

数据库的设计的重要性，软件设计重要，数据库是软件设计的第一步

获得有用数据库的第一步是了解数据库的目标。要完成什么任务，怎样使数据库有用

数据库是信息的容器，由此，笔记本，名片盒，档案柜都能做为数据库

归纳各种信息容器的优缺点，获得有用数据库应具有的特性

具备CURD

以不同的方式检索

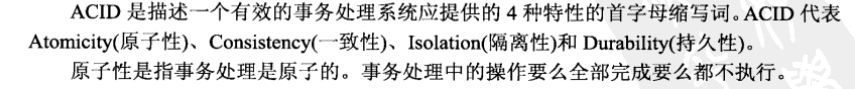
一致性，检索相同行要得到一致的结果

有效性，对更改，删除，插入根据规则进行验证

轻松的更正错误

速度

原子事务处理，提供4种特性ACID，



**一致性**

一个事务可以封装状态改变（除非它是一个只读的）。事务必须始终保持系统处于一致的状态，不管在任何给定的时间[**并发**](http://baike.baidu.com/item/%E5%B9%B6%E5%8F%91)事务有多少。

也就是说：如果事务是[**并发**](http://baike.baidu.com/item/%E5%B9%B6%E5%8F%91)多个，系统也必须如同串行事务一样操作。其主要特征是保护性和不变性(Preserving an Invariant)，以转账[案例](http://baike.baidu.com/item/%E6%A1%88%E4%BE%8B)为例，假设有五个账户，每个账户余额是100元，那么五个账户总额是500元，如果在这个5个账户之间同时发生多个转账，无论[**并发**](http://baike.baidu.com/item/%E5%B9%B6%E5%8F%91)多少个，比如在A与B账户之间转账5元，在C与D账户之间转账10元，在B与E之间转账15元，五个账户总额也应该还是500元，这就是保护性和不变性



**持久性**

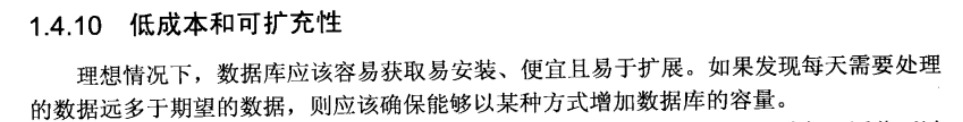
在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

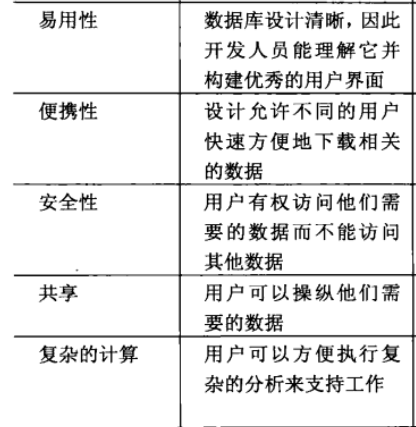
由于一项操作通常会包含许多子操作，而这些子操作可能会因为硬件的损坏或其他因素产生问题，要正确实现ACID并不容易。ACID建议数据库将所有需要更新以及修改的资料一次操作完毕，但实际上并不可行。

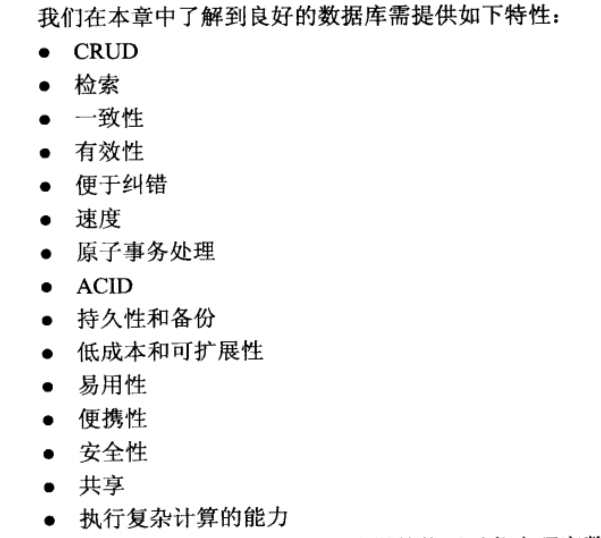
目前主要有两种方式实现ACID：第一种是Write ahead logging，也就是日志式的方式(现代数据库均基于这种方式)。第二种是Shadow paging。

WAL 的中心思想是对数据文件 的修改（它们是表和索引的载体）必须是只能发生在这些修改已经 记录了日志之后 -- 也就是说，在日志记录冲刷到永久存储器之后． 如果我们遵循这个过程，那么我们就不需要在每次事务提交的时候 都把数据页冲刷到磁盘，因为我们知道在出现崩溃的情况下， 我们可以用日志来恢复数据库：任何尚未附加到数据页的记录 都将先从日志记录中重做（这叫向前滚动恢复，也叫做 REDO） 然后那些未提交的事务做的修改将被从数据页中删除 （这叫向后滚动恢复 - UNDO）。也就是，提交事务后日志写到硬盘，数据可以在内存先不写硬盘

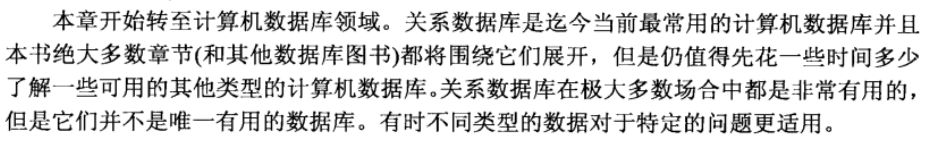
备份

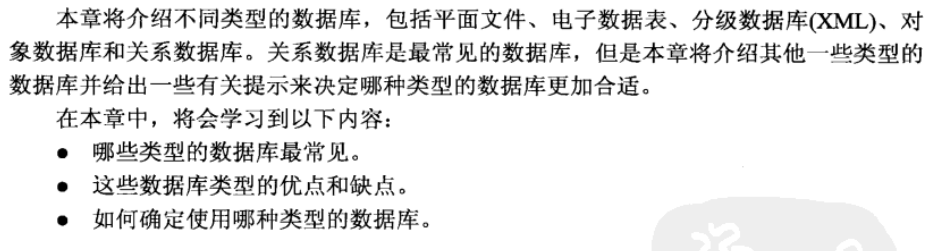


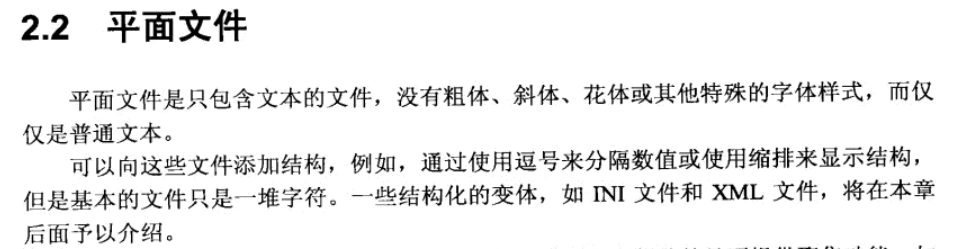


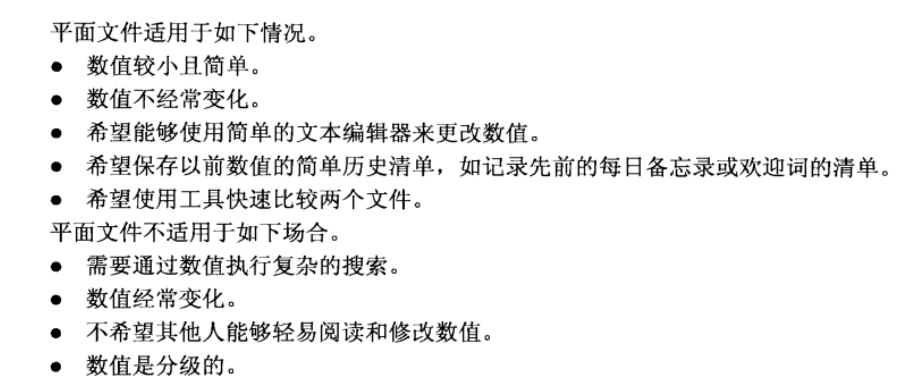


数据库类型

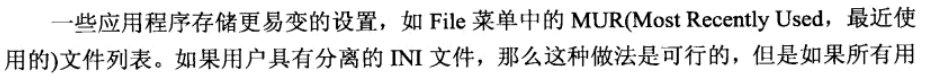


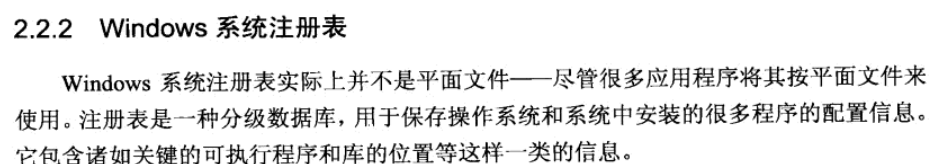


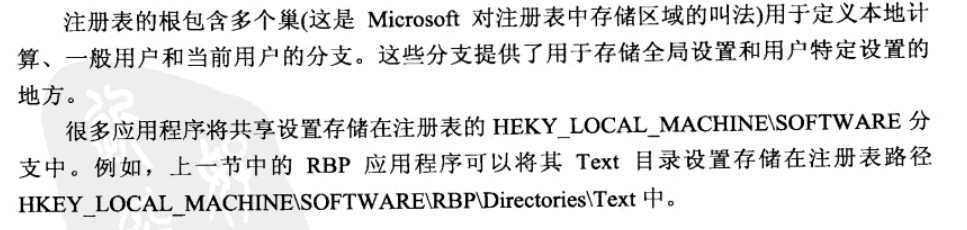












没懂

