数据库

1. 范式

在面向对象的思想指导下，遵循模式建立的表称为模型（Model），具体的记录称为实体（Entity），表字段称为字段，具体的记录的某个字段称为属性（Property）。

1.1. 1NF

实体的每个属性都是原子属性，不存在多值属性。无重复属性。满足1NF才可以称之为关系型数据库。

> 字段是最小的的单元不可再分

1.2. 2NF

在不存在多值属性的基础上。要求定义主属性。实体的属性们要完全依赖于主属性。不存在非主属性对主属性的部分依赖。

> 满足1NF，表中的字段必须完全依赖于全部主键而非部分主键（一般我们都会做到）

1.3. 3NF

在不存在多值属性、不存在实体的属性部分依赖于主属性的基础上。要求任何非主属性不依赖于其他非主属性。这称之为消除传递依赖。

> 满足2NF，非主键外的所有字段必须互不依赖

1.4. BCNF

在不存在多值属性、不存在实体的属性部分依赖于主属性、不存在非主属性依赖于其他非主属性的基础上。主属性内部不能有部分或传递依赖。这将消除对主属性子集的依赖，使主属性保持最简。

1.5. 4NF

在不存在多值属性、不存在实体的属性部分依赖于主键、不存在非主属性依赖于其他非主属性、主属性内部不能有部分或传递依赖的基础上。消除多值依赖，只允许函数依赖。

> 满足3NF，消除表中的多值依赖

1.6. 5NF

在不存在多值属性、不存在实体的属性部分依赖于主属性、不存在非主属性依赖于其他非主属性、主属性内部不能有部分或传递依赖、不存在多值依赖的基础上。消除连接依赖，并且必须保证数据完整性。

关系型数据库的几种设计范式（1NF 2NF 3NF BCNF 4NF 5NF） - 简书

2. 什么是事务

数据库事务（简称：事务）是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。

一个数据库事务通常包含了一个序列的对数据库的读/写操作。它的存在包含有以下两个目的：

- 为数据库操作序列提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。
- 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。

3. ACID

ACID，是指数据库管理系统（DBMS）在写入或更新资料的过程中，为保证事务（transaction）是正确可靠的，所必须具备的四个特性：

- 原子性（atomicity，或称不可分割性）：一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。
- 一致性（consistency）：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。
- 隔离性（isolation，又称独立性）：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（serializable）。
- 持久性（durability）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

4. 什么是事务隔离，事务隔离的级别?

事务隔离（Transaction Isolation）定义了数据库系统中一个操作的结果在何时以何种方式对其他并发操作可见。

并发控制描述了数据库事务隔离以保证数据正确性的机制。为了保证并行事务执行的准确执行，数据库和存储引擎在设计的时候着重强调了并发控制这一点。典型的事务相关机制限制数据的访问顺序（执行调度）以满足可序列化和可恢复性。限制数据访问意味着降低了执行的性能，并发控制机制就是要保证在满足这些限制的前提下提供尽可能高的性能。经常在不损害正确性的情况下，为了达到更好的性能，可序列化的要求会减低一些，但是为了避免数据一致性的破坏，可恢复性必须保证。

两阶段锁是关系数据库中最常见的提供了可序列化和可恢复性的并发控制机制，为了访问一个数据库对象，事务首先要获得这个对象的锁。对于不同的访问类型（如对对象的读写操作）和锁的类型，如果另外一个事务正持有这个对象的锁，获得锁的过程会被阻塞或者延迟。

很多数据库管理系统定义了不同的“事务隔离等级”来控制锁的程度。在很多数据库系统中，多数的数据库事务都避免高等级的隔离等级（如可序列化）从而减少对系统的锁定开销。

ANSI/ISO SQL定义的标准隔离级别如下：

4.1. 可序列化

最高的隔离级别。

在基于锁机制并发控制的DBMS实现可序列化，要求在选定对象上的读锁和写锁保持直到事务结束后才能释放。在SELECT的查询中使用一个“WHERE”子句来描述一个范围时应该获得一个“范围锁”（range-locks）。这种机制可以避免“幻影读”（phantom reads）现象。

当采用不基于锁的并发控制时不用获取锁。但当系统探测到几个并发事务有“写冲突”的时候，只有其中一个是允许提交的。

4.2. 可重复读

在可重复读（REPEATABLE READS）隔离级别中，基于锁机制并发控制的DBMS需要对选定对象的读锁（read locks）和写锁（write locks）一直保持到事务结束，但不要求“范围锁”，因此可能会发生“幻影读”。

4.3. 提交读

在提交读（READ COMMITTED）级别中，基于锁机制并发控制的DBMS需要对选定对象的写锁一直保持到事务结束，但是读锁在SELECT操作完成后马上释放（因此“不可重复读”现象可能会发生）。和前一种隔离级别一样，也不要求“范围锁”。

4.4. 未提交读

未提交读（READ UNCOMMITTED）是最低的隔离级别。允许“脏读”（dirty reads），事务可以看到其他事务“尚未提交”的修改。

通过比低一级的隔离级别要求更多的限制，高一级的级别提供更强的隔离性。标准允许事务运行在更强的事务隔离级别上。(如在可重复读隔离级别上执行提交读的事务是没有问题的)

事务隔离 - 维基百科，自由的百科全书

5. MySQL的MyISAM引擎与InnoDB引擎的区别

- 设计目标：InnoDB设计目标是处理大容量的数据，而MyISAM追求的是性能，两者产生的差异也是基于这点。InnoDB是MySQL的默认存储引擎。
- 事务处理：InnoDB支持事务和外键，MyISAM不支持。MyISAM强调的是性能，InnoDB支持的功能更加完整。InnoDB支持事务带来了一个好处，发生故障时可以通过事务日志来恢复数据库，MyISAM特别要命的一点是崩溃后不能安全恢复，所以对于表比较大的情况不要用。
- 效率和锁：两种存储引擎的效率差异来自于锁的方式差异，MyISAM是表锁，对数据库进行写操作时会锁住整个表，效率很低；确定要修改数据的范围时，InnoDB是行锁，只锁一行的数据，写操作很快。
- 索引：MyISAM支持全文索引，InnoDB不支持。
- `COUNT(*)`：MyISAM保存了表的行数，InnoDB没有。也就是说，执行`SELECT COUNT(*) FROM student`的操作时，MyISAM可以直接给出结果，而InnoDB要先扫描全

表。不过对于加了where条件的查询操作，效果是一样的。

- AUTO_INCREMENT：InnoDB下只能对自增字段单独建索引，MyISAM下可以和其它列一起建联合索引。

6. 悲观锁与乐观锁：

乐观并发控制(乐观锁)和悲观并发控制（悲观锁）是并发控制主要采用的技术手段。

6.1. 悲观锁

悲观并发控制（又名“悲观锁”，Pessimistic Concurrency Control，缩写“PCC”）是一种并发控制的方法。它可以阻止一个事务以影响其他用户的方式来修改数据。如果一个事务执行的操作都某行数据应用了锁，那只有当这个事务把锁释放，其他事务才能够执行与该锁冲突的操作。悲观并发控制主要用于数据争用激烈的环境，以及发生并发冲突时使用锁保护数据的成本要低于回滚事务的成本的环境中。

悲观锁的流程：

- 在对任意记录进行修改前，先尝试为该记录加上排他锁（exclusive locking）。
- 如果加锁失败，说明该记录正在被修改，那么当前查询可能要等待或者抛出异常。
- 如果成功加锁，那么就可以对记录做修改，事务完成后解锁。
- 其间如果有其他对该记录做修改或加排他锁的操作，都会等待解锁或直接抛出异常。

6.2. 乐观锁

乐观并发控制（又名“乐观锁”，Optimistic Concurrency Control，缩写“OCC”）是一种并发控制的方法。它假设多用户并发的事务在处理时不会彼此互相影响，各事务能够在不产生锁的情况下处理各自影响的那部分数据。在提交数据更新之前，每个事务会先检查在该事务读取数据后，有没有其他事务又修改了该数据。如果其他事务有更新的话，正在提交的事务会进行回滚。

实现数据版本有两种方式，第一种是使用版本号，第二种是使用时间戳。