数据库的 事务、锁、数据库优化、索引

数据库与系统设计

事务一般在框架中配置全局事务

事务：

1、特性

2、隔离

索引：

1、优化

2、机制

目录： 1、Hash （工作的时候一般用这种形式存储数据）

key值是有排列，例如查询张三，先对张这个key值查询，查询出value值姓张的所有记录，然后再选择张三。这样子查询速度快！！！

2、B-tree（困难）

什么时候会用上索引呢？

20%

1、经常用 ：查询条件（where）但是,不是每一个用来做查询条件就可以做索引！

1.高维度的列创建索引（维度是数据列中不重复的值出现的个数，这个数量越高维度就越高）；

例如：性别这一列就不能够创建索引，因为这列高纬度低

2.查询条件尽量能快速将千万条数据筛选出 几十条的范围！（缩小查询范围）

3.变更的时候，影响数据的量

2、变更少

3.索引的缺点：

①创建索引和维护索引需要时间成本，这个成本随着数据量的增加而加大

②创建索引和维护索引需要空间成本，每一条索引都要占据数据库的物理存储空间，数据量越大，占用空间也越大（数据表占据的是数据库的数据空间）

索引用法： 顶级域名 能够使用到索引。 Like “%word%”百分号在前面是无法使用索引

分顶级域名与二级域名

如www.baidu.com

先查询baidu 100万条数据 然后www 50万条数据 再然后定位到md5 这一条数据.

utf-8格式 每个段占3个字节

char类型 只能 char(190) //765/4=191

**事务**

数据库事务无非就两种：读取事务（select）、修改事务（update,insert）。在没有事务隔离控制的时候，多个事务在同一时刻对同一数据的操作可能就会影响到最终期望的结果，通常有四种情况：

（1） 两个更新事务同时修改一条数据时，很显然这种情况是最严重的了，程序中无论如何也不能出现这种情况，因为它会造成更新的丢失！

（2） 一个更新事务更新一条数据时，另一个读取事务读取了还没提交的更新，这种情况下会出现读取到脏数据。

（3） 一个读取事务读取一条数据时，另一个更新事务修改了这条数据，这时就会出现不可重现的读取。

（4）一个读取事务读取时，另一个插入事务（注意此处时插入）插入了一条新数据，这样就可能多读出一条数据，出现幻读。

以上四种情况描述完毕，相信大家也发现规律了，前三种是对同一条数据的并发操作，对程序的结果可能产生致命影响，尤其是金融等实时性，准确性要求极高的系统，绝不容许这三中情况的出现，

相比第四种情况不会影响数据的真实性，在很多情况下是允许的，如社交论坛等实时性要求不高的系统！

综上四个情况，我们可以大致这样简单的理解（最初说的两种事务的自由组合2\*2=4）：

A） 修改时允许修改（丢失更新）

B） 修改时允许读取（脏读）

C) 读取时允许修改（不可重复读）

D) 读取时允许插入（幻读）

从上到下问题越来越不严重，但所需的性能开销却越大。因为不同的系统允许不同级别的情况，所以就出现了事务隔离这么一个东东，来允许我们设定数据库的并发行为。

事务具有四个特征：原子性（ Atomicity ）、一致性（ Consistency ）、隔离性（ Isolation ）和持续性（ Durability ）。这四个特性简称为 ACID 特性。

1 .原子性

事务是数据库的逻辑工作单位，事务中包含的各操作要么都做，要么都不做

2 .一致性

事 务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。因此当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。如果数据库系统 运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是 不一致的状态。

3 .隔离性

一个事务的执行不能其它事务干扰。即一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰。

4 .持续性

也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。接下来的其它操作或故障不应该对其执行结果有任何影响。

隔离级别：

### Repeatable Read（可重读）经常用

### Read Committed（读取提交内容）

### Read Uncommitted（读取未提交内容）不建议用

### Serializable（可串行化）

设置事务隔离级别：set session transaction isolation level Read uncommitted;

查询隔离级别：select @@tx\_isolation;

开启事务： start transaction

提交事务： commit

如果没有开启事务则每次insert/update/delete 都会自动提交事务。

乱码问题：

编码流程：

浏览器：请求编码 ↓↑

后端代码：文本编码↓↑

数据库：存储编码 ↑

解决问题： 1、在struts2中action获取前段数据的地方做断点，查看是否有乱码。

2、数据库查询出来的结果是否正确！

乐观锁与悲观锁

锁一定是在事务中。

1、乐观锁：使用版本号处理repeatRead隔离等级 的时候 多个事务同时执行同一条数据的时候。

Update content\_url set content = ‘test1’,version=’1’ where id = 1 and version=0;

2、悲观锁：使用 for update

select \* from content\_url where id = 1 for update;

执行了这个以后，其他事务不能再对该条数据进行操作！（会阻塞其他人操作）

间断锁1<10

行锁:lock in share node;

独占锁：for update;

reodis锁：单线程

并发：减少锁的范围

数据库创建sql语句：

create table if not exists `content\_url` (

`id` bigint(20) unsigned not null primary key auto\_increment,

`content` varchar(10) not null)

engine=Innodb default charset=utf8mb4 collate=utf8mb4\_unicode\_ci;

**系统设计**

是什么？

分析阶段，确定系统的功能要求、逻辑模型，最终确定一个开发方案。

通常包含：

1、技术选型：框架

2、建模 ：（数据库中则是实体，java中是类）

3、数据库设计：（ER图）

4、概要设计（模块划分）：基本功能要求划分；面向对象划分

最后要给出文档内容有，接口返回什么数据，传入什么数据需要确定！

5、详细设计：模块实现（算法），局部结构设计

地理位置的存储：elastic-search 、 redis

原则：

1、结构稳定性

2、可扩展性及可修改性

3、健壮性

4、标准化原则

领域模型

是什么？

领域内的概念类或显示世界中对象的抽象；专注于分析问题领域（即需求）。

更能够完成解耦的动作！

1、从业务描述中提取名词。

2、从提取出来的名词中总结业务实体，区分名词中的属性、实体，形成问题域中操作实体的集合；

3、确定模型之间的关系：数据库ER图；系统设计UML。

对外暴露的接口 用 protected

MVCS

controler控制器 ：

1、接受请求参数

2、校验参数

基于功能需求划分；面向对象模型划分。

主要是为了实现 解耦：降低复杂度，提高系统的可扩展性、健壮性

划分原则

1、职责单一：只处理它职责范围内的事情，并且要完整

2、独立性&封闭性：模块内高内聚，模块间松耦合；开放出来的接口尽可能少，并且简单

3、消除重复：建立公用模块，减少冗余

4、易于了解：职能明确，逻辑清晰

5、命名：动词+名词、需要前后统一！

get(单个)、list(多个)、add(添加)、del(删除)、update(更新)

模块划分：判断操作是否属于同一个模块，不属于则放到其他模块中。

命名：时间+版本号+名字

面试问题：项目中 重点是什么，难点是什么？怎么解决的？

简历中 项目中写解决了什么问题 以及关键字