**数据库**

[**https://github.com/CyC2018/CS-Notes/blob/master/notes/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F%E5%8E%9F%E7%90%86.md**](https://github.com/CyC2018/CS-Notes/blob/master/notes/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F%E5%8E%9F%E7%90%86.md)

[**https://zhuanlan.zhihu.com/p/329865336**](https://zhuanlan.zhihu.com/p/329865336)

简述关系型数据库与菲关系形数据库的区别与联系

**简述数据库的事务**

事务（Transaction）是一个操作序列，不可分割的工作单位，以BEGIN TRANSACTION开始，以ROLLBACK/COMMIT结束

比如，从A账户转账到B账户，整个过程就是一个事务。

原子性（Atomicity）：

逻辑上是不可分割的操作单元，事务的所有操作要么全部提交成功，要么全部失败回滚（用回滚日志实现，反向执行日志中的操作）；

一致性（Consistency）：

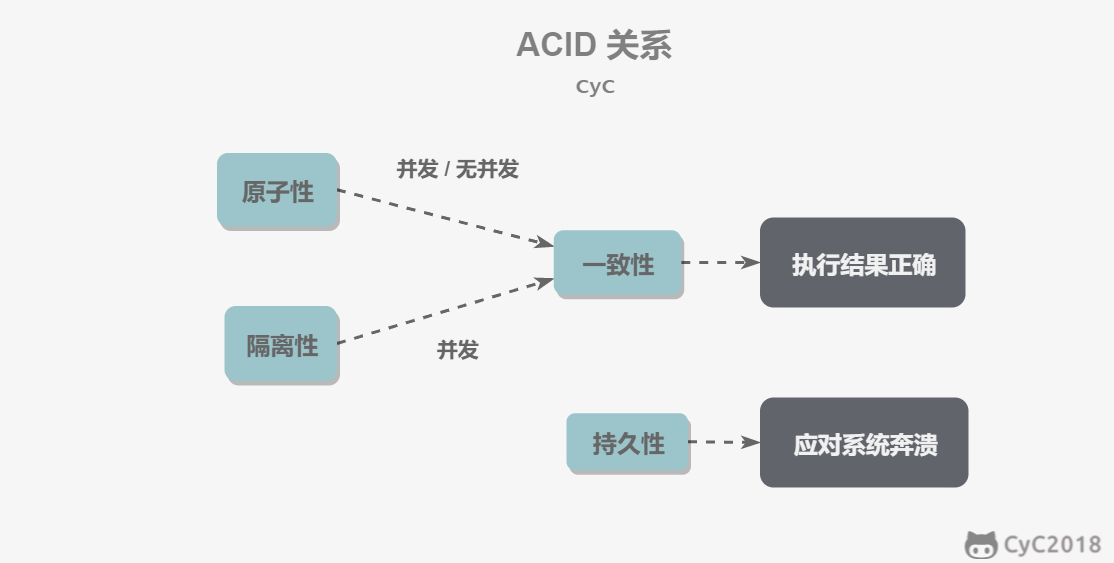
事务的执行必须使数据库保持一致性状态。在一致性状态下，所有事务对一个数据的读取结果都是相同的

隔离性（Isolation）：

一个事务所做的修改在最终提交以前，对其它事务是不可见的（并发执行的事务之间不能相互影响）

持久性（Durability）：

一旦事务提交成功，对数据的修改是永久性的



**会出现哪些并发一致性问题？**

丢失修改：

一个事务对数据进行了修改，在事务提交之前，另一个事务对同一个数据进行了修改，覆盖了之前的修改；

脏读（Dirty Read）：

在不同的事务下，当前事务可以读到另外事务未提交的数据。造成两个事务得到的数据不一致；

不可重复读（Nonrepeatable Read）：

在同一个事务中，某查询操作在一个时间读取某一行数据和之后一个时间读取该行数据，发现数据已经发生修改（针对update操作）；

幻读（Phantom Read）：

当同一查询多次执行时，由于其它事务在这个数据范围内执行了插入操作，会导致每次返回不同的结果集（和不可重复读的区别：针对的是一个数据整体/范围；并且针对insert/delete操作）

**锁的类型**

锁的层次

MySQL提供了两种封锁粒度：行级锁，表级锁

锁定的数据量越少，发生争用的可能性越小，并发程度越高

但，加锁消耗资源，封锁粒度越小，系统开销越大

需要在并发程度和开销之间，做一个权衡

锁的类型

互斥锁，也就是写锁。加锁期间其他事务不能对A 加任何锁

共享锁，也就是读锁。加锁期间只能读取，不能更新。其他事务可以对他加读锁，不能加写锁。

意向锁：表示想要加锁，先试试。

没有意向锁的话，事务对表或行加写锁前，还要先检查一遍有没有被加过，麻烦

意向锁是表的锁

一个事务想要对哪里加读锁时，先要获得表的读意向锁，或更强的锁

想要对哪里加写锁时，先要获得表的写意向锁。

于是，加读锁或写锁时，总要先检查意向锁，就不用再去一个一个检查了。

悲观锁：

认为数据随时会被修改，因此每次读取数据之前都会上锁，防止其它事务读取或修改数据；应用于数据更新比较频繁的场景；

乐观锁：

操作数据时不会上锁，但是更新时会判断在此期间有没有别的事务更新这个数据，若被更新过，则失败重试；适用于读多写少的场景。

乐观锁的实现方式有：

加一个版本号或者时间戳字段，每次数据更新时同时更新这个字段；

先读取想要更新的字段或者所有字段，更新的时候比较一下，只有字段没有变化才进行更新

**封锁协议**

**一级封锁协议**

事务T要修改数据时，必须先加写锁，直到T结束时才释放锁

可以解决丢失修改的问题。因为不能同时有两个事务对同一个数据修改。事务的修改就不会被其他人覆盖

二级封锁协议

在一级的基础上，事务T要读取数据时，必须加读锁，读完之后马上释放。

可以解决脏读问题。因为如果人家在修改，你就不能加读锁了，也就不会读入数据

三级封锁协议

在二级的基础上，读取数据必须加读锁，直到事务结束时才能释放

解决不可重复读的问题。因为在读的时候，其他事务就不能对这条数据加锁，避免读的期间数据发生改变。

**隔离级别**

未提交读 Read uncommitted

事务中的修改，即使没有提交，对其他事务也是可见的

提交读 Read Committed

一个事务只能读取已经提交的事务 所做的修改。即，一个事务所做的修改在提交之前对其他事务是不可见的。

可重复读 Repeatable read

保证在同一个事务中，多次读取同一数据的结果是一样的

可串行化 serializable

强制事务串行执行。则多个事务互不干扰，不会出现并发性一致问题。

需要加锁实现，只有加锁才能保证同一时间只有一个事务执行，才能保证事务串行



**范式**

不符合范式会产生的异常

冗余数据

修改异常： 修改了一个记录中的信息，但另一个记录中相同的信息却没有被修改

删除异常：删除一个东西，丢了不该丢的东西

插入异常：想插插不了

范式

第一范式 1 NF

属性不可分

第二范式 2NF

每个非主属性完全函数依赖于键码

就是一张大表拆成小表

第三范式 3NF

非主属性不传递函数依赖于键码

ER 图

索引

一个排序的数据结构，它可以对数据库表中一列或多列的值进行排序，以协助更加快速的访问数据库表中特定的数据。

就是事先排好序，要查的时候就可以二分了

有索引的话，可以由全表扫描 变为 只扫索引不管数据表有无[**索引**](http://mp.weixin.qq.com/s?__biz=MzI3ODcxMzQzMw==&mid=2247489169&idx=1&sn=5188961c066ccb5f1065e8611a8c2012&chksm=eb5393a7dc241ab1d0268266b9263053a87b4b2d2ea81b86d66d49b393d6d5f1ed09e6857a4e&scene=21#wechat_redirect)，首先在SGA的数据缓冲区中查找所需要的数据，如果数据缓冲区中没有需要的数据时，服务器进程才去读磁盘。

无索引，直接去读表数据存放的磁盘块，读到数据缓冲区中再查找需要的数据。

有索引，先读入索引表，通过[索引](http://mp.weixin.qq.com/s?__biz=MzI3ODcxMzQzMw==&mid=2247489169&idx=1&sn=5188961c066ccb5f1065e8611a8c2012&chksm=eb5393a7dc241ab1d0268266b9263053a87b4b2d2ea81b86d66d49b393d6d5f1ed09e6857a4e&scene=21#wechat_redirect)表直接找到所需数据的物理地址，并把数据读入数据缓冲区中。

B树索引, hash 索引

B树 B+树结构

B 树

插入搜索删除都是logN

B+ 树

N 叉树和有序数组的结合体

层次更低，更扁平，IO次数更少

每次都需要查询到叶子节点，查询性能稳定

叶子节点形成有序链表，范围查询方便

**Redis**

NoSQL非关系型内存key-value 数据库

key 只能为string

value 可以是5种: string，列表，集合，散列表，有序集合

JVM

[**https://github.com/zhengjianglong915/note-of-interview/blob/master/java/jvm.md**](https://github.com/zhengjianglong915/note-of-interview/blob/master/java/jvm.md)