1.悲观锁

1.定义

悲观锁，正如其名，它指的是对数据被外界（包括本系统当前的其他事务，以及来自外部系统的事务处理）修改持保守态度，因此，在整个数据处理过程中， 将数据处于锁定状态。

悲观锁的实现，往往依靠数据库提供的锁机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了 加锁机制，也无法保证外部系统不会修改数据）。

一旦锁住,其他事务就不能操作锁住的数据,查找也不能

2.使用场景(以MySQL InnoDB为例)

商品goods表中有一个字段status，status为1代表商品未被下单，status为2代表商品已经被下单，那么我们对某个商品下单时必须确保该商品status为1。假设商品的id为1。

2.1如果不采用锁，那么操作方法如下：

```
//1.查询出商品信息
select status from t_goods where id=1;
//2.根据商品信息生成订单
insert into t_orders (id,goods_id) values (null,1);
//3.修改商品status为2
update t_goods set status=2;
```

上面这种场景在高并发访问的情况下很可能会出现问题。

前面已经提到，只有当goods status为1时才能对该商品下单，上面第一步操作中，查询出来的商品status为1。但是当我们执行第三步Update操作的时候，有可能出现其他 人先一步对商品下单把goods status修改为2了，但是我们并不知道数据已经被修改了，这样就可能造成同一个商品被下单2次，使得数据不一致。所以说这种方式是不安全的。

2.2使用悲观锁来实现：

在上面的场景中，商品信息从查询出来到修改，中间有一个处理订单的过程，使用悲观锁的原理就是，当我们在查询出goods信息后就把当前的数据锁定，直到我们修改完毕后再解锁。那么在这个过程中，因为goods被锁定了，就不会出现有第三者来对其进行修改了。

注：要使用悲观锁，我们必须关闭mysql数据库的自动提交属性，因为MySQL默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。

我们可以使用命令设置MySQL为非autocommit模式：

```
set autocommit=0;
```

设置完autocommit后，我们就可以执行我们的正常业务了。具体如下：

2.3开始事务

```
begin;/begin work;/start transaction; (三者选一就可以)
//1.查询出商品信息
select status from t_goods where id=1 for update;
//2.根据商品信息生成订单
insert into t_orders (id,goods_id) values (null,1);
//3.修改商品status为2
update t_goods set status=2;
//4.提交事务
commit;/commit work;
```

注：上面的begin/commit为事务的开始和结束，因为在前一步我们关闭了mysql的autocommit，所以需要手动控制事务的提交，在这里就不细表了。

上面的第一步我们执行了一次查询操作：select status from t_goods where id=1 for update;

与普通查询不一样的是，我们使用了select...for update的方式，这样就通过数据库实现了悲观锁。此时在t_goods表中，id为1的 那条数据就被我们锁定了，其它的事务必须等本次事务提交之后才能执行。这样我们可以保证当前的数据不会被其它事务修改。

注：需要注意的是，在事务中，只有SELECT ... FOR UPDATE 或LOCK IN SHARE MODE 同一笔数据时会等待其它事务结束后才执行，一般SELECT ... 则不受此影响。拿 上面的实例来说，当我执行select status from t_goods where id=1 for update;后。我在另外的事务中如果再次执行select status from t_goods where id=1 for update;则第二个事务会一直等待第一个事务的提交，此时第二个查询处于阻塞的状态，但是如果我是在第二个事务中执行select status from t_goods where id=1;则能正常查询出数据，不会受第一个事务的影响。

3.行锁和表锁-Row Lock与Table Lock

上面我们提到，使用select...for update会把数据给锁住，不过我们需要注意一些锁的级别，MySQL InnoDB默认Row-Level Lock，所以只有「明确」地指定主键，MySQL 才会执行Row lock (只锁住被选取的数据) ，否则MySQL 将会执行Table Lock (将整个数据表单给锁住)。

举例说明：

数据库表t_goods，包括id,status,name三个字段，id为主键，数据库中记录如下;

```
mysql> select * from t_goods;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 |      1 | 道具 |
| 2 |      1 | 装备 |
+----+--------+------+
2 rows in set

mysql>
```

注：为了测试数据库锁，我使用两个console来模拟不同的事务操作，分别用console1(加锁)、console2(加锁后操作)来表示。

例1: (明确指定主键，并且有此数据，row lock)

console1：查询出结果，但是把该条数据锁定了

```
mysql> select * from t_goods where id=1 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 | 1 | 道具 |
+----+--------+------+
1 row in set

mysql>
```

console2：查询被阻塞

```
mysql> select * from t_goods where id=1 for update;
```

console2：如果console1长时间未提交，则会报错

```
mysql> select * from t_goods where id=1 for update;
ERROR 1205 : Lock wait timeout exceeded; try restarting transaction
```

例2: (明确指定主键，若查无此数据，无lock)

console1：查询结果为空

```
mysql> select * from t_goods where id=3 for update;
Empty set
```

console2：查询结果为空，查询无阻塞，说明console1没有对数据执行锁定

```
mysql> select * from t_goods where id=3 for update;
Empty set
```

例3: (无主键，table lock)

console1：查询name=道具 的数据，查询正常

```
mysql> select * from t_goods where name='道具' for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 | 1 | 道具 |
+----+--------+------+
1 row in set

mysql>
```

console2：查询name=装备 的数据，查询阻塞，说明console1把表给锁住了

```
mysql> select * from t_goods where name='装备' for update;
```

console2：若console1长时间未提交，则查询返回为空

```
mysql> select * from t_goods where name='装备' for update;
Query OK, -1 rows affected
```

例4: (主键不明确，table lock)

console1：查询正常

```
mysql> begin;
Query OK, 0 rows affected

mysql> select * from t_goods where id>0 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 |      1 | 道具 |
| 2 |      1 | 装备 |
+----+--------+------+
2 rows in set

mysql>
```

console2：查询被阻塞，说明console1把表给锁住了

```
mysql> select * from t_goods where id>1 for update;
```

例5: (主键不明确，table lock)

console1：

```
mysql> begin;
Query OK, 0 rows affected

mysql> select * from t_goods where id<>1 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 2 | 1 | 装备 |
+----+--------+------+
1 row in set

mysql>
```

console2：查询被阻塞，说明console1把表给锁住了

```
mysql> select * from t_goods where id<>2 for update;
```

console1:提交事务

```
mysql> commit;
Query OK, 0 rows affected
```

console2：console1事务提交后，console2查询结果正常

```
mysql> select * from t_goods where id<>2 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 | 1 | 道具 |
+----+--------+------+
1 row in set

mysql>
```

以上就是关于数据库主键对MySQL锁级别的影响实例，需要注意的是，除了主键外，使用**索引也会影响数据库的锁定级别**

举例：

我们修改t_goods表，给status字段创建一个索引

修改id为2的数据的status为2，此时表中数据为:

```
mysql> select * from t_goods;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 |      1 | 道具 |
| 2 |      2 | 装备 |
+----+--------+------+
2 rows in set

mysql>
```

例6: (明确指定索引，并且有此数据，row lock)

console1:

```
mysql> select * from t_goods where status=1 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 1 | 1 | 道具 |
+----+--------+------+
1 row in set

mysql>
```

console2：查询status=1的数据时阻塞，超时后返回为空，说明数据被console1锁定了

```
mysql> select * from t_goods where status=1 for update;
Query OK, -1 rows affected
```

console2：查询status=2的数据，能正常查询，说明console1只锁住了行，未锁表

```
mysql> select * from t_goods where status=2 for update;
+----+--------+------+
| id | status | name |
+----+--------+------+
| 2 | 2 | 装备 |
+----+--------+------+
1 row in set

mysql>
```

例7: (明确指定索引，若查无此数据，无lock)

console1：查询status=3的数据，返回空数据

```
mysql> select * from t_goods where status=3 for update;
Empty set
```

console2：查询status=3的数据，返回空数据

```
mysql> select * from t_goods where status=3 for update;
Empty set
```

4.总结

悲观锁是在高并发的情况下,防止数据被重复操作.在数据库层实现悲观锁,能真正保证数据访问的排他性.在系统加锁并不能保证其他系统会修改数据.加锁前关闭自动提交

加锁之后:

- 行锁(有主键或索引): 不能加锁之后的行,查也不行,太久没有释放锁会报错.当除了加锁之外的数据还是能访问的.
- 表锁(无主键无索引): 当某一条数据加锁,整个表都不能访问.直到事务提交.

2.乐观锁

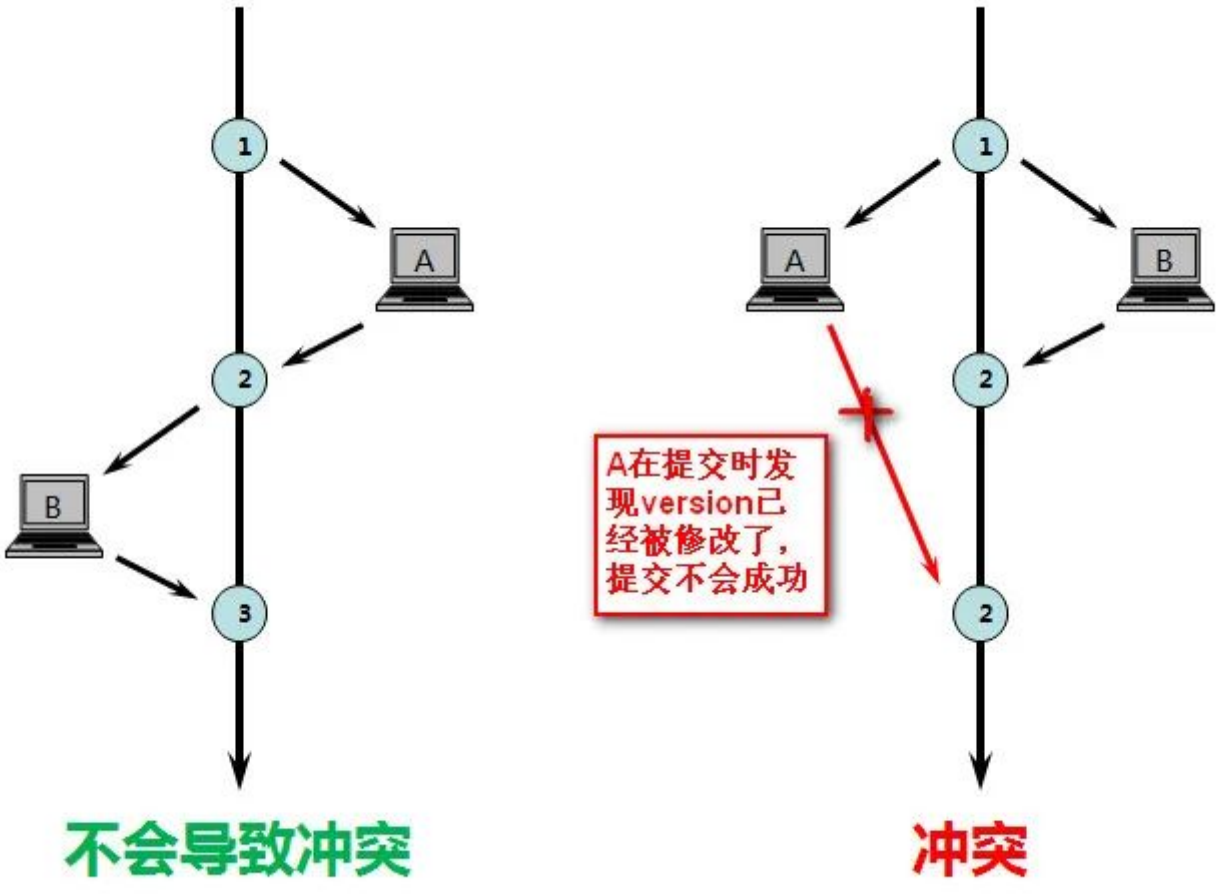
1.定义

乐观锁（ Optimistic Locking ） 相对悲观锁而言，乐观锁假设认为数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则让返回用户错误的信息，让用户决定如何去做

2.实现方式

1.使用数据版本（Version）记录机制

这是乐观锁最常用的一种实现 方式。何谓数据版本？即为数据增加一个版本标识，一般是通过为数据库表增加一个数字类型的 “version” 字段来实现。当读取数据时，将version字段的值一同读出，数据每更新一次，对此version值加一。当我们提交更新的时候，判断数据库表对应记录 的当前版本信息与第一次取出来的version值进行比对，如果数据库表当前版本号与第一次取出来的version值相等，则予以更新，否则认为是过期数 据。用下面的一张图来说明：



如上图所示，如果更新操作顺序执行，则数据的版本（version）依次递增，不会产生冲突。但是如果发生有不同的业务操作对同一版本的数据进行修 改，那么，先提交的操作（图中B）会把数据version更新为2，当A在B之后提交更新时发现数据的version已经被修改了，那么A的更新操作会失 败。

2.乐观锁定的第二种实现方式--时间戳

乐观锁定的第二种实现方式和第一种差不多，同样是在需要乐观锁控制的table中增加一个字段，名称无所谓，字段类型使用时间戳 （timestamp），和上面的version类似，也是在更新提交的时候检查当前数据库中数据的时间戳和自己更新前取到的时间戳进行对比，如果一致则OK，否则就是版本冲突。

3.使用举例

商品goods表中有一个字段status，status为1代表商品未被下单，status为2代表商品已经被下单，那么我们对某个商品下单时必须确保该商品status为1。假设商品的id为1。

下单操作包括3步骤：

1.查询出商品信息

```
select (status,status,version) from t_goods where id=#{id}
```

2.根据商品信息生成订单

3.修改商品status为2

```
update t_goods
set status=2,version=version+1
where id=#{id} and version=#{version};
```

那么为了使用乐观锁，我们首先修改t_goods表，增加一个version字段，数据默认version值为1。

t_goods表初始数据如下：

```
mysql> select * from t_goods;
+----+--------+------+---------+
| id | status | name | version |
+----+--------+------+---------+
|  1 | 1 | 道具 | 1 |
| 2 |      2 | 装备 |       2 |
+----+--------+------+---------+
2 rows in set

mysql>
```

对于乐观锁的实现，我使用MyBatis来进行实践，具体如下：

Goods实体类：

```
/**
 * ClassName: Goods <br/>
 * Function: 商品实体. <br/>
 * date: 2013-5-8 上午09:16:19 <br/>
 * @author chenzhou1025@126.com
 */
public class Goods implements Serializable {

    /**
     * serialVersionUID:序列化ID.
     */
    private static final long serialVersionUID = 68037919081488805871;

    /**
     * id:主键id.
     */
    private int id;

    /**
     * status:商品状态：1未下单、2已下单.
     */
    private int status;

    /**
     * name:商品名称.
     */
    private String name;

    /**
     * version:商品数据版本号.
     */
    private int version;

    @Override
    public String toString(){
        return "good id:"+id+",goods status:"+status+",goods name:"+name+",goods
version:"+version;
    }
```

```
    //setter and getter

}
```

GoodsDao

mapper.xml

```
<update id="updateGoodsUseCAS" parameterType="Goods">
    <![CDATA[
        update t_goods
        set status=#{status},name=#{name},version=version+1
        where id=#{id} and version=#{version}
    ]]>
</update>
```

GoodsDaoTest测试类

```
@Test
public void goodsDaoTest(){
    int goodsId = 1;
    //根据相同的id查询出商品信息，赋给2个对象
    Goods goods1 = this.goodsDao.getGoodsById(goodsId);
    Goods goods2 = this.goodsDao.getGoodsById(goodsId);

    //打印当前商品信息
    System.out.println(goods1);
    System.out.println(goods2);

    //更新商品信息1
    goods1.setStatus(2);//修改status为2
    int updateResult1 = this.goodsDao.updateGoodsUseCAS(goods1);
    System.out.println("修改商品信息1"+(updateResult1==1?"成功":"失败"));

    //更新商品信息2
    goods1.setStatus(2);//修改status为2
    int updateResult2 = this.goodsDao.updateGoodsUseCAS(goods1);
    System.out.println("修改商品信息2"+(updateResult2==1?"成功":"失败"));
}
```

输出结果：

good id:1,goods status:1,goods name:道具,goods version:1

good id:1,goods status:1,goods name:道具,goods version:1

修改商品信息1成功

修改商品信息2失败

主要在于---version=#{version}

3.乐观锁与悲观锁区别

悲观锁: 费时间，用于多线程时效率低,用户要等待直到锁释放.

乐观锁: 用户不需要等待着获取锁。

选择:

追求响应速度,大量进行读取数据操作的时候，用乐观锁是做好的选择.

在向[数据库中写入数据的时候，如果冲突较多建议使用悲观锁，这样在数据库层面进行数据比较，不至于在业务代码之间来回检查。

4.存储引擎 MyISAM和InnoDB

MyISAM : 米森亩

1、MyISAM和InnoDB的区别

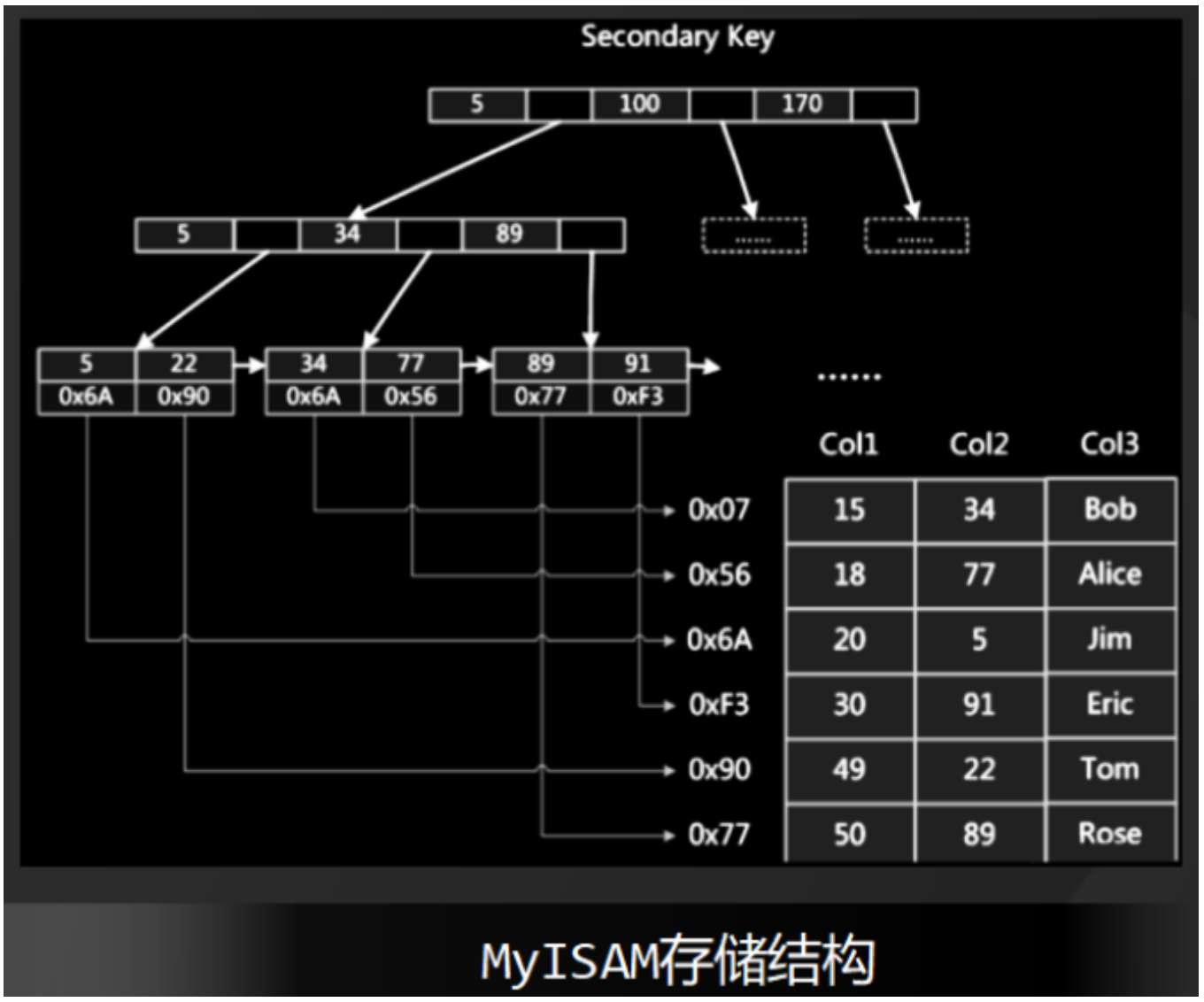
1.数据的存储结构不同

每个MyISAM在磁盘上存储成三个文件，它们以表的名字开头来命名。

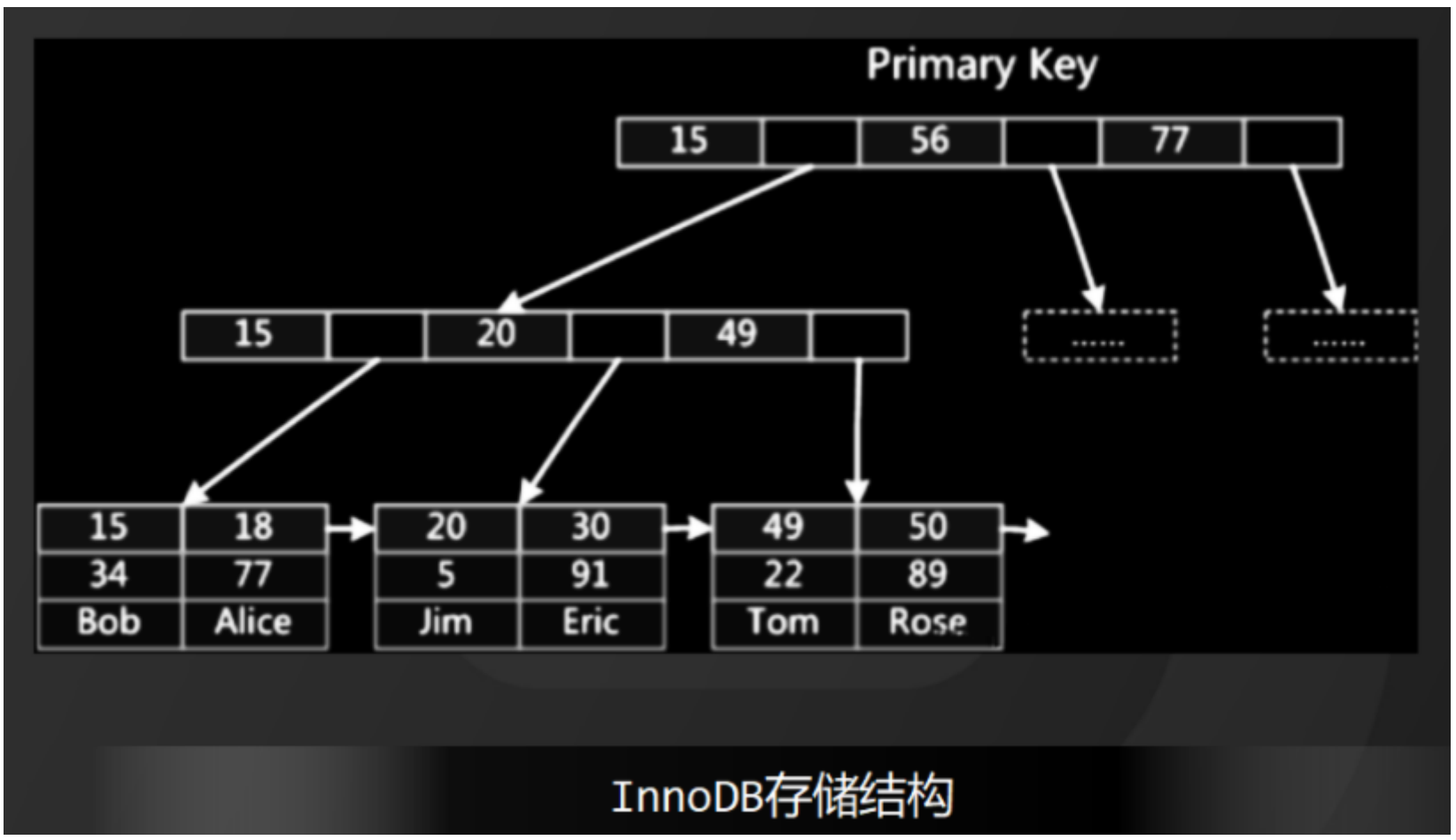
1. .frm文件存储表定义。
2. .MYD(MYD)存储数据文件。
3. .MYI(MYIndex)存储索引文件。

InnoDB在磁盘上保存为两个文件。.frm文件同样存储为表结构文件，.ibd文件存储的是数据和索引文件。

由于MyISAM的索引和数据是分开存储的，因此索引查找的时候，MyISAM的叶子节点存储的是数据所在的地址，而不是数据。



而InnoDB叶子节点存储的是整个数据行所有的数据。



2.存储空间的消耗不同

MyISAM可被压缩，存储空间较小。支持三种不同的存储格式：静态表（默认，但是注意数据末尾不能有空格，会被去掉）、动态表、压缩表。

InnoDB需要更多的内存和存储，它会在主内存中建立其专用的缓冲池用于高速缓冲数据和索引。InnoDB所在的表都保存在同一个数据文件中（也可能是多个文件，或者是独立的表空间），InnoDB表的大小只受限于操作系统文件的大小，一般为2GB。

3. 对事务的支持不同

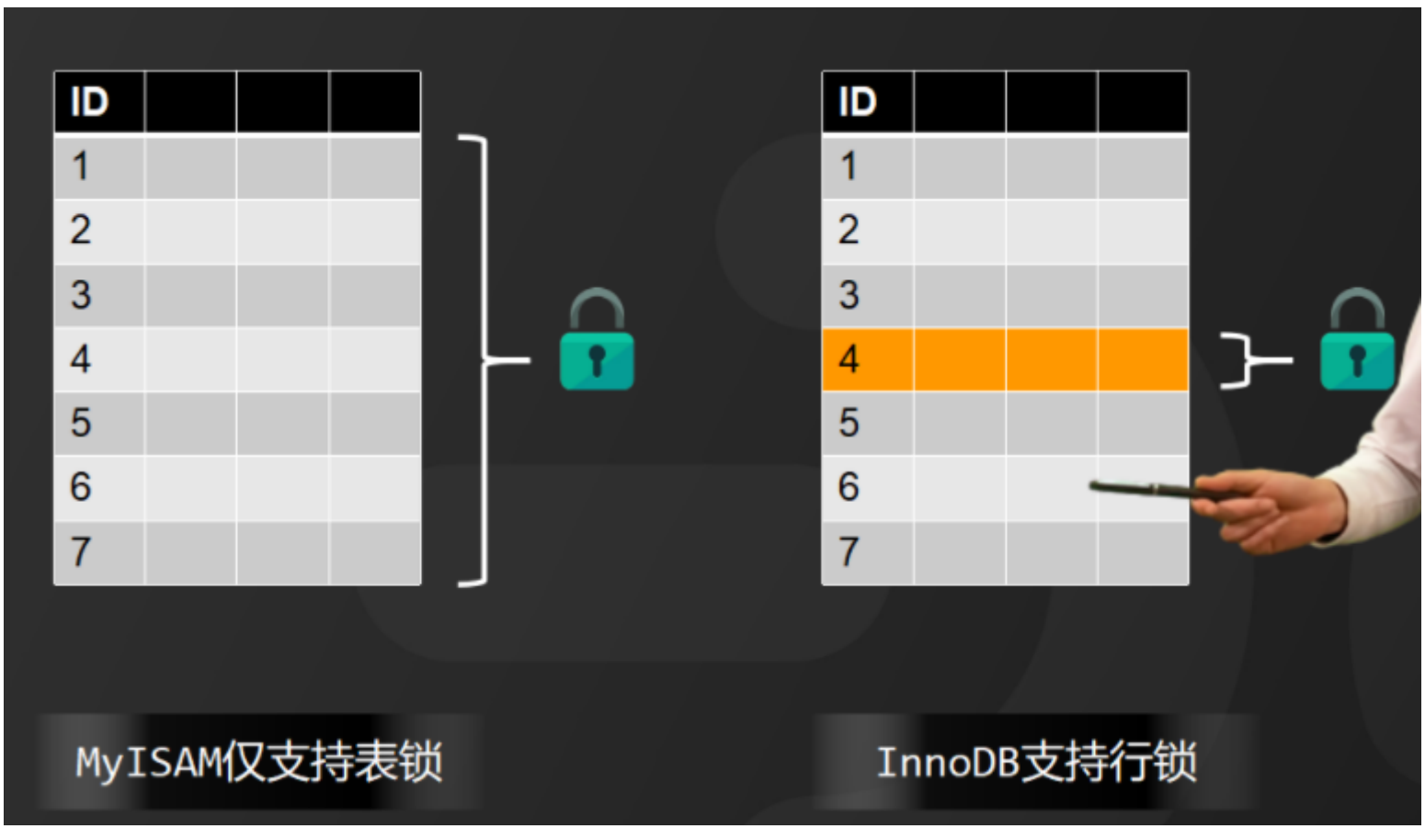
MyISAM强调的是性能，每次查询具有原子性，其执行速度比Innodb类型更快，但是不提供事务支持。

InnoDB除了提供事务支持和外部键等高级数据库功能。还具有事务提交（commit）、回滚（rollback）和崩溃修复能力等这些事务安全型表。

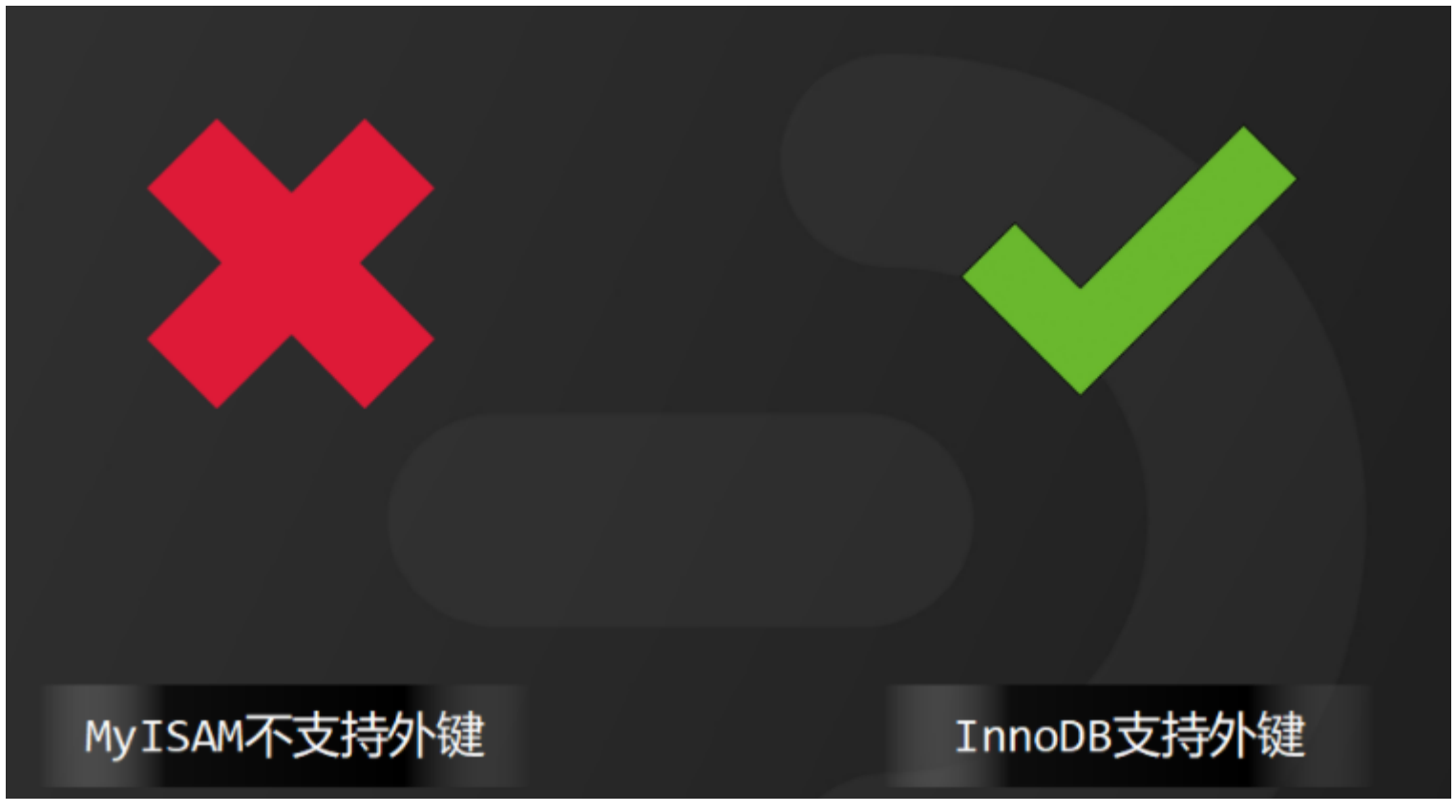
4. 对锁的支持不同

如果只是执行大量的查询, MyISAM是更好的选择。MyISAM在增删的时候需要锁定整个表格，效率会低一些。

而innoDB支持行级锁，删除插入的时候只需要锁定操作行就行。如果有大量的插入、修改删除操作，使用InnoDB性能能会更高。



5. 对外键的支持不同



2.引擎选择

1、如果需要支持事务，选择InnoDB，不需要事务则选择MyISAM。

2、如果大部分表操作都是查询，选择MyISAM，有写又有读选InnoDB。

3、如果系统崩溃导致数据难以恢复，且成本高，不要选择MyISAM。

SQL优化

1.对查询进行优化，应***尽量避免全表扫描***，首先应考虑在 where 及 order by 涉及的列上建立索引。

2.应**尽量避免在 where 子句中对字段进行 null 值判断**，否则将导致引擎放弃使用索引而进行全表扫描

3.应**尽量避免在 where 子句中使用!=或<>操作符**，否则将引擎放弃使用索引而进行全表扫描。

4.应**尽量避免在 where 子句中使用 or 来连接条件**，否则将导致引擎放弃使用索引而进行全表扫描，如：

```
select id from t where num=10 or num=20
可以这样查询：
select id from t where num=10
union all
select id from t where num=20
```

5.in 和 not in 也要慎用，否则会导致全表扫描，如：

```
select id from t where num in(1,2,3)
**对于连续的数值，能用 between 就不要用 in 了**：
select id from t where num between 1 and 3
```

6.模糊查询(开始字段)也将导致全表扫描：

```
select id from t where name like '%abc%'
可以
select id from t where name like 'abc%'
```

7.**不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算**，否则系统将可能无法正确使用索引。

```
select id from t where num/2=100
改为
select id from t where num=100*2
```

11.不要写一些没有意义的查询，如需要生成一个空表结构：

12.很多时候**用 exists 代替 in** 是一个好的选择：

```
select num from a where num in(select num from b)
替换
select num from a where exists(select 1 from b where num=a.num)
```

13.并不是所有索引对查询都有效，SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。

14.索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。**一个表的索引数最好不要超过6个**，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

15.**尽量使用数字型字段，**若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

16.**尽可能的使用 varchar 代替 char** ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

17.任何地方都不要使用 select * from t ，用具体的字段列表代替“*”，不要返回用不到的任何字段。

18.避免频繁创建和删除临时表，以减少系统表资源的消耗。

22.尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

20.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

25.尽量避免大事务操作，提高系统并发能力。

26.尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

MySQL 事务的四大特性

：原子性、一致性、隔离性、持久性，这四个特性简称 ACID 特性

一、原子性（Atomicity ）：一个事务是一个不可再分割的整体，要么全部成功，要么全部失败

二、一致性（Consistency ）：一个事务可以让数据从一种一致状态切换到另一种一致性状态

三、隔离性（Isolution ）：一个事务不受其他事务的影响，并且多个事务彼此隔离

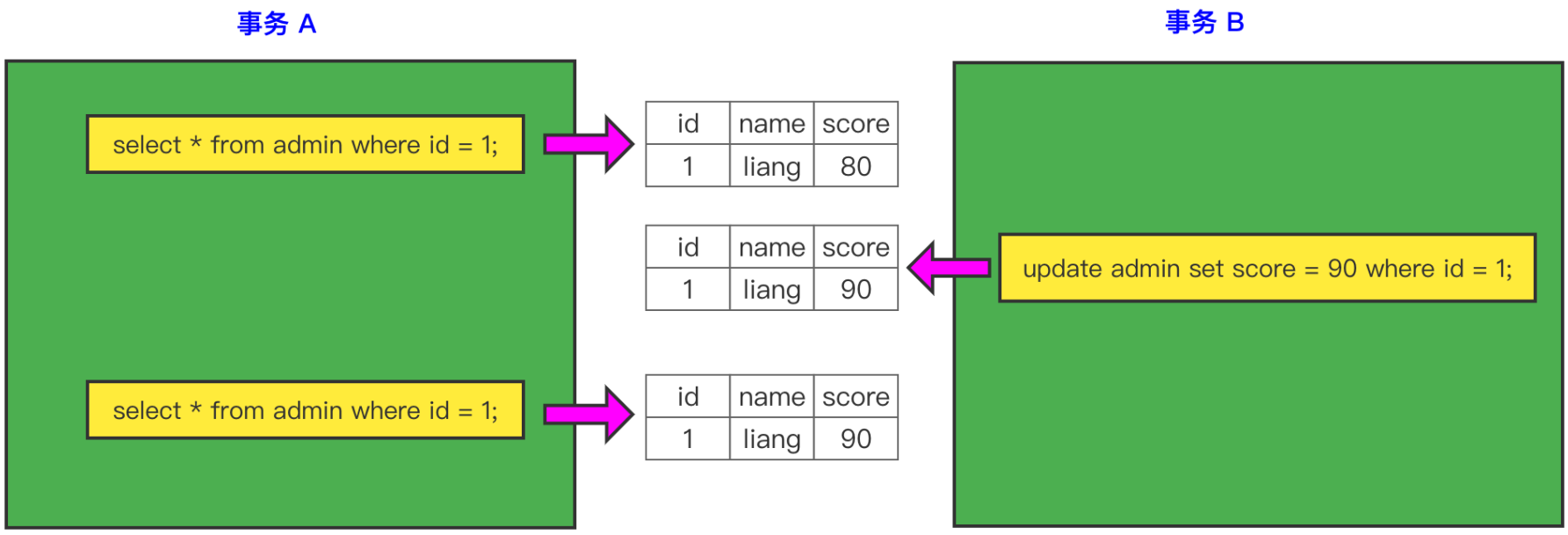
四、持久性（Durability ）：一个事务一旦被提交，在数据库中的改变就是永久的，提交后就不能再回滚

MySQL 事务的并发问题

有多个任务时，应当让多个事务同时执行，这就是事务的并发。既然事务存在并发执行，那必然存在两个事务操作同一个数据的冲突问题.

脏读、不可重复读、幻读时会涉及到事务隔离级别

一、脏读



测试脏读: 将事务隔离级别修改为读未提交

```
# READ-UNCOMMITTED 读未提交
set session transaction isolation level read uncommitted;
```

脏读: 在一个事务里面，由于其他事务修改了数据并且没有提交，而导致前后两次读取的数据不一致的情况，这种事务并发问题称之为 “脏读”

二、不可重复读

事务A读取了数据Data后,就去做了一些其他事(事务A还未结束),但此时,另一个事务,另一个事务B去修改了Data,并提交了,事务A再次读取时Data时,Data已经改变

```
# READ-COMMITTED 读已提交
set session transaction isolation level read committed;
```

不可重复读: 一个事务读取到其他事务已提交的数据导致前后两次读取数据不一样的情况

三、幻读

```
# REPEATABLE-READ 可重复读（默认的事务隔离级别）
set session transaction isolation level repeatable read;
```

幻读: 一个事务前后两次读取的数据不一致，是因为其他事务**插入**数据导致的事务并发情况

MySQL 事务的隔离级别

隔离级别	脏读	不可重复读	幻读
Read uncommitted（读未提交）	是	是	是
Read committed（读已提交）	否	是	是
Repeatable read（可重复读，默认的隔离级别）	否	否	是
Serializable（可串行化）	否	否	否

2.面试对话

1.Java面试发挥一塌糊涂也不要急

你好，你简单做一个自我介绍嘞。
啊，面试官你好啊，我来自湖南湘潭，今年23岁，对软件编程有着浓厚的兴趣，本科毕业两年后从事java后台的开发，在两年工作期间，一共参与参与开发了四个医疗方面的项目。技术方面，熟练掌握后端的一系列基数，能够流畅的进行项目开发，在空余时间会继续学习新技术来丰富自己的知识，提高自己的力。我的介绍就到这里，希望能望加入贵公司.
你最近做的是哪个项目啊？
最近做的是医院信息管理项目简单介绍一下这个项目来。
这个项目是我们公司为一家民营企业开发的一个医院信息管理系统是基于Dubbo的一个微服务项目,为医院的各个工作环节提高高效服务。我主要负责的是药品的出入库，还有门诊医生工作站对一些病人资料的获取一些情况等等，还有对护士站的祝医生服务的一些功能的事项。把我们数据库用的是哪所前端mvc框架使用用后端mvc使用使用十分感VC啊，后端主要的肌肤是使用gpsmm权限制这方面我们使用的语音搜索对对。对全对用户的权限进行呃判判断，呃，采用拆除，他们就对系统和之间的交互进行一些结构，还有使用的软体词作为放放假，减轻处理后的方。
那只是一个简单介绍，嗯，你说你是用Dubbo做的一个微服务，一个项目是吧？
是的，

你简单跟我说一下你你这里面都用到为辅道做的哪些组件呢？
呃，组建一个呃，组建我们用的呃呃呃做的东西，是新开普，然后我们具体用量是现在哦，我不是太记得了哦。
嗯，那你这个应该应该不是一个v服项目吧，你这个是不是就是一个普通的一个分布式项目啊？因为现在的v服项目不都是有几个重要的组件啊，什么重色中心啊什么吊用啊，什么熔断啊，那些东西的嘛。
哦，那这方面是我的错，错了啊，是吧不是的啊okok
那顺便问到这儿，我问一下你，你有没有对SpringCloud有个了解啊哦？
有过简单的了解。
你简单说一下嘞，然后十分苛勒的是斯普瑞在基于斯分获得的这个开发基础上开发的银行一款而分不是呃分布式呃框架呃他呃常用的注册中心有优乐卡还有呃呃，看法也可以使用。然后还有一总办功能，还有主办气，
嗯嗯，你的这个项目总共分成多少个项目啊？就是你这个整个大木项目大的项目分成多少个项目啊，就是分成多少个模块儿进行一个独立的开发的呀？
主要分成七个大模块，然后下面还有一些在细分，嗯，这个叫有个叫药药品物流是你做的是吧？你简单介绍一下这个模块的一个业务了。
呃，这个业务是饿我们away这个一医院的一些进货，还有一些出货的一个记录，还有一些他这个是否一些要给人家过期还有一些啊？呃，我们嗯这个服务药品占的话有很多，分分了，不要分了五个小芙会里面具体有一些药品，还有一些固定资产的一些管。
嗯。你这儿说的是药品物流啊，是不是我理解错了，药品物流是不是指运送运送物品的呀晕？运送药品的呀，还是不是啊哦？
对象物品的一些订购，还有一些采购，还有退货等等，呃呃，你里面有没有出现那种？
嗯，是，可是对外销售的吗？
不对外？

嗯嗯，有没有可能同时有有两个人来操作？同一个药品啊，然后到时候比如说是第一个药品，他们都只有两个的时候，然后一个人来减掉两个，另一个人再来捡两个有正常点吗？那你简单跟我说一下，你是怎么解决副科总问题的呢？呃，我们用用了所对她所当有一一个用户金钱对他呃呃，数量削减寿命会来锁住，然后就是那个数量减掉标识后，我们才会把那个给吃坏你用的是什么锁啊，北关锁是吧，怎么实现的呢？呃，废话所呢呃是基于数据库，然后呃，我们呃访问对他操作之后，然后就把他锁住，嗯，然后访问呃。操作完之后，我们就把他释放。哈，你们这个项目做完了没有？现在，呃，当时我离职的时候还有一点点没有完成，做了多久啊？哦哦，做到额，从3月到7月就差不多五六个月这样住六个月是吧？你们这个项目是多少人所做的呀？哦，20人左右，20人，然后做了3月到7月，大概四四个月左右是吧？哎，你简单说一下你们这个项目的做做项目的一个人员组成的。哦，没有清楚嗯嗯，就是参与这个项目的一个人员组成人员组成啊嗯呃dbaddb工程师额项项目经理下面，然后db工程师还有些运维，还有就是这些海报工程嗯。总共有20个人吧，呃，20人左右，后面又后面，后面又调走了一些人。啊，那做后台的有多少人呀？后后来就是十几个人，我们交所的就行，后台后台做后台的有几个人啊，后台的话主要只有十个左右，十个左右是吧？那你大概跟我说一下，你认识比较熟悉的其中的三个人，他都分别做哪些模块？哦，他们比比较熟的一个，一个是他是做登录的，登录界面的一些基本功，还有一些就是哦。还有要和我一起做户口，要物流管理的一起合分的工作，嗯嗯嗯嗯嗯嗯，在做这个项目的过程中有没有遇到啥比较棘手的问题啊？哦哟，给他说一下嘞，然后因为我们用那个专辑做一个话筒哈对一些热点数据做起划算了然后我拿着我有些数据呢，然后当时设计的它过的时间把它都是默认的，然后他有时候同时过期的，然后就全部访问出去了。嗯，你逃过挨艾斯的这个过期时间是默认的，是吧？嗯嗯，好的嗯嗯。哦，简单啊，这项目我就了解这么多啊，我简单的问你一下，关于技术上的东西吧。你好，你在做那个项目的过程中有没有用到过县城啊？哦，用的比较少，我负责的功能的嘛？用的比较少，是用过还是没用过啊没有用过是吧？okok，你简单跟我说一下，是不是都用到了哪些设计模式啊？呃，十分钟呃呃呃，一个模板模式，对一些重复的代码进行呃，对一些重复的代码解决。还有爱情模式啊！工啊，是工工厂厂还是兵工厂还是？嗯。还有还有我得通他大嗯，好，我看那个哎，你说吧嗯嗯，它的前端的一个模板方哦磨磨合社服帮助提供了一些对JSP的一些呃标签。嗯。嗯，你用的比较多的是买白鞋子还是那个GPA啊，聊几个GPA吗？你简单说一下，他们两个的一个联系和区别呢？哦，是吗？被里头的GPA的区。呃GPA是hell呃呃呃呃呃呃hell的一个呃规范哦她嗯，它集成了呃hellcat一些嗯，对，他对他们冷啊！提供了一些接口，会不会让我们呃使用呃？自动生成一些简单的回一句。哦，卖片子是一个半自动的OL盟框架，然后他被灵活的使用搜狗与灵活的使使用搜狗与就进行对啊，不同的业务指导使用。呃他萤火的程度很高，但是被搜狗一帜的风景比较要求比较高。然后卖贝尼斯对迅速的移植性比较差，呃呃呃，他们很好，他们的移植性很好。嗯，我看你几个项目都用到死门部的是吧？有没有了解过那个守门部的一个自动配置原理啊？哦，是不是不能是工科的原理？你有了解过吗？哦，有点傻不起来嗯，好嘛，没关系啊，那你就简单跟我说一下，你对你用死不认，不同的过程中，你的一些体会吧！嗯，非常的方便，它大大节省我们创建和启动项目的时间，而且对对方价的整合性很好，嗯，你的项目是怎么不说的呀？我们是先在本本机起启动，然后再不迟到红旗厂？啊，有没有做过？有没有做过那个多克的部署啊？没有做过是吧？Ookok，有没有用过那个东营的小姐动力吗？简简单简单说一下你在哪用的呢？用的是哪个产品？呃，我们在嗯这个嗯，因为是一管理系统中的来帮我们呃。和挂号的时候，如果人妖比较多，我们会用石头被教师格列的行行行，还有系统那些交流的时候，我们会用这个还没过来性的解我。嗯嗯嗯。用在哪种场景用来集合啊？他就是当我们一个系统要调用另外一个系统的功能的时候，他会用它来解解我。嗯，那我不能直接调吗？为啥要用他家偶嘞光直接调的话是可以？但是这个性比较高，而且我们还要等待这个。呃，将对系统的系统的可维护性不同，有用过那个一些呃，你们的图，你们的图片是怎么存储的吧？我们用了一个分布式的图图文件系统。叫什么名字啊？呃，叫八子的敌人白子是吧啊你是你们是怎么搭建的呢？嗯，我们部署在服务器上，然后屙在屙屙在背景中对她进行眼睛，你写你写配置。你那我们引入了这个法式的第200次以后我们的整个项目的几个是怎么样的呀？就是我们是怎么操作法死跌白死的？引入了以后呃，储存在这个文件系统上，然后我们有，然后把那个他会有一个普遍地址，然后我们保存在数据库上，然后我们当然要展现的时候就好在页面展现出来。嗯，我说的意思就是说现在我引入了一个法式的颠簸时，以后比如说要上传一个图片，会经历哪些步骤，会经历哪些福气啊！嗯。这方面具体的流程我没有去了解过啊，你的模块没涉及到吗？哦，有，但是我好像只了解用还没有了解什么，私下okok有用，你项目中有用到缓存吗？哎呦，你简单说一下呗。哦，我们首先向我启动时候会将呃需要展现的数据先查一遍，然后转到那几次当中，哦，这样就大大减少了用户的体验感，呃，操作的时候就可以直接展现出来，不用去多访问数据库嘛？用的是啥啥是用的是什么的什么的缓存技术啊瑞ID 4是吧？你操作员ID 4是怎么用的呀？哦，用的是加加我的那个，呃呃，客户叫kt 4。绝对是是吧，有没有有没有在死不认输死不死当中进行集成用的啥呀？嗯，骄傲不是太记得了，那个胶是不是太记得了哈哈哈哈？没关系啊嗯。额，管理这个技术这方面啊，我就了解这么多，嗯嗯，这个技术就面试完了我。给你几个建议吧，嗯嗯嗯，第一个啊，我感觉你的知识体系当中好像去拿了那个真正的一个微服架构啊，然后你对微幅佳构的理解可能是有问题的啊！你你那个大伯做的，他其实是一个坟墓是的啊，微幅的价格是它要把它拆成每就是把整个系统啊把它拆成很多很多的小幅，每个服务都是一个团队，然后进行独立的技术选型独立的部署啊，独立开发，独立部署的呀和这不太一样啊，和你刚开始的那个。哦哦，知道了，第二个是啥呢？现在外面，现在外面啊，大部分起义当中，传统的那种部署已经用的不多啦，你可以以后玩那个啥嘞，就是多克啊，以多克的编排技术啊，像k8s那方面去，自己自学一点嘛。哦，至于多少容器的多数对其多多克容器的部署，虽虽然外面他都有对应的运维来做嘛，但是对我们开发人员最基本的多克呀，然后k8s这些还是需要有所了解嘛，才能增长一点啊然后的话，嗯，其他的刚才。嗯嗯，你说的是你是不是本来是不是就没有用过县城啊，你有没有想过我们为啥没用过县城啊？不是这样的啊，其实我们天天都在用现成啊，你想吧，就是我们从前台发一个请求到我们的死不的里面是不是不同的底层不就是？不就是挨他们开的吗？那其实很简单呀，一个请求来了以后，到时候他是不是都会通过他们开的创建一个县城？是不是来劲处理啊？哦，也就是说她他是封装好了的，我们是做假网，web开发不需要我们自己去控制县城嘛，是吧嗯。嗯。然后

你对那个lidc操作，刚才你可能记不起来了，现在有一种操作，VIP斯最简便的方法有个叫斯芬斯塔瑞帝斯啊！啊你你有没有了解过锁门推塔呀贝塔极品雷塔？极品斯比尔斯塔瑞狄斯斯斯斯斯斯斯斯斯斯斯斯斯斯斯斯一个产品啊，这个心理现在是用的很多的呀。哦哦。没关系啊，这个啊，我们今天以前那也是做过交流嘛，哎。呃，大概就跟你说这么多，你还有什么想了解的没有哎，你说嘛啊！回答的好烂啊，觉得自己还还好啦还好啦呃这个呃呃呃，这个东西以后以后东面几家多练习一下嘛哎。那没有其它问题的话，我们就到这儿吧。哎哈哈哈嗯嗯，各位线上的同学或者是线上的网友啊，呃呃，我作为一个还是有十年左右，在it打听的一个it人吧，给大家一点建议啊！第一个呢就是咱们现在学习it，尤其是做JAVA后台啊，一定要把那个为服务套给搞懂了，包括了现在呃呃，或者说是前几年用的比较熟的那个斯普瑞cloud的那一套，一定要掌握了。当然，现在来慢慢的把国内阿里巴巴嘛，你出了一个斯文克劳德斯斯文克劳德的知识啊，现在是逐渐流行起来的啊，所以我们对这个斯本人对埃斯本家族包括了传统的奈特弗莱克斯以及啥呢？现在可能将来要火的叫做埃斯特克劳德里巴巴呀，我们对对他的工具一定要好好的学习一下这样的话。哎，无论是你换工作啊，还是啥嘞，你还去找工作啊，对你都是有好处的，就是维护这套架构体系啊，我们一定要先去学习。第二个是啥呢？现在外面企业啊，大部分都是用死不认不出来价格的，所以我们对死不死不来哎哎哎哎不不能仅仅局限于哎？怎么用啊？还要知道一下子门部头的，唉，一些原理啊，也就是他的那个自动配置原理啊，我们也需要指导一下。第三个是啥呢现在啊，随着这个软件技术的一个发展啊，哎，对我们开发人员啊，不仅仅是存在要写代码了啊，唉，这个时候呢，我们开发人员也是需要懂一些部署技术的，比如像啥做基本的，你那个命令啊一家。多克尔的不爽，一节多克的编排工具，比如像最简单的到坑他们pose啊一节到k8，s等编排工具啊，我们作为后台研发人员也是需要掌握的啊。这是关于咱们咱们技术方面啊，主流嘛斯特尔克拉克和斯波尔布特嘛，这些是需需要掌握的，然后呢，就是我们平时啊，唉，刚才这个同学在面试的时候来，他说了他们。他们整个团队有20多个人是吧？那我让她把这20多个人介绍出来，那他其实是不是不是就唉，不太好介绍了20个人是不是太多了一节？他刚才说了有十个后台，那我是不是问他十个后台粉笔开发什么？这些我是不是都是检测它有没有真实工作经验的呀？我感觉刚才这个面试者啊，其实是不是？呃，可能没多大真正的工作经验啊，因为一个团队，哪有这么多人嘛。这个是要给大家一些建议的，就是我们平时在面试的时候啊，一定要把你的这个项目的人员组成业绩呢，哪个人负责哪些模块一定要搞清楚？这样的话你才不会。嗯，这样才才啥嘞哎，才会真正的找到工作啊，其实和我们工作中也是一样的啊，我们工作中也是走的是这样一一条路线啊，唉，一个团队的人员组成，每个人干哪个事儿？你肯定是需要走大的啊！嗯。然后的话就是我们在面试的时候呢，一定要把你最近做的项目或者啥嘞，你准备拿去面试的项目啊，一定要把他的业务了解清楚，以及里面用到的一些技术点啊！技术点啊，现在问的比较多的啊，像卧底似的。缓存一一以及zada的缓存里面会涉及到一些啥呢？叫iPad双双写一枝新芽。还有虚呃缓存雪崩啊，缓存穿透啊缓存集团啊，这些是怎么解决的？也是问的比较多的啊，这是关于缓存啊。然后接下来呢？还有一个叫啥嘞？我们分布式文件系统啊，也是用的很多的，尤其是我们引入了文件是文件系统，以后我们项目中是怎么使用的？这些细节啊，咱们都要去想一下啊，当然还有啥嘞。还有我们的那个全文缩影啊，就是全文检索框架像伊拉斯提斯尔帖啊，然后索隆啊，这些东西我们也是需要下去准备的啊。呃，当然，为了高敏发嘛，咱们做那个啥嘞哎，软笔的michael也是必须要用的嘛，比如我们做一个秒杀啊然后呢？我想我只有一个产品，我免费让别人来买嘛，那到时候我让他免费来买的话，那这时候会出现一个情况啊，就是很多人是不是一起来买啊？这个时候呢，其实我们采用一个。才对的一个策略啊，无论你多少人来了，反正我只要第一个啊，唉，这个其实也就是秒杀的功能啊，会用到一些小起动力啊当然详细对你来，不仅作用于对象哎，就像刚才那个同学说的一样，可以挨家处峰值啊然后啥嘞调多个接口的时候。哎，比如我这个接口我这个佛掉掉了十多20个服，那这个时候呢，我们是不是要依赖于十多20个服务即可啊？那这样不太好，我们呢？把我们的消息发到那个消息堆里面去，让其他的人直接依赖这个消息，东西就可以了，他可以借欧啊，等等啊，这些都是很重要的一些知识点啊！好了啊，这是关于对技术上面对大家的一个面试吧，对对大家的一个忠告告吧，或者说对大家的一点建议吧，然后嘞，最后说一下面试心态啊，其实刚才这个面试者啊，他的心态是很好的。哎，他有的东西他知道就知道不知道就不知道有有的同学出去面试啊，不值得一点乱扯一卡，然后好多说出来都是错的啊这样也是不行的啊，第二个来要做到不卑不亢。比如说刚才这个面试者，然后然呢有的她没听清楚她是不是就可以问我？让我再说一遍这些都是非常好的啊，哎，总之一句话吧，就说咱们出去再做面试的时候，我们要有一种心态啊，什么心态呢就是。哎，你是去找一份工作，然后呢？然后公司呢，也是要找一个人来帮她做事是不是啊？那我们就思考一下啊，就说我们去找工作，不是去求他啊，我们是啥嘞？哎，是不是他招聘了我？我为他创造价值了，她只是把一部分价值还给我的，我们作为面试点啊，不要做到什么低三下四的啊，我们都是平等的啊，所以有了这种形态呢？咱们在面试的时候才能真正的做到一个不卑不亢的一个情况啊！哎，这是关于我哎对哎，网上的一些网友的一些建议吧，技术上面该掌握的要掌握好，这是第一个嘛第二个啥嘞？面试心态一定要好然后呢最后唉，就是在面试过程中，靠自己的发挥了嘛，然后就能找到一份好好的工作嘛？啊，我大概就说这么多吧，哎，最后哎，就看这个视频的所有的网友啊啊或者说是所有的网友的网友的同学啊，要找工作了，或者要换工作啦，等等找一份，哎，得到一份高薪的工作啊，好了啊，我就说这么大的这么多啊，谢谢大家啊！