Ch05 数据库的保护

1. 数据库保护概况

1. 数据库破坏类型：非法用户、各种故障、非法数据、多用户并发访问
2. DBMS对于数据库的保护措施：

- 非法用户——设置权限
- 各种故障——提供故障恢复
- 非法数据——完整性约束
- 多用户并发访问——提供并发控制

2. 数据库的安全性

1. 数据库的安全性概况：
   非法使用数据库的情况：

- 通过合法程序绕过DBMS及其授权机制
- 直接或通过应用程序执行非授权操作
- 通过多次合法地查询数据库中的数据，推导出了一些保密数据

2. 用户标识与鉴别——系统提供的最外层的安全保护措施
3. 常用存取控制方法

- 自主存取控制（DAC）：用户对不同的数据对象有不同的存取权限，不同用户对同一对象有不同存取权限
- 强制存储权限（MAC）：每个数据对象被标识一个密级，每一用户对应某一级别的许可证。只有拥有合法许可证的用户拥有对某一对象的存取操作

4. 自主存取控制（DAC）

存取权限由两个要素构成：数据对象和操作类型

授权：`GRANT 权限 ON 数据对象  TO 用户 WITH GRANT OPTION`

撤权：`REVOKE 权限 ON 数据对象 FROM 用户`

接受权限的用户可以是一个或多个，如果是全体用户，可简写成`PUBLIC`

全部权限可以简写成`ALL PRIVILEGES`

`WITH GRANT OPTION`使得被授权的用户可以继续传播他拥有的权限，如果不写上这一句，不能传播他的权限

例1: 把对Student表的查询权限授予用户U1

```
GRANT SELECT ON Student TO U1
```

例2: 把对Student和Course表的所有权限授予用户U2和U3

```
GRANT ALL PRIVILEGES ON Stuent, Course TO U2, U3
```

例3: 把对SC表的查询权限授予所有用户

```
GRANT SELECT ON SC TO PUBLIC
```

例4: 把查询Student表和修改学生学号的权限授予用于U4

```
GRANT SELECT, UPDATE(sno) ON Student TO U4
```

例5: 把U4修改学生学号的权限收回

```
REVOKE UPDATE(sno) ON Student FROM U4
```

本节习题:

1. SQL中的视图提高了数据库系统的安全性
2. 安全性控制的防范对象是非法用户
3. 在数据库系统中，定义用户可以对哪些数据对象进行何种操作被称为授权
4. 把关系SC的属性GRADE的修改权限授予用户ZHAO的SQL语句是

   `GRANT UPDATE(GARDE) ON SC TO ZHAO`
5. 对下面两个关系模式
   学生（学号，姓名，年龄，性别，家庭住址，班级号）
   班级（班级号，班级名，班主任，班长）

   ```
   -- 授予用户U1对两个表的所有权限，并可给其他用户授权
   GRANT ALL PRIVILEGES ON TABLE 学生, 班级 TO U1 WITH GRANT OPTION

   -- 授予用户U2对学生表具有查看权限，对家庭住址具有更新权限
   GRANT SELECT, UPDATE(家庭住址) ON TABLE 学生 TO U2

   -- 将对班级表查看的权限授予所有用户
   GRANT SELECT ON TABLE 班级 TO PUBLIC

   -- 将对学生表的查询、更新权限授予角色R1
   GRANT SELECT, UPDATE ON TABLE 学生 TO R1

   -- 将角色R1授予用户U1，并且U1可继续授权给其他角色
   GRANT R1 TO U1 WITH ADMIN OPTION
   ```
6. 用DCL完成如下操作权限设置:

   职工(职工号, 姓名, 年龄, 职务, 工资, 部门名)

用户杨兰具有从每个部门职工中查询最高工资、最低工资、平均工资的权限，但她不能查看每个人的具体工资。

```
-- 创建视图
CREATE VIEW Salary
    AS
    SELECT 部门名, MAX(工资), MIN(工资), AVG(工资)
    FROM 职工 GROUP BY 部门名

-- 授权
GRANT SELECT ON Salary TO 杨兰
```

数据库的完整性约束

1. 静态约束：隐式约束、固有约束、显式约束
   动态约束：变迁约束
2. 静态约束

- 隐式约束：域约束、主键约束、唯一约束、一般性约束（检查约束和断言）和参照完整性
- 固有约束：关系的属性不可分，即原子性

3. 显式约束：不能通过前两种约束满足约束要求的静态约束时，使用显式约束

故障恢复

1. 事务的ACID准则

- 原子性（Atomicity）：一个事务对数据库的所有操作是一个整体，要么成功，要么回退
- 一致性（Consistency）
- 隔离性（Isolation）：并发的几个事务在用户看来是单独执行的
- 持久性（Durability）：事务一旦执行成功，对数据库的影响是持久的

2. 故障恢复技术：

- 仅以后备副本为基础的恢复技术
- 以后备副本和日志为基础的恢复技术
- 基于多副本的恢复技术

3. 日志（Log）：供恢复用的DB运行情况的记录，其内容有：前像、后像、和事务状态（各种事务表）。
4. 前像（BI）：事务所在的物理块在更新前的映像。如果要撤销更新（undo），使DB恢复到更新前的数据，可使用前像覆盖事务所在的物理块。
5. 后像（AI）：事务所在的物理块在更新后的映像。如果要重做更新（redo），使DB恢复到更新后的数据，可使用后像覆盖事务所在的物理块。
6. 日志基本内容：

- 活动事务表（ATL）：记录正在执行、尚未提交的事务ID
- 提交事务表（CTL）：记录已经提交的事务ID

- 前像文件
- 后像文件

7. 发生故障时，

- 如果事务未提交，用前像恢复（undo）
- 如果事务已提交，用后像恢复（redo）

DBMS围绕更新事务做的工作

1. 提交规则：后像应该在事务提交前写入DB或日志
2. 先记后写规则：如果后像在事务提交前写入日志，需要先将前像写入日志，以便事务失败的情况下undo.

故障类型及恢复对策

1. 事务的成功执行与结束：

- 一个事务以`BEGIN TRANSACTION`的成功执行开始
- 以`COMMIT`或`ROLLBACK`结束

2. 提交点：`COMMIT`创建的一个提交点（Commit Point）
3. 可恢复的故障：

- 事务故障
- 系统故障
- 介质故障

并发控制

1. 并发控制可能出现的问题；

- 丢失更新
- 读脏数据
- 读值不可复现

2. 丢失更新（覆盖未提交的数据）

原因：两个事务对同一数据并发写入，写-写冲突

3. 读脏数据（读未提交的数据）

原因：一事务读取了另一事务尚未提交的数据，写-读冲突

4. 读值不可复现（两次读取的数据不同）

原因：由于另一事务对同一事务的写入，一事务对数据两次读取的值不同，读-写冲突

5. 用前趋图求解。类似拓扑图。

本节习题：

1. 事务有多个性质，其中不包含 B

A. 一致性　B. 唯一性　C. 原子性　D. 隔离性

2. 对数据库并发操作可能带来的问题包括　A

A. 读取脏数据　B. 带来数据的冗余　C. 未被授权的用户非法存取数据　D. 破坏数据独立性

基于锁的并发控制协议

1. 基本加锁类型：

- 互斥锁（X锁）
- 共享锁（S锁）

本节习题：

1. 为解决“丢失更新”问题，事务在更新一个数据集合前，必须获得它的X锁
2. 解决并发操作带来的数据不一致问题普遍采用加锁技术
3. 如果事务T获得了数据项Q上的互斥锁，则T对Q可读可写