效率**
2. 表设计
3. 将数据划分为两大类，热数据和冷数据

热数据 3个月内订单数据，查询实时性较高

冷数据A 3个月-12个月订单数据，查询频率不高

冷数据B 1年前的订单数据，几乎不会查询，只有偶尔查询

1. 将三类数据规划

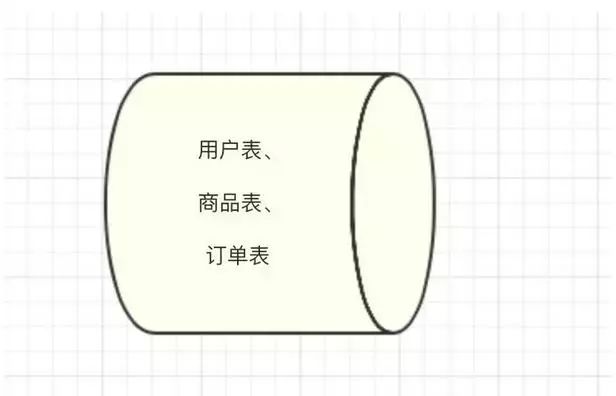
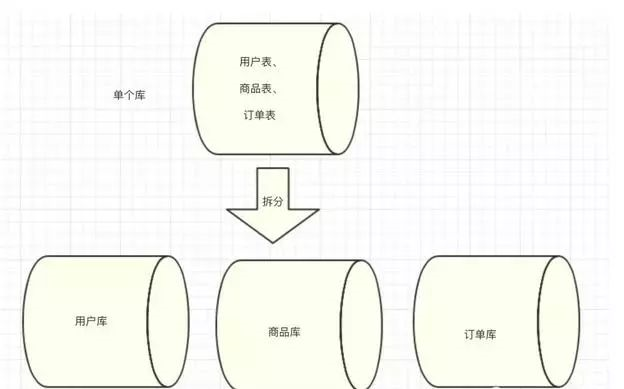
热数据 使用mysql/oracle存储，当然要分库分表

冷数据A可以存入Elasticsearch，利用搜随引擎也可以做到较快冷数据B 不经常查询数据当道Hive数据仓库中

需要将冷数据分别迁移到ES和hive中

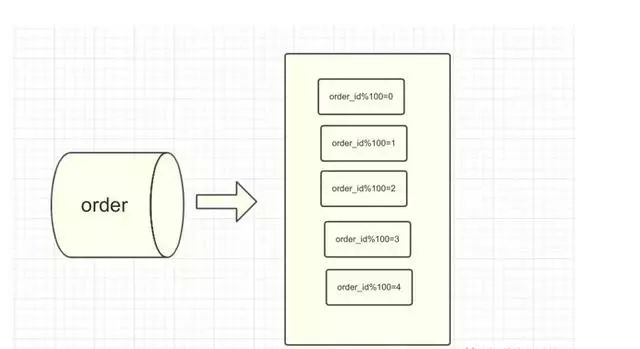
1. 分库策略

通常一半的电商平台包括用户、商品、订单等几大模块是在同一个库中建表，随着业务提升，所有业务放在一个库中变得越来越难维护，我们将不同的业务放到不同的库中

1. 分表策略

以订单为例，在订单表中订单ID肯定不重复对的，因此将该字段做shard key(路由) 非常合适，订单号做路由规则去对应的分表操作



1. 分表分表

常见的sharding中间件

cobar

阿里 b2b 团队开发和开源的，属于 proxy 层方案，就是介于应用服务器和数据库服务器之间。应用程序通过 JDBC 驱动访问 cobar 集群，cobar 根据 SQL 和分库规则对 SQL 做分解，然后分发到 MySQL 集群不同的数据库实例上执行。早些年还可以用，但是最近几年都没更新了，基本没啥人用，差不多算是被抛弃的状态吧。而且不支持读写分离、存储过程、跨库 join 和分页等操作。

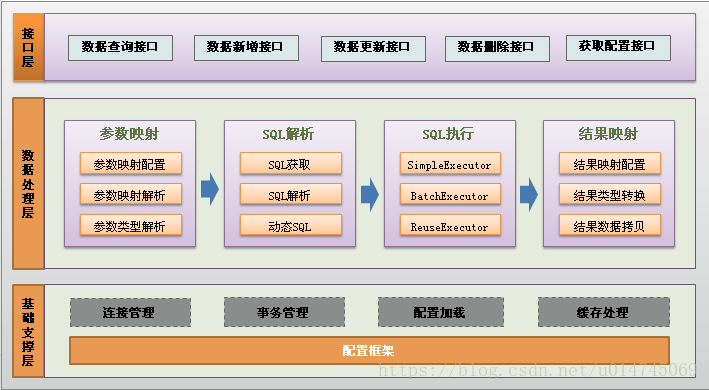
TDDL  
淘宝团队开发的，属于 client 层方案。支持基本的 crud 语法和读写分离，但不支持 join、多表查询等语法。目前使用的也不多，因为还依赖淘宝的 diamond 配置管理系统。

atlas  
360 开源的，属于 proxy 层方案，以前是有一些公司在用的，但是确实有一个很大的问题就是社区最新的维护都在 5 年前了。所以，现在用的公司基本也很少了。

sharding-jdbc  
当当开源的，属于 client 层方案。确实之前用的还比较多一些，因为 SQL 语法支持也比较多，没有太多限制，而且目前推出到了 2.0 版本，支持分库分表、读写分离、分布式 id 生成、柔性事务（最大努力送达型事务、TCC 事务）。而且确实之前使用的公司会比较多一些（这个在官网有登记使用的公司，可以看到从 2017 年一直到现在，是有不少公司在用的），目前社区也还一直在开发和维护，还算是比较活跃，个人认为算是一个现在也可以选择的方案。

mycat  
基于 cobar 改造的，属于 proxy 层方案，支持的功能非常完善，而且目前应该是非常火的而且不断流行的数据库中间件，社区很活跃，也有一些公司开始在用了。但是确实相比于 sharding jdbc 来说，年轻一些，经历的锤炼少一些。

1. 查询优化
2. 尽量避免全表扫描，首先应在where order by建立索引
3. 避免在where子句上进行null值判断，否则将导致放弃使用索引而进行全表扫描
4. 索引列有大量重复数据是查询可能不会去走索引
5. 索引并非越多越好，索引可以提高select效率，但也同事降低insert/update效率，应为insert/update有时可能会重建索引
6. 尽量避免更新索引，因为索引就是记录物理排列顺序，一单改值改变将导致整个表的记录顺序改变，会消耗相当大的资源
7. 尽量使用数字型字段，应为数字型字段只比较一次，字符型要挨个字段去比较
8. 尽量使用varchar/nvarchar代替char/nchar，节约存储空间
9. 查询优化
10. 避免在where子句中使用 != <> not in 等操作符
11. 避免在where子句中使用表达式 where num/2=100 改为num= 100\*2
12. 避免在where字句中使用函数操作  where substring(name,1,3)=‘abc
13. 避免在where子句中=左边进行函数运行
14. 使用exists 代替 in 避免使用select \* from，使用具体字段代替\*
15. **Spring MVC面试题**
16. SpringMVC基本原理
17. 用户向服务发送请求通过SpringMVC中的前端控制器DispatchServlet捕获到
18. DispatchServlet通过对Url解析获取到资源标识符URL，然后调用handlerMapping将请求映射到handleExecutorChain
19. DispatchServlet根据获取到的handler选择一个合适的DandlerAdapter适配器处理
20. Handler处理完成后返回一个ModeAndView给DispatchServlet
21. Handler返回ModeAndView是一个逻辑试图，DispatchServlet需要调用ViewResolver解析师徒转化为真正的view
22. DispatchServlet通过model解析ModeAndView中的参数将其转化为一个完整的View视图并返回给客户端
23. MVC基本原理
24. Model 是应用程序处理程序数据逻辑的部分，常用语数据库存取数据
25. View 是应用程序处理数据显示的部分
26. Controller 是应用程序处理数据交互的部分，通常又控制器负责读数据，控制用户输入，并向模型发送数据
27. SpringMVC常用注解
28. @Controller
29. @ RequestMapping
30. @Autowired
31. @ requestParam
32. @ResposeBody
33. @Component
34. 乱码解决
35. Web.xml 配置编码过滤器encodingFilter
36. @RequestMapping注解中配置 produces = "application/json; charset=utf-8"，缺点是每个方法都要配。
37. Spring-MVC.xml中配置字符串转换器取代默认转换器
38. Sring容器的主要核心是
39. 控制反转IOC 传统的java开发模式中，当我们需要一个对象时我会自己new 或getInstace去创建一个，而在Spring开发模式中spring容器使用了工厂模式为我们创建了我们所需的对象，不需要我们自己创建了，直接调用spring提供的对象就可以了
40. 依赖注入DI spring使用JavaBean对象的set方法或者构造器方法为我们创建对象时将其熟悉自动设置所需要的值得过程就是依赖注入
41. 面向切面编程AOP 将一个个对象某些类似的方面横向抽成一个切面，对这个切面进行一些如权限控制，事务管理，记录日志等公共操作的处理过程就是面向切面编程，AOP底层是动态代理
42. Sring设计模式有哪些
43. 单例模式
44. 模板方式
45. 前端控制器模式
46. 视图帮助
47. 依赖注入
48. 工厂模式
49. **Mybatis**
50. Mybatis原理
51. 加载mybatis全局配置文件（数据源、Mapper文件等）、基于XML配置文件生成Configuration和MappedStatement包括了参数映射配置、动态SQL语句、结果映射配置
52. SqlSessionFactoryBuilder通过Configuration 构建生成SqlSessionFactory用来开启SqlSession
53. SqlSession用于完成和数据库的交互
54. 用户调用Mybatis的接口层Api（Dao层接口）
55. SqlSession通过调用Api的StatementId查找到对应的MappedStatement对象
56. 通过Executor将MappedStatement对象进行解析，sql参数化，动态sql拼接，生成jdbc Satatement
57. Jdbc执行sql
58. 借助MappedStatement中的结果映射关系转化结果并返回



1. Mybatis 中一级缓存与二级缓存？
2. 一级缓存：基于HashMap的本地缓存，其作用域为Session，当fluse或close后Cache被清空，默认开启
3. 二级缓存也是基于HashMap的存储，其作用域为Mapper，可自定义存储源，作用域namespace所对应的的配置文件中select操作结果都被缓存，这样不同线程下就可以公用