# 数据库

## 机票系统

查看数据库机器状态：

http://mdc.jd.com/monitor/chart?ip=10.191.202.105

主库：

IP: 10.191.202.105:3358

域名：airplanem.mysql.jddb.com

从库：

IP: 10.187.4.142:3358

域名: airplanesa.mysqlmjq.jddb.com

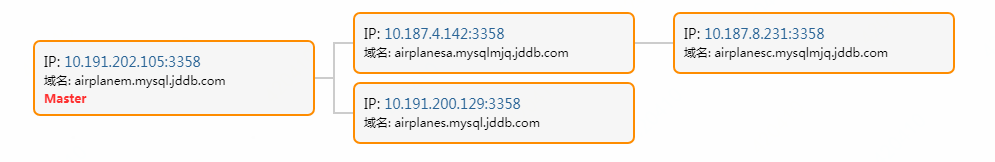
IP: 10.191.200.129:3358

域名: airplanes.mysql.jddb.com

IP: 10.187.8.231:3358

域名: airplanesc.mysqlmjq.jddb.com

<dbs.jd.com>中，数据库服务-主从架构查询



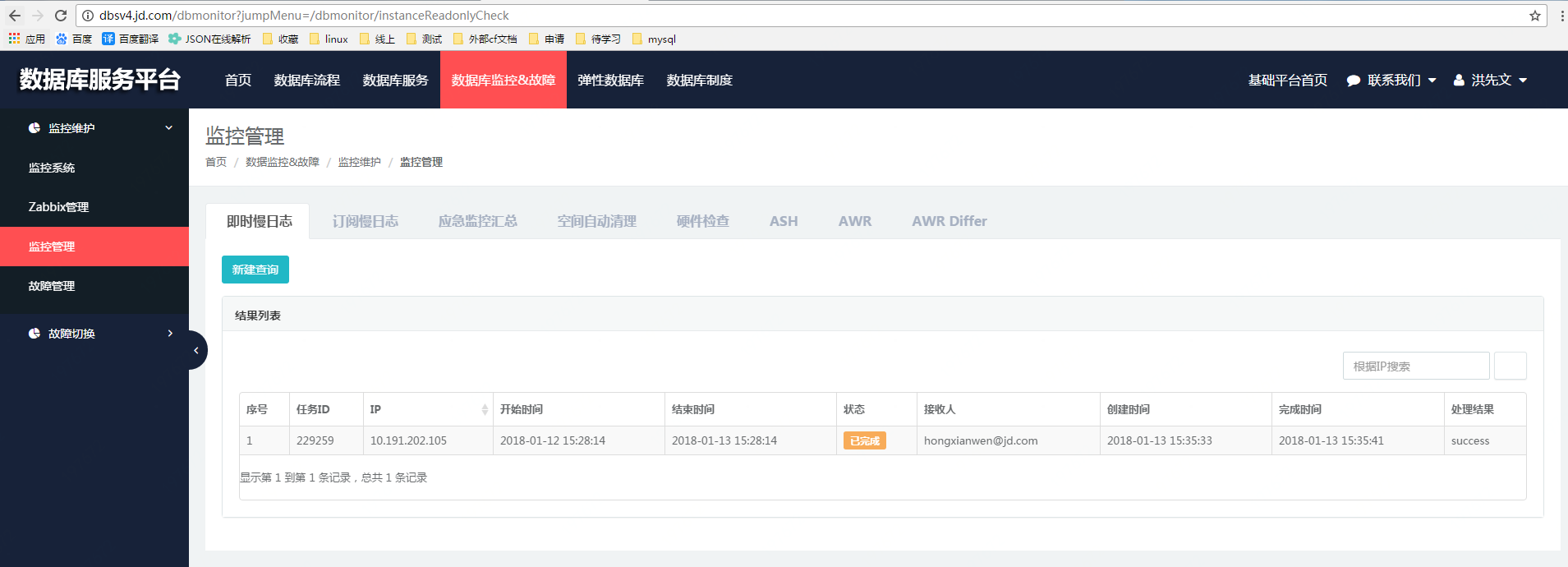


### 慢查询日志

数据库服务平台：

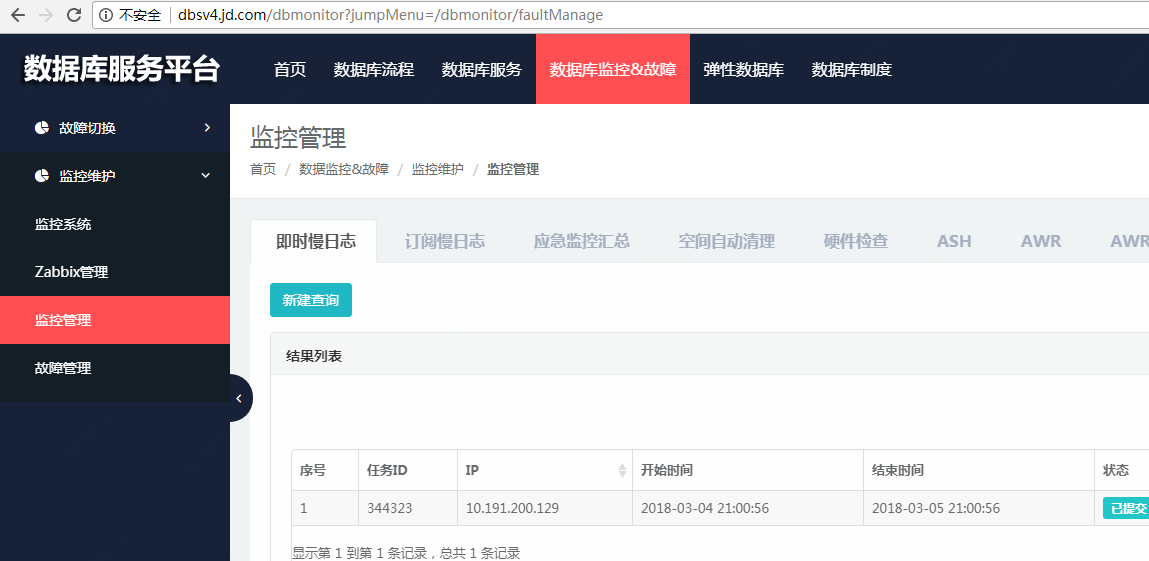
<http://dbsv4.jd.com/dbmonitor?jumpMenu=/dbmonitor/instanceReadonlyCheck>

提供了监控管理工具



将数据库的ip地址输入





可以订阅慢日志，发送到指定邮箱

#### 执行计划explain

用法：explain + sql

显示结果：



结果字段解释：

##### id

select 查询的序列号

##### select\_type

SIMPLE：查询中不包含子查询或者UNION

##### table

表名

##### type

type显示的是访问类型，是较为重要的一个指标，结果值从好到坏依次是：   
system > const > eq\_ref > ref > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL ，一般来说，得保证查询至少达到range级别，最好能达到ref。

**const**

一般是查询条件使用了唯一主键id时会发生，比如：

select \* from shop\_order where id =929;

**eq\_ref**

唯一性索引扫描，对于每个索引键，表中只有一条记录与之匹配

**ref**

非唯一性索引扫描，返回匹配某个单独值的所有行

示例：select \* from shop\_order where erp\_order\_id =60286843215;

**range**

扫描部分索引树，比如使用>或<或in查找东西时发生的情况

示例：

select \* from shop\_order where erp\_order\_id in (60286843215,60270391333);

**index**

扫描全部索引树

**ALL**

使用了全表扫描

##### possible\_keys

显示可能应用在这张表中的索引

##### key

实际使用的索引

##### key\_len

使用的索引的长度。在不损失精确性的情况下，长度越短越好

##### ref

显示索引的哪一列被使用了，如果可能的话，是一个常数

##### rows

MYSQL认为必须检查的用来返回请求数据的行数

##### Extra

Only index，这意味着信息只用索引树中的信息检索出的，这比扫描整个表要快。

using where是使用上了where限制，表示MySQL服务器在存储引擎受到记录后进行“后过滤”（Post-filter），如果查询未能使用索引，Using where的作用只是提醒我们MySQL将用where子句来过滤结果集。

impossible where 表示用不着where，一般就是没查出来啥。

Using filesort（MySQL中无法利用索引完成的排序操作称为“文件排序”）当我们试图对一个没有索引的字段进行排序时，就是filesoft。它跟文件没有任何关系，实际上是内部的一个快速排序。

Using temporary（表示MySQL需要使用临时表来存储结果集，常见于排序和分组查询），使用filesort和temporary的话会很吃力，WHERE和ORDER BY的索引经常无法兼顾，如果按照WHERE来确定索引，那么在ORDER BY时，就必然会引起Using filesort，这就要看是先过滤再排序划算，还是先排序再过滤划算。

### Zabbix管理

# 限流、分流

### Nginx

1、针对固定IP进行限流, 这种策略主要用于同一个IP反复请求服务器,类似于洪水攻击或者DDos攻击,对单一ID进行限制可以有效的。

2、针对IP限制并发请求

3、配置白名单：白名单的作用就是 对某些特定的IP不做请求限制,比如我们自己的IP,或者公司内网的IP等

4、负载均衡可能也算一种吧 , 当然他是把自己的流量倒到别的机子上去而已

### 单个JVM下的接口限流

#### 方案一

使用jdk自带的concureent包下的信号量Semaphore实现

超过阀值的线程在队列中等待

#### 方案二



与分布式不一样的地方，就是key命名，每个JVM下的key都是不一样的

### 分布式环境下的接口限流



各个jvm下的key值是一样的

### 分布式环境下调用外部接口限流

与分布式限流类似，控制的点房贷调用外部系统那

# 高可用设计

## 降级

### 商家接口

有手动降级开关，从产品查询处，屏蔽该商家产品展示。

支持自动升、降级

### 优惠券、促销

手动降级，不再支持优惠券、促销

## 容灾

1、系统多机房部署。

2、数据库主从灾备。

3、支付反查。

4、接口异常重试、报警机制。

5、自动容错：缓存失效、或者缓存宕机，从数据库或者zk中拿数据，或者通过接口实时获取

# 大促期间，故障演练

## Tomcat宕机

现象：UMP监控到实例：10.191.74.40，收到UMP报警

操作步骤：

1、np.jd.com 中 从各组vip中摘除 ip

2、从j-one中启动实例

## cpu暴增

现象：10.191.74.40，收到UMP CUP使用率99.95% 报警，UMP监控显示CPU 过高。

操作：

1. np.jd.com 中 从各组vip中摘除 相应IP的实例

2、 运维解决后，将这俩实例重写加入到VIP中

## 磁盘暴增

现象：实例：10.187.107.160 磁盘使用率过高。收到报警短信

操作：

1、根据告警，从np.jd.com 中摘除实例

2、bmt平台中选中故障实例，清理磁盘

3、运维回复后 实例加入VIP

## 网络阻塞

实例：10.187.107.160 收到UMP 目标地址网络异常报警

1、从np.jd.com 从VIP中摘除实例

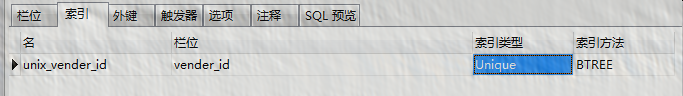
2、运维回复后 实例加入VIP

# mysql的乐观锁

**sql语句**

INSERT INTO AIR\_PROM\_BDATA\_ACTIVITY  
 (  
 `ID`,  
 `ACT\_ID`,  
 `ACT\_NAME`,  
 `BEGIN\_TIME`,  
 `END\_TIME`,  
 `CREATOR`,  
 `ACT\_STATUS`,  
 `USER\_TYPE`,  
 `USER\_LEVEL`,  
 `CREATED`,  
 `MODIFIED`)  
 SELECT  
 #{id},  
 #{activityId},  
 #{name},  
 #{beginTime},  
 #{endTime},  
 #{creator},  
 #{statusEnum,typeHandler=com.jd.airplane.base.dao.handler.ActivityStatusEnumCodeHandler},  
 #{promotionEnum,typeHandler=com.jd.airplane.base.dao.handler.PromotionEnumCodeHandler},  
 #{userLevelEnum,typeHandler=com.jd.airplane.base.dao.handler.UserLevelEnumCodeHandler},  
 now(),  
 now()  
FROM  
 dual  
WHERE NOT *EXISTS*(SELECT 1 FROM AIR\_PROM\_BDATA\_ACTIVITY WHERE ACT\_ID = #{activityId})

或者在数据表中添加索引



# 索引

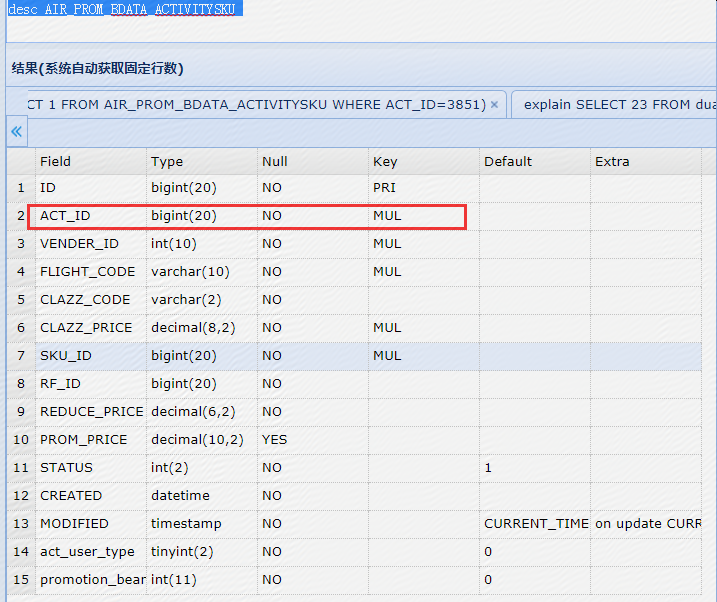
Show index

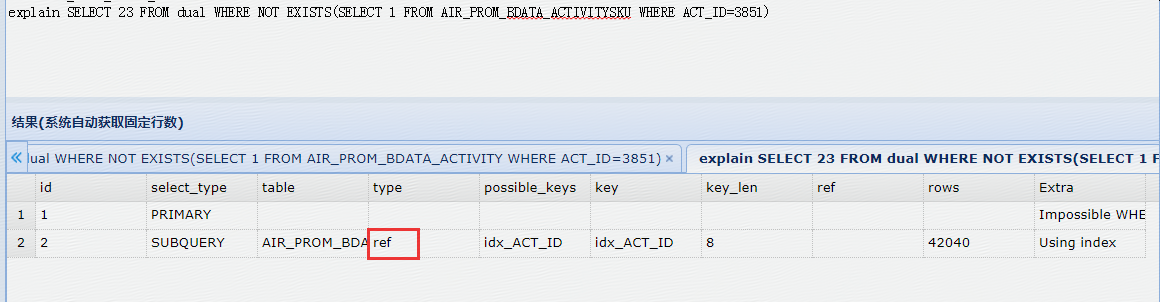
[**MySQL SHOW INDEX语法的实际应用**](https://blog.csdn.net/zhxp_870516/article/details/8434539)

[**mysql索引总结----mysql 索引类型以及创建**](http://www.cnblogs.com/lihuiyong/p/5623191.html)

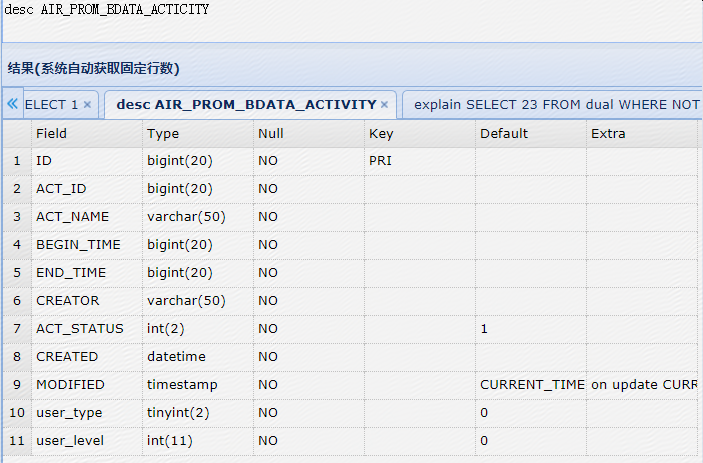
air\_prom\_bdata\_activitySku表中有700多万条数据，但是在ACT\_ID上创建索引后，rows为4万多。

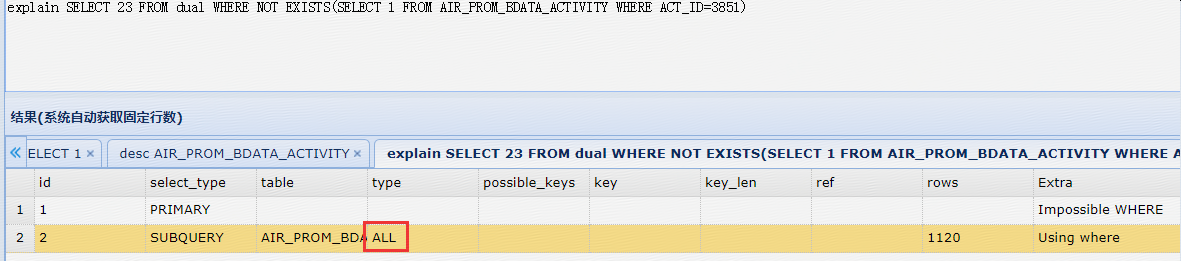
ALTER TABLE air\_prom\_bdata\_activitysku ADD INDEX idx\_act\_id (act\_id)





air\_prom\_bdata\_activity表没有索引，所以type是all，全表扫描





# 死锁问题-update/insert

[Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException: Deadlock found when tryin](https://blog.csdn.net/zheng0518/article/details/54695605)

Do not use index merge when single index is good enough

Try to avoid using index merge in UPDATE to not provoke deadlocks

使用多个索引，多线程去写数据

# 主从查询

air\_base库：





# dbQuery申请

<http://dbsv4.jd.com/workflow?jumpMenu=/workflow/DbqueryApply>

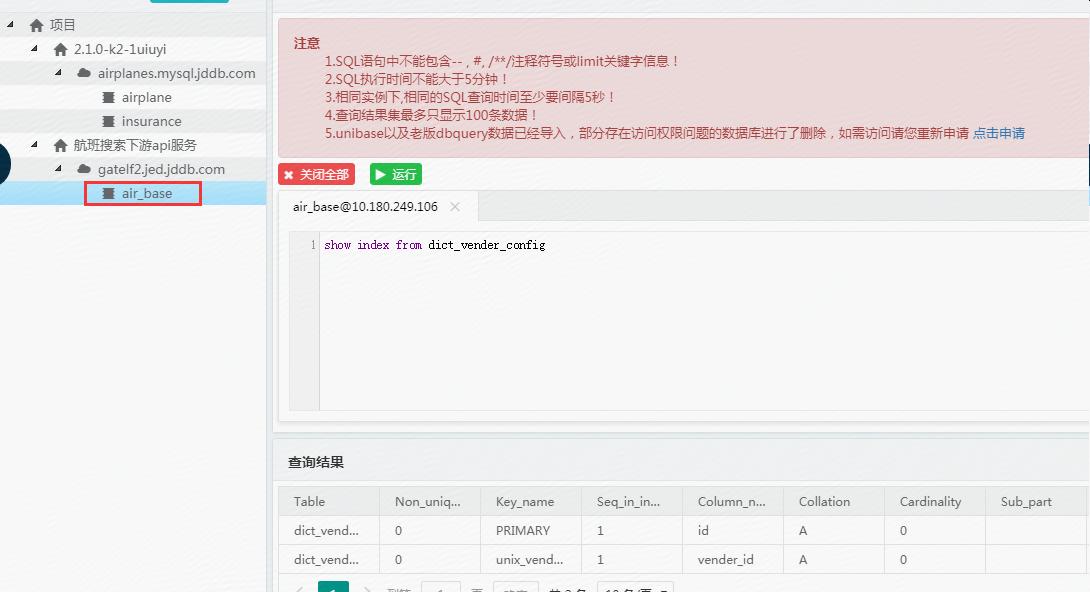
填写申请单，冬哥审批就可以





提交申请就可以





# 线上sql执行

添加字段：ALTER TABLE dict\_vender\_config ADD COLUMN `cache\_time\_factor` int(11) unsigned NOT NULL DEFAULT '0' COMMENT '缓存时间系数';

修改字段自增：

alter table rule\_airline modify column `id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT COMMENT '主键';