 一个好的数据库产品不等于就有一个好的应用系统，如果不能设计一个合理的数据库模型，不仅会增加客户端和服务器段程序的编程和维护的难度，而且将会影响系统实际运行的性能。一般来讲，在一个MIS系统分析、设计、测试和试运行阶段，因为数据量较小，设计人员和测试人员往往只注意到功能的实现，而很难注意到性能的薄弱之处，等到系统投入实际运行一段时间后，才发现系统的性能在降低，这时再来考虑提高系统性能则要花费更多的人力物力，而整个系统也不可避免的形成了一个打补丁工程。笔者依据多年来设计和使用数据库的经验，提出以下一些设计准则，供同仁们参考。 命名的规范 不同的数据库产品对对象的命名有不同的要求，因此，数据库中的各种对象的命名、后台程序的代码编写应采用大小写敏感的形式，各种对象命名长度不要超过30个字符，这样便于应用系统适应不同的数据库。 游标（Cursor）的慎用 游标提供了对特定集合中逐行扫描的手段，一般使用游标逐行遍历数据，根据取出的数据不同条件进行不同的操作。尤其对多表和大表定义的游标（大的数据集合）循环很容易使程序进入一个漫长的等特甚至死机，笔者在某市《住房公积金管理系统》进行日终帐户滚积数计息处理时，对一个10万个帐户的游标处理导致程序进入了一个无限期的等特（后经测算需48个小时才能完成）(硬件环境：Alpha/4000 128Mram ,Sco Unix ,Sybase 11.0)，后根据不同的条件改成用不

同的UPDATE语句得以在二十分钟之内完成。示例如下：
Declare Mycursor cursor for 0select count_no from COUNT Open Mycursor Fetch Mycursor into @vcount_no While (@@sqlstatus=0) Begin If @vcount_no=’’条件1 操作1 If @vcount_no=’’条件2 操作2 Fetch Mycursor into @vcount_no End 改为 Update COUNT set 操作1 for 条件1 Update COUNT set 操作2 for 条件2
在有些场合，有时也非得使用游标，此时也可考虑将符合条件的数据行转入临时表中，再对临时表定义游标进行操作，可时性能得到明显提高。笔者在某地市〈电信收费系统〉数据库后台程序设计中，对一个表（3万行中符合条件的30多行数据）进行游标操作(硬件环境：PC服务器，PII266 64Mram ,NT4.0 Ms Sqlserver 6.5)。 示例如下： Create #tmp /* 定义临时表 */ (字段1 字段2 ) Insert into #tmp 0select * from TOTAL where 条件 /* TOTAL中3万行 符合条件只有几十行 */ Declare Mycursor cursor for 0select * from #tmp /*对临时表定义游标*/ 索引(Index)的使用原则 创建索引一般有以下两个目的：维护被索引列的唯一性和提供快速访问表中数据的策略。大型数据库有两种索引即簇索引和非簇索引，一个没有簇索引的表是按堆结构存储数据，所有的数据均添加在表的尾部，而建立了簇索引的表，其数据在物理上会按照簇索引键的顺序存储，一个表只允许有一个簇索引，因此，根据B树结构，可以理解添加任何一种索引均能提高按索引列查询的速度，但会降低插入、更新、删除操作的性能，尤其是当填充因子（Fill Factor）较大时。所以对索引较多的表进行频繁的插入、更新、删除操作，建表和索引时因设置较小的填充因子，以便在各数据页中留下较多的自由空间，减少页分割及重新组织的

工作。 数据的一致性和完整性 为了保证数据库的一致性和完整性，设计人员往往会设计过多的表间关联（Relation），尽可能的降低数据的冗余。表间关联是一种强制性措施，建立后，对父表（Parent Table）和子表(Child Table)的插入、更新、删除操作均要占用系统的开销，另外，最好不要用Identify属性字段作为主键与子表关联。如果数据冗余低，数据的完整性容易得到保证，但增加了表间连接查询的操作，为了提高系统的响应时间，合理的数据冗余也是必要的。使用规则（Rule）和约束（Check）来防止系统操作人员误输入造成数据的错误是设计人员的另一种常用手段，但是，不必要的规则和约束也会占用系统的不必要开销，需要注意的是，约束对数据的有效性验证要比规则快。所有这些，设计人员在设计阶段应根据系统操作的类型、频度加以均衡考虑。 事务的陷阱 事务是在一次性完成的一组操作。虽然这些操作是单个的操作，SQL Server能够保证这组操作要么全部都完成，要么一点都不做。正是大型数据库的这一特性，使得数据的完整性得到了极大的保证。 ---- 众所周知，SQL Server为每个独立的SQL语句都提供了隐含的事务控制，使得每个DML的数据操作得以完整提交或回滚，但是SQL Server还提供了显式事务控制语句 BEGIN TRANSACTION 开始一个事务 COMMIT TRANSACTION 提交一个事务 ROLLBACK TRANSACTION 回滚一个事务 事务可以嵌套，可以通过全局变量@@trancount检索到连接的事务处理嵌套层次。需要加以特别注意并且极容易使编程人员犯错误的是，每个显示或隐含的事物开始都使得该变量加1，每个事务的提交使该变量减1，每个事务的回滚都会使得该变量置0，而只有当该变量为0时的事务提交

（最后一个提交语句时），这时才把物理数据写入磁盘。 数据库性能调整 在计算机硬件配置和网络设计确定的情况下，影响到应用系统性能的因素不外乎为数据库性能和客户端程序设计。而大多数数据库设计员采用两步法进行数据库设计：首先进行逻辑设计，而后进行物理设计。数据库逻辑设计去除了所有冗余数据，提高了数据吞吐速度，保证了数据的完整性，清楚地表达数据元素之间的关系。而对于多表之间的关联查询（尤其是大数据表）时，其性能将会降低，同时也提高了客户端程序的编程难度，因此，物理设计需折衷考虑，根据业务规则，确定对关联表的数据量大小、数据项的访问频度，对此类数据表频繁的关联查询应适当提高数据冗余设计。 数据类型的选择 数据类型的合理选择对于数据库的性能和操作具有很大的影响，有关这方面的书籍也有不少的阐述，这里主要介绍几点经验。 Identify字段不要作为表的主键与其它表关联，这将会影响到该表的数据迁移。 Text和Image字段属指针型数据，主要用来存放二进制大型对象（BLOB）。这类数据的操作相比其它数据类型较慢，因此要避开使用。 日期型字段的优点是有众多的日期函数支持，因此，在日期的大小比较、加减操作上非常简单。但是，在按照日期作为条件的查询操作也要用函数，相比其它数据类型速度上就慢许多,因为用函数作为查询的条件时，服务器无法用先进的性能策略来优化查询而只能进行表扫描遍历每行。例如：要从DATA_TAB1中（其中有一个名为DATE的日期字段）查询1998年的所有记录。 Select * from DATA_TAB1 where datepart(yy,DATE)=1998 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com