* 封锁对象的大小称为封锁粒度，封锁对象可以是逻辑单元，也可以是物理单元。
* 三层客户机/服务器结构把数据库应用系统分为表示层、功能层、数据层等三个层次。
* SQL Server提供了基于日志文件的恢复技术和基于数据备份的恢复技术两种数据恢复技术。
* SQL Server的企业管理器（Enterprise Manager）是一个集成化的数据库操作环境工具。
* 试分析系统设置UNDO操作和REDO操作的必要性。

UNDO和REDO是系统内部命令。

在DB恢复时，对于已经COMMIT但更新仍停留在缓冲区的事务要执行REDO（重做）操作，即根据日志内容把该事务对DB修改重做一遍。

对于还未结束的事务要执行UNDO（撤销）操作，即据日志内容把该事务对DB已作的修改撤销掉。 设置UNDO和REDO操作，是为了使数据库具有可恢复性。

* 试分析“检查点机制”，COMMIT语句与检查点时刻的操作如何协调。

“检查点机制”的主要思想是在检查点时刻才真正做到把对数据库的修改写到磁盘。

在数据库恢复时,只有那些在最后一个检查点到故障点之间还在执行的事务才需要恢复。

事务在COMMIT时,事务对数据库的更新已提交,但对数据库的更新可能还留在内存的缓冲区,在检查点时刻才真正写到磁盘。

因此事务的真正结束是在COMMIT后还要加上遇到检查点时刻。

* SQL跟踪器用于供数据库管理员和应用程序开发员监测和记录数据库的活动
* 系统故障-造成数据库处于不一致状态的原因：

1.某些未完成的事务对数据库的更新已写入数据库；

2.有些已经提交的事务对数据库的更新还留在缓冲区内没来得及写入数据库。

* 故障恢复步骤：

1.正向扫描日志文件，找出在故障发生前已经提交的事务，并将这些事务标记记入重做队列。同时还要找出故障发生时尚未完成的事务(只有BEGIN TRANSACTION记录，没有相应的COMMIT记录)，并将这些事务标记记入撤消队列。

2.对撤消队列中的各个事务进行撤消(UNDO)处理：反向扫描日志文件，对每个UNDO事务的更新操作执行逆操作。

3.对重做队列中的各个事务进行重做(REDO)处理：正向扫描日志文件，对每个REDO事务重新执行日志文件登记的操作。

* DBMS利用事务日志文件保存所有数据库事务的更新操作。
* 数据库中数据发生错误，往往是由非法的更新引起的。
* 判断一个并发调度是否正确，可以用可串行化概念来解决。
* 事务的原子性是由DBMS的事务管理子系统实现的。
* 并发事务发生死锁属于事务故障