**企业实际**

**（20181118）**

# DBA软要求

## 态度积极

责任心

数据库往往是企业的心脏，DBA 必须对于公司有超额恪尽职守的觉悟。不论是半夜三点还是隔着四分之一个地球，一个电话打过去就可以如临现场。

抗压

数据库宕机的每一秒钟都在给企业造成损失，对内无法开工，对外无法服务。电话此起彼伏，几十双眼睛盯着他的时候，仍旧能保持沉着，方寸不乱。

能独立工作

不仅和别人搭档的时候一切正常，独立要完成任务的时候也不会不知所措，不会怀疑自己做的对不对、质疑这样行不行。没有同事的时候，不会分心，也不偷懒。

善于情势判断

两方面：一是数据库宕机时往往是连锁反应，症状复杂，此时要快速找到根源，所需要的排查能力；二是修复时能分清轻重缓急，知道应该先通知前端把网站转入维护状态，然后去换烧坏的硬盘控制器，最后才是去看上一次备份是在半个月 前还是半年之前。

细致周全

Schema / Index / Temp table 设计合理。防御式 ETL，设定层级报警。

文档写得细致干净，逻辑清晰。

做事规律

知道上一次备份是在几小时之前；

定期打补丁；定期查看系统记录；定期更换密码；硬盘排线要上束绳、打标签；记笔记——如果他不幸身故，同事可以通过他的笔记本迅速接过整个业务。

耐心

解决问题或者和别人沟通时，不会轻易不耐烦 / 愤怒 / 产生挫折感 / 进入防御模式。

略有洁癖

主要是对于数据上的。

善于沟通

口拙做不了 DBA，不管是解释事故原因还是争取检修机会 / 硬件升级资金。

对硬件着迷

不解释。

精通不包括 shell script 在内的至少一种脚本语言不解释。

不爱出风头

其实没什么不好，但是这一条往往和以上所有品质不协调。

最适合担任 DBA 的人，是英语好，已婚，身体强健，生活稳定，无不良嗜好的中年工程师（不，我不是。这个答案本质上是在描述我的两个同事）。——真正优秀的 DBA，是平时看起来无所事事的人，他们可以做救火英雄，但他们每天都在避免救火英雄故事的出现。请付出对得起他们的薪水。

--------------------------------------------------------------------------------------------------------

作者：涛吴

链接：https://www.zhihu.com/question/20112937/answer/14017416

来源：知乎

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 熟悉业务

1.产品类别

2.用户行为

3.熟悉行业：

选择最主流的数据库软件学习，提升自我价值

4.综上产生的需求：

满足以上业务需要部署的软件

PS：懂业务才能成为架构师

## 技术

1.熟悉操作系统

2.精通mysql的原理

3.每个命令敲下去的后果

4.为什么要这么做？这样做的好处是？有没有更好的方法…..

## 喜欢数据库

1.三维本质

# DBA硬要求

## 应用API及获取方法（非SA职责范围）

1.应用连接数据库需要加载对应语言程序的API

PS：mysql提供了多种语言的接口程序包

e.g：比如php-mysql连接驱动，就是mysql提供的与php的驱动程序；网卡连接，需要网卡驱动程序……

2.那么如何获取与mysql连接的驱动程序？

mysql官网→downloads→connector

## 如何判断mysql运行正常，可以对外服务？

Q1：怎么判断自己的mysql数据可以对外提供服务

A1：

1）确认服务正常

netstat -lnp|grep 3306或者ps -ef|grep mysql

2)通过TCP/IP方式测试连接

mysql -uwordpress -p123 -h10.0.0.52

3）非SA职责：比如php-mysql连接驱动，就是mysql提供的与php的驱动程序；网卡连接，需要网卡驱动程序……

## DBA与开发人员关于设计数据库的时候

开发人员设计数据库的时候，DBA必须在场，因为开发人员在导入数据后，可能会忘记创建索引，导致数据库可用性不高

---可以在开发人员建好表以后，在添加索引

# 企业面试题

## 你上一家公司使用的是什么版本数据库？？？

面试场景：

1、你对数据库熟悉吗，看你简历数据库会的挺多

2、那你们原来公司用的什么版本？

参考回答：

第一种情况：5.6.34/5.6.36/5.6.38----稳定，已经使用了2年，没有出现过任何因为版本出现的问题

第二种情况：5.7.18/5.7.20

-----我们公司新上的项目使用了5.7.20这个版本中的高可用架构（MGR）

（1）是啊，我主要负责的数据库方面的日常运维工作。

（2）我们公司使用的是5.6.38这个版本。

（3）为什么使用这个版本哪呢？

（4）因为，我们公司的业务系统是基于这个版本开发的，

（5）经过长时间的测试以及运行，没有发现明显异常，比较稳定

（6）基于应用的兼容性思考，我们暂时没有打算上5.7。

（7）但是我们已经准备好将来在必要的时候，更新为新版本

（8）我对5.7的新特性也比较了解。

PS：

选择的版本号一般是偶数---因为偶数版本一般从测试的单数版本改进而来

## 开发人员需求：请给我开一个用户

【背景知识】

开发环境：windows MySQL

测试环境：Linux MySQL

生产环境：Linux MySQL

【沟通事项】

1.你需要对那些库或者表做开发操作？

2.进行哪些操作（权限）？

3.你从哪（网段）连接过来？

4.用户名和密码有没有要求？

==密码不能太简单，否则以后出了问题SA也得担责。

【授权命令】

grant select,update,delete,insert on oldboy.\* to [oldboy@’10.0.0.%’identified](mailto:oldboy@'10.0.0.%25'identified) by ‘123’;

#select,update,delete,insert ---设定可操作权限为更新、删除和插入

## 开发人员又说：把root权限给我呗？？？

SA：

根本原则：不给，因为哪怕是开发人员误操作了，运维也要负起大部分责任

1）刚入职，在公司位置比较低：

拿流程和规范拒绝他，请找我们部门领导审批签字，CTO签字。

2）非常牛：相信自己是最专业的！直接拒绝

→为什么DBA受重视？

→因为他掌握的是公司的核心数据库，技术要求也比较高。

3）权利与义务等价交换思维方式

--如果你要root权限，那这块管理就全部给你了，我以后就不管了……

# 企业实际业务

## 数据库业务特点

数据库DB-----通过-----NBU（备份软件）-----通过log—查看数据库运行情况----

--------备份到-----磁带库-----（需要做定期演练，备份到测试库）---因为磁带库只能通过NBU软件才能操作。

## 淘宝架构

【前端】

java

【后端】

LNT+Alisql

alisql---阿里二次开发的数据库

数据库的缓存：

tair/redis or memecache二次开发的（双11前两周，热点数据需要导入到缓存）

MQ---队列

商品的队列导航/索引存储的数据库----mongodb

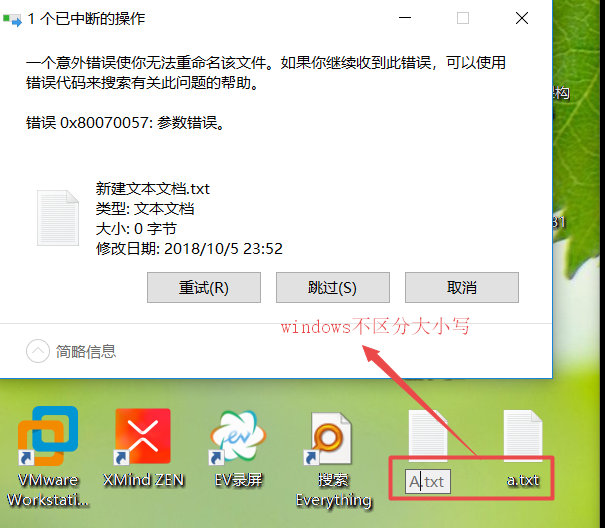
商品搜索/索引—ES----ELEcti search？？？？

支付业务—后端的OB---osenbase(有银行的接口)

## windows平台上开发创建的库、表导入到linux系统问题之库、表名称大小写（可作为简历“碰到的问题”或面试中“公司出现的问题”）

【背景知识】

1.windows系统创建文件，文件名称不区分大小写



2.linux系统创建文件，文件名称区分大小写

[root@db02 tmp]# touch a

[root@db02 tmp]# touch A

[root@db02 tmp]# ll

total 16

-rw-r--r-- 1 root root 0 Oct 5 23:54 a

-rw-r--r-- 1 root root 0 Oct 5 23:54 A

【问题及原因】

开发阶段在windows平台上对库、表的命名用大写字母，但是在代码中使用数据库又使用小写字母应用（windows不区分字母大小写），库、表导入到linux系统（区分字母大小写），数据库在使用的时候一直报错

【解决办法及优化】

#解决办法

企业实际：有一个业务在开发阶段的时候，创建库、表的时候，使用了大写字母，但是开发代码应用数据库的时候，使用的是小写字母，导致数据库在使用的时候一直报错

#在linux中有一个将大小写的字母全部转换为小写字母，但是不能对已有数据库更新，所以一般用于数据库刚刚搭建的时候。

#千万别在已经运行的数据库使用参数“lower\_case\_table\_names”

#如果担心开发阶段使用了大写字母命名库、表，可以加上如下参数“lower\_case\_table\_names”

mysql> show variables like '%lower%';

+------------------------+-------+

| Variable\_name | Value |

+------------------------+-------+

| lower\_case\_file\_system | OFF |

| lower\_case\_table\_names | 0 |

+------------------------+-------+

2 rows in set (0.01 sec)

mysql>

#优化

订立规则：

凡是涉及的数据库的库命名的均使用小写字母串

## 定责：如何查看开发人员建库建表的语句，分析问题的根源。

show create database oldboy;

show create table t1;

## 字符集举例一：

xshell（gbk） --发出请求data---》linux（utf8）----发送给--》mysql---》linux---》shell

### 说明一：

上述设置不会出现乱码，因为linux和mysql中设置的utf8包含了gbk

## 程序代码级-业务

### 生产环境更改数据库（含数据）字符集的方法

【需求】

发现设计的库、表有问题，需要修改其字符集，比如需要往表中存入中文

【思路】

第一种情况：设计阶段

清空，然后重新导入库表---因为数据库中数据没有或只有小部分数据

第二种情况：数据运行了一段时间

--比如导入法文的时候，出现乱码情况

--做好备份，并模拟恢复，成功以后再进行恢复

--‘严格超集’—控制作用范围（使得影响范围最小，改小不改大）

---尽量不要改

【修改命令】

alter database oldboy CHARACTER SET utf8 collate utf8\_general\_ci;

alter table t1 CHARA

## 使用伪删除

### 在表中添加一列状态信息

ALTER TABLE test ADD state ENUM(‘0’,’1’) DEFAULT ‘1’ AFTER AGE;

### 将需要删除的列的状态值改为“0”—定义为“删除”（实际未删除）

UPDATE test SET state=’0’WHERE id=1;

### select查询时，排除已“删除”的内容

SELECT \* FROM test WHERE state=’1’;

## 使用触发器

能不用trigger，尽量不要使用，因为它会影响数据库性能

## 数据库巡检？？？--未解决问题

## 数据库存储引擎innodb与myisam的区别

|  |  |  |
| --- | --- | --- |
| Innodb | MyISAM | note |
| MVCC(multi-versioning concurrency control)多版本并发控制 | 不支持 |  |
| 锁（行级别） | 表级别 |  |
| 外键 | 不支持 |  |
| 事务 | 不支持 | 重点 |
| hot热备 | 温备 |  |
| CSR(crash safety recovery)故障自动回复（理论能恢复断电之前的状态） | 不支持 |  |
|  |  |  |



## 建立索引的规范

建立索引的原则（运维规范）

select count(distinct user) from mysql.user;

select count(distinct user,host) from mysql.user;

# 数据库索引的设计原则：

为了使索引的使用效率更高，在创建索引时，必须考虑在哪些字段上创建索引和创建什么类型的索引。

那么索引设计原则又是怎样的?

## 0.建表时一定要有主键,如果相关列可以作为主键,做一个无关列

## 1．选择唯一性索引

唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录。

例如，学生表中学号是具有唯一性的字段。为该字段建立唯一性索引可以很快的确定某个学生的信息。

如果使用姓名的话，可能存在同名现象，从而降低查询速度。

主键索引和唯一键索引，在查询中使用是效率最高的。

select count(\*) from world.city;

select count(distinct countrycode) from world.city;

select count(distinct countrycode,population ) from world.city;

注意：如果重复值较多，可以考虑采用联合索引

## 2．为经常需要排序、分组和联合操作的字段建立索引

经常需要ORDER BY、GROUP BY,join on等操作的字段，排序操作会浪费很多时间。

如果为其建立索引，可以有效地避免排序操作。

## 3．为常作为where查询条件的字段建立索引

如果某个字段经常用来做查询条件，那么该字段的查询速度会影响整个表的查询速度。因此，

为这样的字段建立索引，可以提高整个表的查询速度。

3.1 经常查询

3.2 列值的重复值少(业务层面调整)

注：如果经常作为条件的列，重复值特别多，可以建立联合索引。

## 4．尽量使用前缀来索引

如果索引字段的值很长，最好使用值的前缀来索引。例如，TEXT和BLOG类型的字段，进行全文检索

会很浪费时间。如果只检索字段的前面的若干个字符，这样可以提高检索速度。

------------------------以上的是重点关注的，以下是能保证则保证的--------------------

## 5．限制索引的数目

索引的数目不是越多越好。每个索引都需要占用磁盘空间，索引越多，需要的磁盘空间就越大。

修改表时，对索引的重构和更新很麻烦。越多的索引，会使更新表变得很浪费时间。

## 6．删除不再使用或者很少使用的索引(percona toolkit)

表中的数据被大量更新，或者数据的使用方式被改变后，原有的一些索引可能不再需要。数据库管理

员应当定期找出这些索引，将它们删除，从而减少索引对更新操作的影响。

## 7.大表加索引,要在业务不繁忙期间操作

建索引原则

(1) 必须要有主键,如果没有可以做为主键条件的列,创建无关列

(2) 经常做为where条件列 order by group by join on的条件(业务:产品功能+用户行为)

(3) 最好使用唯一值多的列作为索引,如果索引列重复值较多,可以考虑使用联合索引

(4) 列值长度较长的索引列,我们建议使用前缀索引.

(5) 降低索引条目,一方面不要创建没用索引,不常使用的索引清理,percona toolkit

(6) 索引维护要避开业务繁忙期

===============================================

# 不走索引的情况（开发规范）

## 1) 重点关注：没有查询条件，或者查询条件没有建立索引

select \* from tab; 全表扫描。

select \* from tab where 1=1;

在业务数据库中，特别是数据量比较大的表。

是没有全表扫描这种需求。

1、对用户查看是非常痛苦的。

2、对服务器来讲毁灭性的。

（1）select \* from tab;

SQL改写成以下语句：

selec \* from tab order by price limit 10 需要在price列上建立索引

（2）

select \* from tab where name='zhangsan' name列没有索引

改：

1、换成有索引的列作为查询条件

2、将name列建立索引

## 2) 查询结果集是原表中的大部分数据，应该是25％以上。

查询的结果集，超过了总数行数25%，优化器觉得就没有必要走索引了。

假如：tab表 id，name id:1-100w ，id列有索引

select \* from tab where id>500000;

如果业务允许，可以使用limit控制。

怎么改写 ？

结合业务判断，有没有更好的方式。如果没有更好的改写方案

尽量不要在mysql存放这个数据了。放到redis里面。

## 3) 索引本身失效，统计数据不真实

索引有自我维护的能力。

对于表内容变化比较频繁的情况下，有可能会出现索引失效。

## 4) 查询条件使用函数在索引列上，或者对索引列进行运算，运算包括(+，-，\*，/，! 等)

例子：

错误的例子：select \* from test where id-1=9;

正确的例子：select \* from test where id=10;

算术运算

函数运算

子查询

## 5)隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误.

这样会导致索引失效. 错误的例子：

------------------------

mysql> alter table tab add index inx\_tel(telnum);

Query OK, 0 rows affected (0.03 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql>

mysql> desc tab;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| id | int(11) | YES | | NULL | |

| name | varchar(20) | YES | | NULL | |

| telnum | varchar(20) | YES | MUL | NULL | |

+--------+-------------+------+-----+---------+-------+

3 rows in set (0.01 sec)

mysql> select \* from tab where telnum='1333333';

+------+------+---------+

| id | name | telnum |

+------+------+---------+

| 1 | a | 1333333 |

+------+------+---------+

1 row in set (0.00 sec)

mysql> select \* from tab where telnum=1333333;

+------+------+---------+

| id | name | telnum |

+------+------+---------+

| 1 | a | 1333333 |

+------+------+---------+

1 row in set (0.00 sec)

mysql> explain select \* from tab where telnum='1333333';

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

| id | select\_type | table | type | possible\_keys | key | key\_len | ref | rows | Extra |

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

| 1 | SIMPLE | tab | ref | inx\_tel | inx\_tel | 63 | const | 1 | Using index condition |

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

1 row in set (0.00 sec)

mysql> explain select \* from tab where telnum=1333333;

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

| id | select\_type | table | type | possible\_keys | key | key\_len | ref | rows | Extra |

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

| 1 | SIMPLE | tab | ALL | inx\_tel | NULL | NULL | NULL | 2 | Using where |

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

1 row in set (0.00 sec)

mysql> explain select \* from tab where telnum=1555555;

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

| id | select\_type | table | type | possible\_keys | key | key\_len | ref | rows | Extra |

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

| 1 | SIMPLE | tab | ALL | inx\_tel | NULL | NULL | NULL | 2 | Using where |

+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

1 row in set (0.00 sec)

mysql> explain select \* from tab where telnum='1555555';

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

| id | select\_type | table | type | possible\_keys | key | key\_len | ref | rows | Extra |

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

| 1 | SIMPLE | tab | ref | inx\_tel | inx\_tel | 63 | const | 1 | Using index condition |

+----+-------------+-------+------+---------------+---------+---------+-------+------+-----------------------+

1 row in set (0.00 sec)

mysql>

---------------------------------------

## 6) <> ，not in 不走索引

EXPLAIN SELECT \* FROM teltab WHERE telnum <> '110';

EXPLAIN SELECT \* FROM teltab WHERE telnum NOT IN ('110','119');

------------

mysql> select \* from tab where telnum <> '1555555';

+------+------+---------+

| id | name | telnum |

+------+------+---------+

| 1 | a | 1333333 |

+------+------+---------+

1 row in set (0.00 sec)

mysql> explain select \* from tab where telnum <> '1555555';

-----

单独的>,<,in 有可能走，也有可能不走，和结果集有关，尽量结合业务添加limit

or或in 尽量改成union

EXPLAIN SELECT \* FROM teltab WHERE telnum IN ('110','119');

改写成：

EXPLAIN SELECT \* FROM teltab WHERE telnum='110'

UNION ALL

SELECT \* FROM teltab WHERE telnum='119'

-----------------------------------

## 7) like "%\_" 百分号在最前面不走

EXPLAIN SELECT \* FROM teltab WHERE telnum LIKE '31%' 走range索引扫描

EXPLAIN SELECT \* FROM teltab WHERE telnum LIKE '%110' 不走索引

%linux%类的搜索需求，可以使用elasticsearch 专门做搜索服务的数据库产品

## 8) 单独引用联合索引里非第一位置的索引列.作为条件查询时不走索引.

列子：

复合索引：

DROP TABLE t1

CREATE TABLE t1 (id INT,NAME VARCHAR(20),age INT ,sex ENUM('m','f'),money INT);

ALTER TABLE t1 ADD INDEX t1\_idx(money,age,sex);

DESC t1

SHOW INDEX FROM t1

走索引的情况测试：

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE money=30 AND age=30 AND sex='m';

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE money=30 AND age=30 ;

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE money=30 AND sex='m'; ----->部分走索引

不走索引的：

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE age=20

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE age=30 AND sex='m';

EXPLAIN SELECT NAME,age,sex,money FROM t1 WHERE sex='m';

# 企业案例事件

## 某企业断电后，mysql启动不了，数据库中有jira库和confluence库需要恢复

### background knowledge

#### fsck

fsck 重启 进行以下操作

#《==自动读取并整理inode，是inode与真实数据一一对应，不能对应的inode会被清理掉。

---》刚删除的文件可以使用文件句柄恢复

#### alter 。。。discard和import的区别

#warning：这个语句删除当前.ibd文件

alter table confulence.t1 discard tablespace;

#将confluence.t1中的ibd文件导入到表空间

alter table confulence.t1 import tablespace;

### 【环境】

jira(bug追踪) 、 confluence(内部知识库) ==107张表 ------>LNMT

-------------------

联想服务器（IBM）

磁盘500G 没有raid

centos 6.8

mysql 5.6.33 innodb引擎 独立表空间

备份没有，日志也没开

------------------

编译→制作rpm

/usr/bin/mysql

/var/lib/mysql

confulence jira

所有软件和数据都在"/"

------------

断电了，启动完成后“/” 只读

fsck 重启

启动mysql启动不了。

结果：confulence库 在 ， jira库不见了

------------

求助：

这种情况怎么恢复？

我问：

有备份没

求助：

连二进制日志都没有，没有备份，没有主从

我说：

没招了，jira需要硬盘恢复了。

求助：

1、jira问题拉倒中关村了

2、能不能暂时把confulence库(107张表)先打开用着

将生产库confulence，拷贝到1:1虚拟机上/var/lib/mysql,直接访问时访问不了的

问：有没有工具能直接读取ibd

我说：我查查，最后发现没有

我想出一个办法来：

create table xxx

alter table confulence.t1 discard tablespace;

alter table confulence.t1 import tablespace;

虚拟机测试可行。

CREATE TABLE `city\_new` (

`ID` int(11) NOT NULL AUTO\_INCREMENT,

`Name` char(35) NOT NULL DEFAULT '',

`CountryCode` char(3) NOT NULL DEFAULT '',

`District` char(20) NOT NULL DEFAULT '',

`Population` int(11) NOT NULL DEFAULT '0',

PRIMARY KEY (`ID`),

KEY `CountryCode` (`CountryCode`),

KEY `idx\_popu` (`Population`)

) ENGINE=InnoDB AUTO\_INCREMENT=4080 DEFAULT CHARSET=latin1;

面临的问题，confulence库中一共有107张表。

1、创建107和和原来一模一样的表。

他有2016年的历史库，我让他去他同事电脑上 mysqldump备份confulence库

mysqldump -uroot -ppassw0rd -B confulence --no-data >test.sql

拿到你的测试库，进行恢复

到这步为止，表结构有了。

2、表空间删除。

select concat('alter table ',table\_schema,'.'table\_name,' discard tablespace;') from information\_schema.tables where table\_schema='confluence' into outfile '/tmp/discad.sql';

source /tmp/discard.sql

执行过程中发现，有20-30个表无法成功。主外键关系

很绝望，一个表一个表分析表结构，很痛苦。

set foreign\_key\_checks=0 跳过外键检查。

把有问题的表表空间也删掉了。

3、拷贝生产中confulence库下的所有表的ibd文件拷贝到准备好的环境中

select concat('alter table ',table\_schema,'.'table\_name,' import tablespace;') from information\_schema.tables where table\_schema='confluence' into outfile '/tmp/discad.sql';

4、验证数据

表都可以访问了，数据挽回到了出现问题时刻的状态（2-8）

=====================================================================

额外扩展

drop database world

rm -rf world

drop table city ----> rm -rf frm ibd 元数据

基于Linux文件句柄恢复rm的文件

### 【思路模拟】

#### 【模拟删除库】

##### 【环境搭建】

#新开一个示例

[root@db01 scripts]# mysqld\_safe --defaults-file=/data/3307/my.cnf &

#将3306数据库中的world库复制到3307数据目录，并修改属主和属组

[root@db01 data]# cp -a /application/mysql/data/world /data/3307/data/

[root@db01 data]# chown -R mysql.mysql /data/3307/data/\*

#测试从3306中复制到3307的数据库world可用性

#登录mysql界面

[root@db01 ~]# mysql -S /data/3307/mysql.sock

#复制的数据库world不能使用，

因为表数据的正常使用必须要具备如下三张表，而上述复制过程中的基表没有复制到3307.

t1.ibd ==t1表的数据和索引。---》oracle中有工具可以直接读取ibd文件（得花钱），但是mysql中没有这样的工具

t1.frm ==t1表的列相关信息

基表 ==其他的属性信息----》A主机中的数据库中的基表是不能拷贝到B主机中的数据库中，并使用的 ---所以从A主机中拷贝的数据库是不能在B中使用的

ymq [world]>show tables;

+-----------------+

| Tables\_in\_world |

+-----------------+

| city |

| country |

| countrylanguage |

+-----------------+

3 rows in set (0.00 sec)

ymq [world]>

ymq [world]>

ymq [world]>

ymq [world]>

ymq [world]>select \* from city;

ERROR 1146 (42S02): Table 'world.city' doesn't exist

ymq [world]>

ymq [world]>desc city

-> ;

ERROR 1146 (42S02): Table 'world.city' doesn't exist

ymq [world]>

##### 【解决办法】

###### desc

0.表只有在表空间文件idbdata和元数据同时存在的情形下，才能正常使用表。

1.表空间文件ibddata可以通过alter table confluence.t1 discard tablespace或者

alter table confluence.t1 import tablespace按需求随意导入导出

2.建表的时候，元数据在基表中生成，并且元数据只能通过DDL数据才能产生

---所以

---理论上，新创建基表，基表中的元数据导入到表中，再导入保留有原始数据的idbdata文件，最后表可以使用

##### 【操作flow】

###### #将3307下world中的表全部删除

[root@db01 world]# ll

total 1188

-rw-rw---- 1 mysql mysql 819200 Nov 20 21:33 city.ibd

-rw-rw---- 1 mysql mysql 163840 Nov 18 23:39 country.ibd

-rw-rw---- 1 mysql mysql 229376 Nov 18 23:39 countrylanguage.ibd

-rw-rw---- 1 mysql mysql 61 Nov 18 23:39 db.opt

[root@db01 world]# rm -f ./\*

[root@db01 world]# ll

total 0

###### 【按照原来的列属性创建表】

ymq [world]>CREATE TABLE `city\_new` (

-> `ID` int(11) NOT NULL AUTO\_INCREMENT,

-> `Name` char(35) NOT NULL DEFAULT '',

-> `CountryCode` char(3) NOT NULL DEFAULT '',

-> `District` char(20) NOT NULL DEFAULT '',

-> `Population` int(11) NOT NULL DEFAULT '0',

-> PRIMARY KEY (`ID`),

-> KEY `CountryCode` (`CountryCode`),

-> KEY `idx\_popu` (`Population`)

-> ) ENGINE=InnoDB AUTO\_INCREMENT=4080 DEFAULT CHARSET=latin1;

Query OK, 0 rows affected (0.02 sec)

ymq [world]>show tables;

+-----------------+

| Tables\_in\_world |

+-----------------+

| city\_new |

+-----------------+

1 row in set (0.00 sec)

ymq [world]>select \* from city\_new;

Empty set (0.00 sec)

root@db01 world]# ll

total 204

-rw-rw---- 1 mysql mysql 8710 Nov 20 23:59 city\_new.frm

-rw-rw---- 1 mysql mysql 131072 Nov 20 23:59 city\_new.ibd

###### 【删除空的ibd文件】

ymq [world]>alter table city\_new discard tablespace;

Query OK, 0 rows affected (0.00 sec)

#查看，发现city\_new.ibd已经被删除

[root@db01 world]# ll

total 12

-rw-rw---- 1 mysql mysql 8710 Nov 20 23:59 city\_new.frm

###### 【将原始的数据文件拷贝到3307world库下的目录】

[root@db01 world]# cp /application/mysql/data/world/city.ibd /data/3307/data/

[root@db01 world]#

[root@db01 world]# ll /data/3307/data/world/

total 812

-rw-rw---- 1 mysql mysql 819200 Nov 20 21:33 city.ibd

-rw-rw---- 1 mysql mysql 8710 Nov 20 23:59 city\_new.frm

[root@db01 world]#

#重命名

[root@db01 world]# mv city.ibd city\_new.ibd

[root@db01 world]# ll /data/3307/data/world/

total 812

-rw-rw---- 1 mysql mysql 8710 Nov 20 23:59 city\_new.frm

-rw-rw---- 1 mysql mysql 819200 Nov 20 21:33 city\_new.ibd

###### 【导入ibd数据文件到表空间】

ymq [world]>alter table city\_new import tablespace;

###### 【测试数据能否使用】

……

| 4079 | Rafah | PSE | Rafah | 92020 |

+------+------------------------------------+-------------+------------------------+------------+

4079 rows in set (0.00 sec)

ymq [world]>

ymq [world]>select \* from city\_new;

#可以正常使用

## 单表删除模拟

### #创建环境

ymq [db]>select \* from t1;

+------+

| id |

+------+

| 1 |

| 2 |

| 30 |

| 5 |

| 6 |

+------+

ymq [oldboy]>show create t1;

CREATE TABLE `t1` (

`id` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

### #删除基表

[root@db01 db]# rm -f /application/mysql/data/db/t1.frm

#退出数据库，发现库已经没有表t1

ymq [(none)]>use db;

Database changed

ymq [db]>show tables;

Empty set (0.00 sec)

### 恢复01：创建新表

ymq [db]>CREATE TABLE `t1\_new` (

-> `id` int(11) DEFAULT NULL

-> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Query OK, 0 rows affected (0.01 sec)

ymq [db]>show tables;

+--------------+

| Tables\_in\_db |

+--------------+

| t1\_new |

+--------------+

1 row in set (0.00 sec)

### 恢复02：清除表空间

ymq [db]>alter table t1\_new discard tablespace;

Query OK, 0 rows affected (0.01 sec)

### 恢复03：将原有的完整数据文件重命名

[root@db01 db]# mv /application/mysql/data/db/t1.ibd /application/mysql/data/db/t1\_new.ibd

### 恢复04：新表导入表空间

ymq [db]>alter table t1\_new import tablespace;

Query OK, 0 rows affected, 1 warning (0.01 sec)

### 确认已经恢复单表

ymq [db]>select \* from t1\_new;

+------+

| id |

+------+

| 1 |

| 2 |

| 30 |

| 5 |

| 6 |

+------+

5 rows in set (0.00 sec)

ymq [db]>

### #总结

关键在于，原有数据未损坏，仅仅是基表（表的列定义信息）损坏，可以通过原有的语句创建新表，然后将数据文件重命名，即可恢复