数据库安全

一.数据库介绍

1.概念

- 保存数据、按一系列规则组织数据、数据间具有关联关系
- 数据库管理员：定义数据的组织形式、控制谁可以访问哪些数据
- 用户通过数据库管理系统(DBMS)和数据库交互

2.组成

- 数据文件：由记录组成
- 记录：由域和字段组成

3.模式

- **模式**：数据库的逻辑结构
- 数据库的子模式: 一个用户仅能访问数据库的部分内容

4.优点

- 建立在文件系统之上
- 访问时共享
- 最小冗余
- 数据一致性
- 数据完整性
- 控制访问

二.安全需求

1.完整性

a.物理数据库完整性

b.逻辑完整性

- 仅授权用户可以更新
- 个人数据项不可读

c.数据项完整性

- 正确、精确

2.审计

a.优点

- 维护完整性，受破坏后可恢复
- 用户累加式访问受保护数据
- 后台记录用户访问记录 : 推理用户意图

b.缺点

- 粒度问题：
    - 记录级
    - 域级

3.访问控制

- 操作系统或计算机系统的组件

  –操作系统中客体--文件: 无关联的数据项

  –域, 记录: 有关联关系

  –大小, 粒度不同, 影响进程的访问效率
- 受限的访问

  –视图, 关系, 域, 记录

4.鉴别用户

- DBMS进行严格的用户鉴别

三.可靠性与完整性

1.数据库完整性

- 防止数据库整体性损坏, 硬盘毁坏、主数据库索引项毁坏. 采用操作系统的完整性控制措施、恢复措施

2.数据项完整性

- 写入特定的数据值、仅授权用户可写. 恰当的访问控制可保护数据库免受非授权用户破坏

3.数据项精度

- 正确的数据值写入数据项. 检查域值可以防止插入不合适的数据值. 设置限制条件，以检测不正确的数据

4.操作系统级保护

- 周期性备份数据库文件和用户文件

  保护文件

  OS 对所有数据做完整性检查
- 两阶段更新

  准备阶段 : 准备更新数据, 不对数据库做改变

  提交阶段: 设置 **提交标志** 写入数据库, 永久性改变

> *eg.数据库中包含公司的办公用品使用和支出：纸、笔、夹子等*
>
> *一个部门申请50盒夹子, 仓库中存有107盒, 如果仓库存量少于100盒则需要外购*
>
> *准备阶段 :*
>
> 1. *检查数据库中的COMMIT-FLAG. 如果值为1, 不能执行操作. 中止或等待, 检查COMMIT-FLAG直到值为0*
> 2. *比较库存夹子数量是否超过申请数量; 如果少于申请数量则中止*
> 3. *计算：TCLIPS = ONHAND - REQUISITION*
> 4. *计算部门开销的生育办公费用BUDGET*
>
> *TBUDGET = BUDGET - COST, 其中COST是50盒夹子的价钱*
>
> 5. *检查TCLIPS是否低于最小库存量; 低于最小库存量, 设置 TREORDER = TRUE; 否则设置 TREORDER = FALSE*
>
> *提交阶段:*
>
> 1. *置COMMIT-FLAG=1*
> 2. *拷贝 TCLIPS 到数据库中的 CLIPS*
> 3. *拷贝 TBUDGET 到数据库中的 BUDGET*
> 4. *拷贝 TREORDER 到数据库中的 REORDER*
> 5. *通知申请部门领取物品. 记录完成交易日志*
> 6. *修改COMMIT-FLAG=0*

5.冗余/内部一致性

- 检测错误、更改
- 冗余项: 记录可以是重复的, 如果出错,冗余域可提供修复数据
- 恢复: 日志

6.并发性/一致性

- 两用户共享同一数据库:

  读: 无冲突　修改: 可能冲突

- 读和写并发访问

  用户A正在更新一个数据值时，另一用户B想读取该值

  锁定读请求，当更新结束后才可读

> *示例：淘宝购物*
> *1、某人想买1件衣服，选中了一件衣服并且衣服只剩1件，*
> *如果两个人同时下单买衣服怎么办，怎么解决抢购问题？*
>
> *选中商品下单时设置提交标志，锁定该项值，其它人不能再选择该商品*
> *select xx from table where yy and commit-flag=0;*
> *commit-flag=1;*
> *Torder=1, set user,address,post,phone,...0*
> *commit-flag=0;*
>
> *2、某人想买1件衣服，选中了一件衣服并且衣服只剩1件，其中一人选中衣服并且下单，但迟迟不交款，会妨碍他人购买衣服，怎么防止恶意下单而不成交？*
>
> *选中商品下单时设置提交标志，锁定该项值，同时设置锁定时间，超时后自动把商品状态设置为空闲状态*
>
> *select xx from table where yy and commit-flag=0;*
> *commit-flag=1;*
> *Torder=1, set user,address,post,phone,...*
> *set lock-time=10*
> *commit-flag=1;*
> *while lock-time=0*
> *{commit-flag=1;*
> *Torder=0;*
> *commit-flag=0;}*

7.监控

- 范围检查
- 状态检查: 数据库的全局条件. 不满足限制条件, 部分值错误
- 交易限制: 必须满足限制条件，数据库才可改变

四.敏感数据

- 敏感数据：数据不能公开
- 数据项是否敏感，取决于数据库和数据的涵义

1.难点

- 限制用户仅能访问合法数据
- 确保敏感数据不泄露给非授权用户

2.导致数据敏感的因素

- 与生俱来的敏感
- 从敏感源来的
- 声称敏感
- 部分属性敏感、部分记录敏感
- 以前信息被泄露了，导致敏感

3.访问决策

- **数据的可用性**
- **访问的可用性**: 仅确定数据是否敏感过于简单, 间接查询、多次查询构成查询序列获得敏感数据
- **确保被鉴别**: 在特定时间可访问

4.泄露类型

- 精确值、范围、负值、是否存在、概率值

5.安全审查精度

- 精度，用来保护所有敏感数据

五.推理

- 推理：根据不敏感数据得到、推理敏感数据.

1.攻击类型

a.直接攻击

- **查询敏感域**

> *Select name from table where sex='M' and drugs=1*
>
> *Select name from table where (sex='M' ∧drugs=1) ∨(sex≠M ∧ sex≠F) ∨(dorm='West')*

b.间接攻击

- 利用统计方法

c.攻击轨迹

- 利用多个查询得到某个具体数值
- 给出n 和 n - 1, 可以容易计算出单个元素信息

> *想知道：Holmes Hall中Caucasians族女性人数*
> ***Count (SEX = F ∧ RACE=C ∧ DORM=Holmes)***

- **防御措施**：DBMS得到答案为1时, 应拒绝查询，因为查询结果会泄露信息

d.线性分析

- 是一种特殊的脆弱性攻击有

  一点逻辑、代数, 一点幸运性, 通过构造一系列查询，得到几个不同集合的值间接推测个体值如, 系统中的 5个查询都不能得到个体信息c.

  但, 5个等式合在一起，则得到每个个体信息，造成泄露

2.防御措施

- 控制统计性推理攻击
- 控制查询: 防止直接获得敏感数据
- 对个体值控制查询:

  不回答: 对查询拒绝响应
- 隐藏: 数据库不提供精确数值
- 对推理问题：无完美的解决方案

  对明显敏感的信息要防止直接查询

  追踪用户企图

  对敏感数据做处理

六.多级数据库

- 数据不仅分为两类：敏感或不敏感
- 敏感性是属性的函数, 敏感性取决于属性和使用
- 数据库安全特征

  数据库中，某个元素的安全性和其他元素的安全性需求不同，需要多个安全级

  采用sum, count等聚合方法时，不同组的元素安全级不同，可高可低

七.多级安全方法