第五章 数据库管理

挑战性问题——数据库管理

一、数据库管理问题探讨

1、在事务编程中，如何避免死锁的产生？

答：①在并发事务执行时，预防死锁；②在死锁出现后，其中一个事务释放资源以解除死锁。

2、事务中，回退点的设置考虑因素有哪些？

答：在每个重要的操作节点后、或后续操作出错可能性较大的节点后设置回退点。

3、在 DBMS 中，如何设置显式事务和隐含事务模式？

答：隐式事务是 SQL Server 为你而做的事务.隐式事务又称自动提交事务。如果运行一条 insert 语句，SQL Server 将把它包装到事务中；如果此 insert 语句失败, SQL Server 将回滚或取消这个事务。每条 SQL 语句均被视为一个自身的事务。显示事务是一种由你自己指定的事务，这种事务允许你自己决定哪批工作必须成功完成，否则所有部分都不完成。语法：`SET IMPLICIT_TRANSACTIONS { ON | OFF }`，选择 `ON` 将连接设置为隐式事务模式，选择 `OFF` 将连接设置为显式事务模式。

4、在 DBMS 中，事务隔离级别应如何设置？

答：

隔离级别	脏读	不可重复读	幻像读	丢失更新
读取未提交	可能	可能	可能	可能
读取已提交	不可能	可能	可能	可能
可重复读	不可能	不可能	可能	可能
可串行化	不可能	不可能	不可能	不可能

二、工程案例的数据库编程优化探讨

针对成绩管理系统，探讨数据库管理问题。

1、在成绩管理系统中，创建教务管理人员账户，并为其赋予相应的权限。

答：

```
CREATE USER "Admin" WITH
    SUPERUSER
    CREATE DB
    CREATE ROLE
    NOINHERIT
    LOGIN
    REPLICATION
    CONNECTION LIMIT -1
    PASSWORD '123456';
GRANT SELECT, INSERT, UPDATE, DELETE ON grade TO "Admin";
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON student TO “Admin”;
GRANT SELECT, INSERT, UPDATE, DELETE ON teacher TO “Admin”;
GRANT SELECT, INSERT, UPDATE, DELETE ON course TO “Admin”;
```

2、在成绩管理系统中，如何确保教师用户只能修改自己承担课程的信息，教材中的数据库设计是否能够满足该要求，如果不能，你考虑增加哪些数据表？

答：教师用户仅能对自己承担课程的课程表进行修改；教材中的数据库设计不能满足该要求，可以增设教师承担课程表及课程成绩表，将两表关联，将得到的此表作为教师用户可以修改的表格，即满足了要求。

3、在成绩管理系统，可能有上万的学生，如果为每个学生建立一个用户，是否是最好的选择，有没有更好的解决方法？

答：不是最好的选择。先创建学生角色（Role），给这些的上万的学生用户，都赋予学生的角色就节省了为每个学生建立用户的操作。

4、在成绩管理系统，哪些操作可能会产生死锁，请描述一个可能的产生死锁的应用流程？

答：可能产生死锁的原因：

(1) 事务对所分配到的资源进行排他性使用，即在一段时间内某资源只由一个事务占用。如果此时还有其他事务请求资源，则请求者只能等待，直至占有资源的事务解锁释放资源。

(2) 事务已经保持至少一个资源，但又提出了新的资源请求，而该资源已被其他事务占有，此时请求事务被阻塞，但又对自己已获得的其他资源保持不放。

(3) 事务占用已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。

(4) 在发生死锁时，必然存在一个事务-资源的等待环路，即事务集合{T0, T1, T2, …, T*n*}中，T0 正在等待一个 T1 占用的资源，T1 正在等待 T2 占用的资源，依此类推，T*n* 正在等待已被 T0 占用的资源。

可能的应用流程：

有两个事务 T1 和 T2，T1 需要对成绩表 Table1、学生表 Table2 进行访问，T2 也需要对这两个表访问，只是与 T1 的访问顺序相反。T1 首先对 Table1 进行排他锁定，同时 T2 对 Table2 进行排他锁定。然后它们对这两个表分别进行访问操作。当 T1 事务完成 Table1 访问处理后，将锁定访问 Table2，但因 Table2 资源仍被 T2 事务锁定，T1 事务处于等待状态。同样，当 T2 事务完成 Table2 访问处理后，将锁定访问 Table1，但因 Table1 资源仍被 T1 事务锁定，T2 事务也处于等待状态。由于 T1 事务和 T2 事务都在等待对方释放自己所需的共享资源锁定后，才能继续执行，因此，出现了谁也执行不下去的死锁状态。