第四五章 数据库安全性与完整性

- 学习内容

- 4.1 计算机安全性概论
- 4.2 数据库安全性控制方法
- 4.3 数据库完整性概述
- 4.4 数据库完整性控制的方法
- 4.5 小结

• 学习目标

- 掌握数据库安全性的基本概念
- 掌握数据库安全性控制的基本方法
- 掌握数据库完整性的基本概念和类别
- 理解数据库安全性与完整性的区别和联系

4.1 计算机安全性概论

- 1. 定义
  - 是指保护数据库,以防止不合法的使用所造成的数据泄露、更改或破坏。
- 2. 重要性
  - 数据库系统中大量数据集中存放,许多用户直接共享。
  - 系统安全保护措施是否有效是数据库系统的主要性能指标之一。
  - 自然安全性+系统安全性

数据库安全问题频发

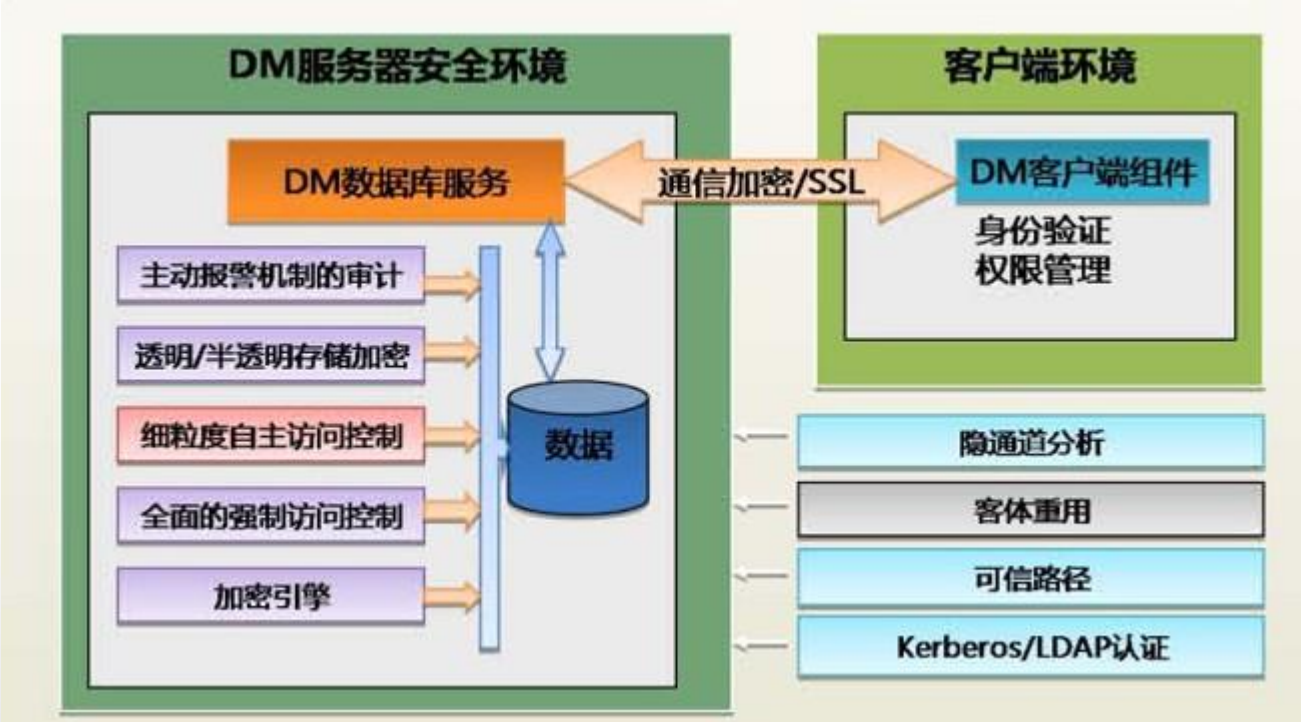
- 实例：Oracle数据库受到网络病毒攻击
- 现象：数据库遭遇重启失败（图为报错现场）

数据库恶意注入攻击

```
Total System Global Area  776646656 bytes
Fixed Size                  2257272 bytes
Variable Size             507514504 bytes
Database Buffers          264241152 bytes
Redo Buffers                2633728 bytes
Database mounted.
ORA-01092: ORACLE instance terminated. Disconnection forced
ORA-00704: bootstrap process failure
ORA-00704: bootstrap process failure
ORA-00600: internal error code, arguments: [16703], [1403], [20], [], [], [],
[], [], [], [], [], []
Process ID: 7425
Session ID: 1 Serial number: 5
```

- 原因查找：
  - Tab$中数据被清空，导致数据库启动时一致性检查失败；
  - 进一步定位：发现客户数据库中存在3个恶意存储过程和2个恶意触发器。

国产数据库-达梦数据库安全版



达梦数据库安全版-安全体系结构图

数据库是数字产业的核心引擎，研发具有自主知识产权的基础软硬件设施，构建国产自主IT底层生态，是数字信息安全的底座，也是产业数字化转型的关键。

数据库安全性

- 问题的提出
  - 数据库的一大特点是数据可以共享
  - 数据共享必然带来数据库的安全性问题
  - 数据库系统中的数据共享不能是无条件的共享

例：军事秘密、国家机密、新产品实验数据、
市场需求分析、市场营销策略、销售计划、
客户档案、医疗档案、银行储蓄数据

数据库安全性

4.1 计算机安全性概论(续)

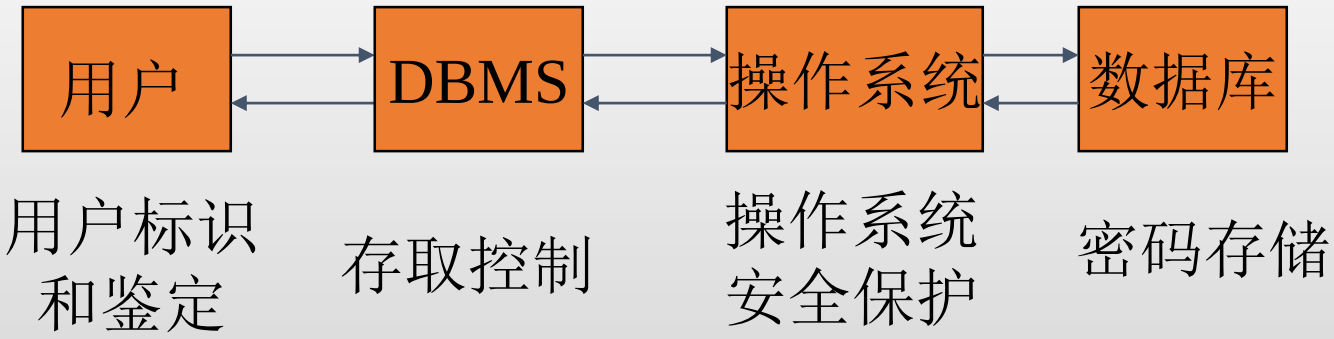
- 3. 计算机系统三类安全性问题
  - 技术安全
  - 管理安全
  - 政策法律类
- 4. 计算机系统安全
  - 与信息安全标准有关的组织
  - 国际信息系统安全标准
  - 国内信息系统安全标准

  TCSEC将系统划分为四组七个等级

  D;C(C1,C2);B(B1,B2,B3);A(A1)

4.2 数据库安全性控制方法

- 4.2.1 用户标识和鉴别
- 4.2.2 存取控制
- 4.2.3 视图机制
- 4.2.4 审计
- 4.2.5 加密



4.2.1 用户标识和鉴别

- 定义
  - 系统提供一定的方式让用户标识自己的名字和身份,系统进行核实,通过鉴定后才提供系统使用权.

- 常用方法
  - 用户标识证实
  - 过程识别
  - 上机密码卡
  - 指纹、声音、照片等识别
  - 回答问题

4.2.2 存取控制

- 1. 存取控制概述
  - 对于获得上机权的用户还要根据系统预先定义好的外模式（视图）或用户权限进行存取控制，保证用户只能存取他有权存取的数据。

- 2. 方法
  - 定义用户权限
  - 合法权限检查

4.2.2 存取控制

- 常用存取控制方法

  - 自主存取控制（Discretionary Access Control，简称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的权限，可转授用户存取权限。
    - C2级
    - 灵活

  - 强制存取控制（Mandatory Access Control，简称 MAC）：每一数据对象标以一定密级，每一用户对应某一级别的许可证，只有具有合法许可证的用户才可以存取某一对象。
    - B1级
    - 严格

4.2.2 存取控制

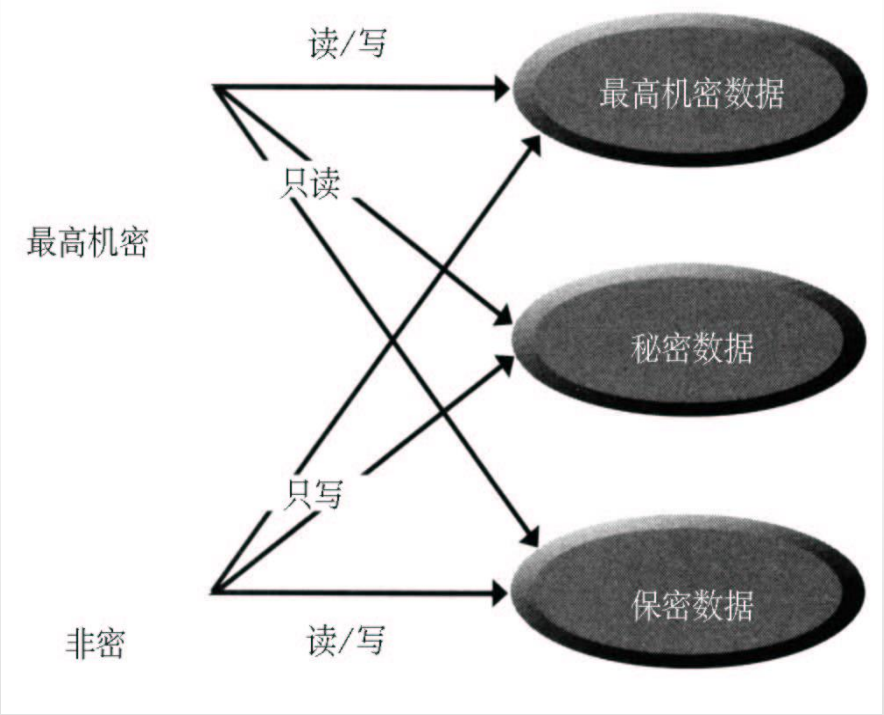
- 3. 自主存取控制(DAC)方法
  - 用户权限
    - 数据对象
    - 操作类型
  - 授权语句
    - GRANT
    - REVOKE

4.2.2 存取控制

- DAC不能实现元组级的授权
- 数据库角色：被命名的一组与数据库操作相关的权限
  - 角色是权限的集合
  - 可以为一组具有相同权限的用户创建一个角色
  - 简化授权的过程

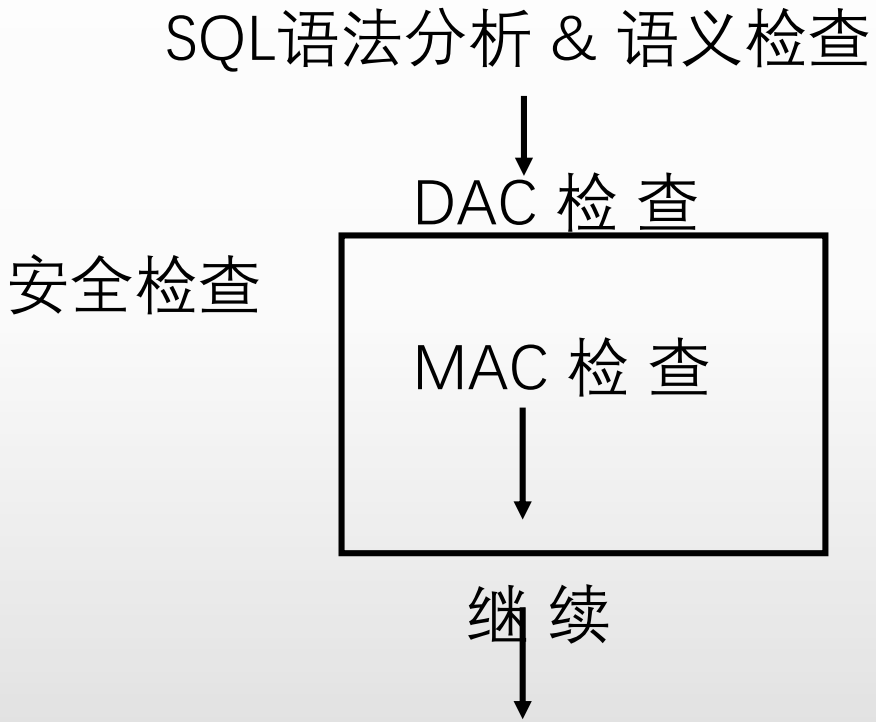
4.2.2 存取控制

- 4. 强制存取控制(MAC)方法
  - 实体类别
    - 主体
    - 客体
  - 敏感度标记
    - 绝密、机密、可信、公开
    - 许可证级别
    - 密级



强制存取控制方法（续）

DAC + MAC安全检查示意图



- 先进行DAC检查，通过DAC检查的数据对象再由系统进行MAC检查，只有通过MAC检查的数据对象方可存取。

4.2.3 视图机制

- 视图就象架设在用户与底层表之间的一道桥梁，用户只能对视图进行操作，而无法直接访问底层表。

CREATE VIEW VIEW－1
AS SELECT COLUMN－1 FROM TABLE－1
GRANT ALL ON VIEW－1 TO USER－2

- USER－2通过VIEW－1可以访问COLUMN－1而无法访问COLUMN－2，这就是VIEW－1的屏作用

4.2.4 审计

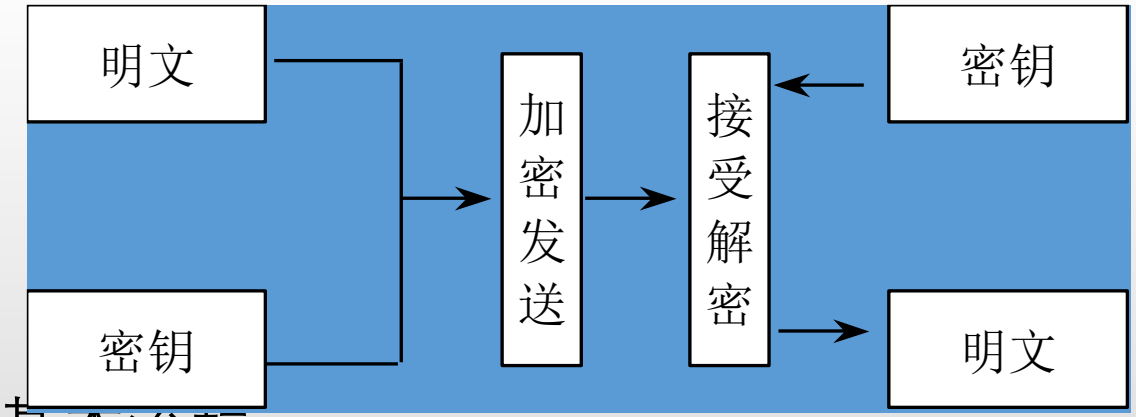
- 1. 功能
  - 设备安全审计
  - 操作审计
  - 应用审计
  - 攻击审计

- 2. 技术
  - 静态技术
  - 动态技术
  - 结果验证

4.2.5 加密

- 思想
  - 数据库中的数据以密码形式存放，使用时由用户设计的解码程序将其转换成用户可读的数据。这样，数据库中的数据即使被窃取，也只能是一些无法辨认的代码。

- 加密算法
  - DES
  - AES



- 数据库密码系统的基本流程

5.1 数据库完整性概述

- 数据库的完整性是指数据的正确性、有效性和相容性，防止错误的数据进入数据库，造成无效操作和错误结果。以至数据库的一致性遭到破坏。

- 系统用一定的机制来检查数据库中的数据是否满足规定的条件，即完整性约束条件。

- 数据的约束条件是语义的体现，完整性约束条件将作为模式的一部分存入数据库中。

数据库完整性机制在解决社会性突发问题起到的作用

- 近年来各类社会性问题频发，我们需要及时控制和妥善处理各类突发公共事件，提高快速反应和应急处理能力，建立健全应急机制，确保人们的生命安全。
- 以疫情期间在超市发现冻品表面新冠检测呈阳性为例

在设计超市数据库时，要考虑在若干关键表之间建立关联性

例如 会员表

会员卡编号	姓名	身份证号	电话号码	会员等级	。。。。

商品

商品编号	商品名	单价	数量	。。。。

销售单据

会员卡编号	商品编号	。。。。		

若某超市在售的某个冻品被发现表面新冠检测呈阳性。首先立即下架该商品，同时通过销售单据表找到买此类商品的会员卡编号，排查与此关联的会员表，找到具体的购物人。这些查找可通过定义上述三张表间的外码联系来实现。

疾控人员可及时地对经超市数据库排查出的密接人员进行核酸检测、隔离疑似人员等措施，尽可能减少对社会危害。

5.1.1 完整性约束条件

- 值的约束和结构的约束
  - 值的约束：数据的取值范围、数据类型
  - 数据之间联系的约束：函数依赖关系、实体完整性约束和参照完整性约束

- 静态约束和动态约束：
  - 静态：每一确定状态的数据应满足的约束条件
  - 动态：数据库从一种状态转变为另外一种状态时新、旧值应满足的约束条件。

5.1.1 完整性约束条件

- 立即执行约束和延迟执行约束
  - 立即执行：执行事务时，对某一更新语句执行完后马上对此数据所应满足的约束条件进行完整性检查。
  - 延迟执行：整个事务执行结束后方对此约束条件进行完整性检查，结果正确才能提交。

5.1.2 完整性控制

- 为了实现完整性控制,DBA应向DBMS提供一组完整性规则,这组规则规定用户在进行更新操作时,什么时候对数据检查,检查什么样的错误,出现错误如何处理。

- 完整性规则的组成
  - 规则的触发条件
  - 规则的约束条件
  - 违约响应

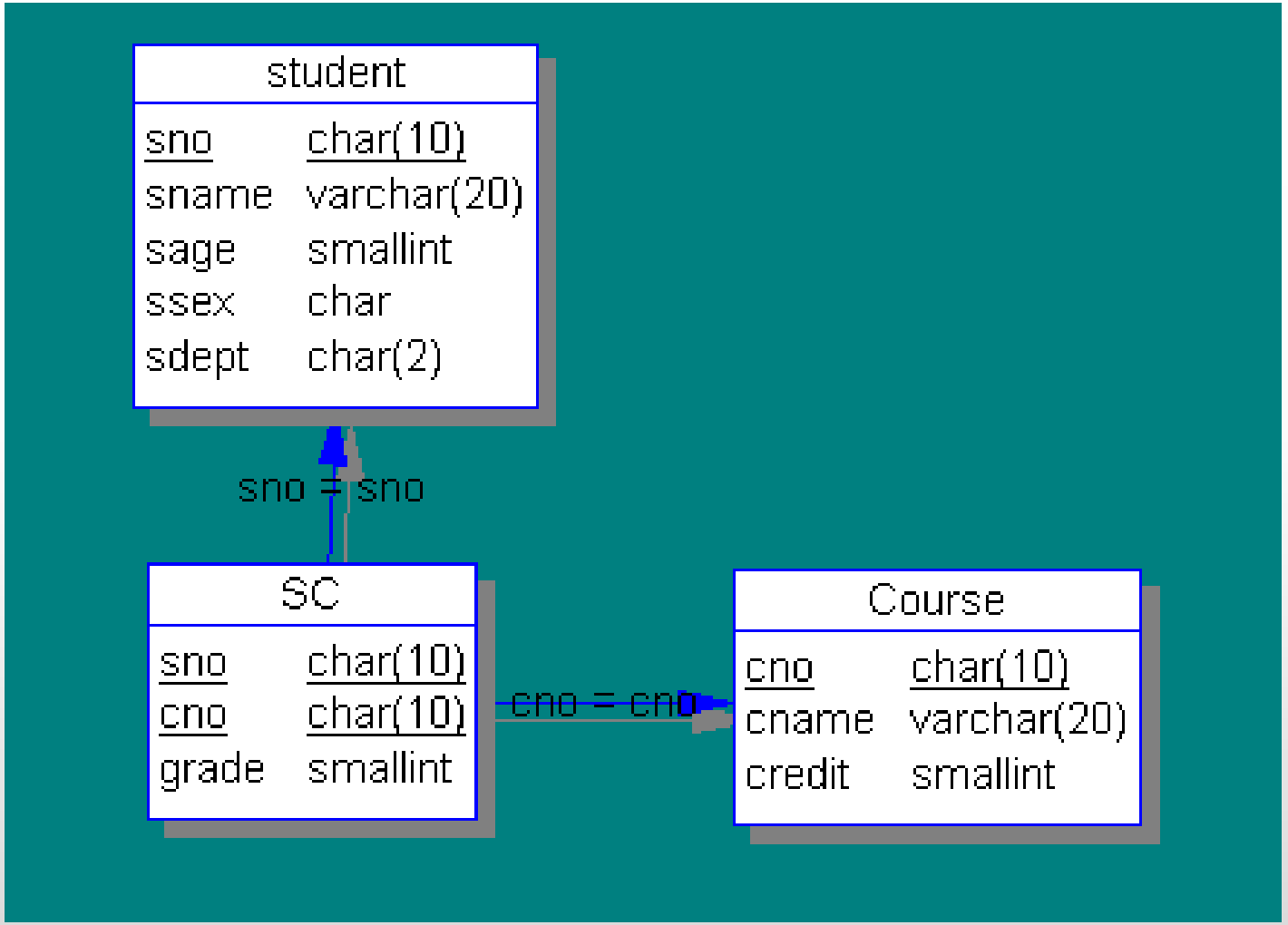
- DBMS提供语句表达完整性规则，且可以修改

5.1.2 完整性控制

- 1.DBMS提供语句表达完整性规则
  - 实体完整性
    - 候选码的属性不能取Null、也不能有重复值
    - 主码和候选吗
      - Primary Key
      - Unique
      - 一般在Primary Key上自动加上Index
      - 在Unique上的Index需另行声明

5.1.2 完整性控制

- 1.DBMS提供语句表达完整性规则
  - 参照完整性和外键
    - 参照&被参照、主表&从表、主键&外键
    - 外键的值不允许参照不存在的主键的值
    - 主键与外键的相容
      - 类型
      - 属性名可以不同
      - 外键允许Null


student
sno char(10)
sname varchar(20)
sage smallint
ssex char
sdept char(2)
sno = sno
SC
sno char(10)
cno char(10)
grade smallint
cno = cno
Course
cno char(10)
cname varchar(20)
credit smallint

5.1.2 完整性控制

- 1.DBMS提供语句表达完整性规则
  - 保持参照完整性
    - 各种可能的情况
      - 从表
        - 插入从表元组，且外键不为Null
        - 修改从表外键，且不为Null
      - 主表
        - 删除主表元组，其已被参照
        - 修改主表主键，其已被参照
        - Drop Table

5.1.2 完整性控制

- 1.DBMS提供语句表达完整性规则
  - 用户自定义完整性
    - 对属性值的约束
      - unique
      - not null
        - Primary Key约束隐含Not Null
        - 不加约束 隐含允许Null
      - Check
        - 在定义属性时约束该属性必须使Check 为True

用户自定义完整性

- 对元组的约束

当学生的性别是男时，其名字不能以Ms.打头。

```
CREATE TABLE Student
    (Sno    CHAR(9),
     Sname  CHAR(8) NOT NULL,
     Ssex   CHAR(2),
     Sage   SMALLINT,
     Sdept  CHAR(20),
     PRIMARY KEY (Sno),
     CHECK (Ssex='女' OR Sname NOT LIKE 'Ms.%')
     /*定义了元组中Sname和 Ssex两个属性值之间的约束条件*/
    );
```

✓ 性别是女性的元组都能通过该项检查，因为Ssex='女'成立；

✓ 当性别是男性时，要通过检查则名字一定不能以Ms.打头

5.1.2 完整性控制-域的完整性

- SQL支持域的概念，并可以用CREATE DOMAIN语句建立一个域以及该域应该满足的完整性约束条件。

［例］建立一个性别域，并声明性别域的取值范围

```
CREATE DOMAIN GenderDomain CHAR(2)
CHECK (VALUE IN ('男'，'女') );
```

这样［例10］中对Ssex的说明可以改写为

```
Ssex GenderDomain
```

［例］建立一个性别域GenderDomain，并对其中的限制命名

```
CREATE DOMAIN GenderDomain CHAR(2)
CONSTRAINT GD CHECK ( VALUE IN ('男'，'女') );
```

5.1.2 完整性控制-域的完整性

［例］ 删除域GenderDomain的限制条件GD。

```
ALTER  DOMAIN  GenderDomain
DROP CONSTRAINT GD;
```

［例］ 在域GenderDomain上增加限制条件GDD。

```
ALTER  DOMAIN  GenderDomain
ADD CONSTRAINT GDD CHECK (VALUE IN ( '1',  '0') );
```

✓通过上两例，就把性别的取值范围由('男'，'女')改为 ('1'，'0')

5.1.2 完整性控制

- 1.DBMS提供语句表达完整性规则
  - 用户自定义完整性
    - 触发器
      - Trigger则是基于对表的操作（动作）的
      - 当指定的表上发生特定的操作，系统便激活Trigger程序
      - 大部分DBMS产品均支持Trigger

5.1.2 完整性控制-触发器

- 触发器（Trigger）是用户定义在关系表上的一类由事件驱动的特殊过程
  - 由服务器自动激活
  - 可以进行更为复杂的检查和操作，具有更精细和更强大的数据控制能力

5.1.2 完整性控制-触发器

- 定义触发器
- 激活触发器
- 删除触发器

5.1.2 完整性控制-定义触发器

- **CREATE TRIGGER语法格式**

```
CREATE TRIGGER <触发器名>
  {BEFORE | AFTER} <触发事件> ON <表名>
   FOR EACH  {ROW | STATEMENT}
   [WHEN <触发条件>]
   <触发动作体>
```

5.1.2 完整性控制-定义触发器

- 定义触发器的语法说明:
  - 1. 创建者：表的拥有者
  - 2. 触发器名
  - 3. 表名：触发器的目标表
  - 4. 触发事件：INSERT、DELETE、UPDATE
  - 5. 触发器类型
    - 行级触发器（FOR EACH ROW）
    - 语句级触发器（FOR EACH STATEMENT）

5.1.2 完整性控制-定义触发器

- 例如,假设在TEACHER表上创建了一个AFTER UPDATE触发器。如果表TEACHER有1000行，执行如下语句：

  UPDATE TEACHER SET Deptno=5;

  - 如果该触发器为语句级触发器，那么执行完该语句后，触发动作只发生一次
  - 如果是行级触发器，触发动作将执行1000次

5.1.2 完整性控制-定义触发器

- 6. 触发条件
  - 触发条件为真
  - 省略WHEN触发条件
- 7. 触发动作体
  - 触发动作体可以是一个匿名SQL过程块
  - 也可以是对已创建存储过程的调用

5.1.2 完整性控制-定义触发器

[例] 定义一个BEFORE行级触发器，为教师表Teacher定义完整性规则“教授的工资不得低于4000元，如果低于4000元，自动改为4000元”。

```
CREATE TRIGGER Insert_Or_Update_Sal
    BEFORE INSERT OR UPDATE ON Teacher
/*触发事件是插入或更新操作*/
    FOR EACH ROW                          /*行级触发器*/
AS BEGIN                                  /*定义触发动作体，是PL/SQL过程块*/
      IF (new.Job='教授') AND (new.Sal < 4000) THEN
      new.Sal :=4000;
      END IF;
END;
```

5.1.2 完整性控制-定义触发器

[例] 定义AFTER行级触发器，当教师表Teacher的工资发生变化后就自动在工资变化表Sal_log中增加一条相应记录

首先建立工资变化表Sal_log

```
CREATE TABLE Sal_log
    (Eno    NUMERIC(4)  references teacher(eno),
     Sal     NUMERIC(7,  2),
     Username  char(10),
     Date   TIMESTAMP
    );
```

5.1.2 完整性控制-定义触发器

[例]（续）

```
CREATE TRIGGER Insert_Sal
    AFTER INSERT ON Teacher              /*触发事件是INSERT*/
    FOR EACH ROW
    AS BEGIN
        INSERT INTO Sal_log VALUES(
            new.Eno，  new.Sal，  CURRENT_USER，  CURRENT_TIMESTAMP);
    END;
```

5.1.2 完整性控制-定义触发器

[例]（续）

```
CREATE TRIGGER Update_Sal
   AFTER UPDATE ON Teacher            /*触发事件是UPDATE */
   FOR EACH ROW
   AS BEGIN
     IF (new.Sal <> old.Sal) THEN INSERT INTO Sal_log VALUES(
          new.Eno，new.Sal，CURRENT_USER，CURRENT_TIMESTAMP);
     END IF;
  END;
```

5.1.2 完整性控制-激活触发器

- 触发器的执行，是由触发事件激活的，并由数据库服务器自动执行
- 一个数据表上可能定义了多个触发器
  - 同一个表上的多个触发器激活时遵循如下的执行顺序:
    - （1） 执行该表上的BEFORE触发器；
    - （2） 激活触发器的SQL语句；
    - （3） 执行该表上的AFTER触发器。

5.1.2 完整性控制-激活触发器

[例20] 执行修改某个教师工资的SQL语句，激活上述定义的触发器。

```
UPDATE Teacher SET Sal=800 WHERE Ename='陈平';
```

执行顺序是：

- 执行触发器Insert_Or_Update_Sal
- 执行SQL语句“UPDATE Teacher SET Sal=800 WHERE Ename='陈平';”
- 执行触发器Insert_Sal;
- 执行触发器Update_Sal

5.1.2 完整性控制-删除触发器

- 删除触发器的SQL语法：

  DROP TRIGGER <触发器名> ON <表名>;

- 触发器必须是一个已经创建的触发器，并且只能由具有相应权限的用户删除。

[例21]　删除教师表Teacher上的触发器Insert_Sal

DROP TRIGGER Insert_Sal ON Teacher;

激活的时机

事件

名称

新值/修改后的值
旧值/修改前的值

执行程序的条件

```
Create Trigger NetWorthTrigger
After Update Of netWorth On MovieExec
referencing
  Old As OldTuple,New As NewTuple
When(OldTuple.netWorth > NewTuple.netWorth)
  Update MovieExec
  Set netWorth = OldTuple.netWorth
  Where cert# = NewTuple.cert#
For Each Row
```

每修改一个元组便激活

激活的时机

事件

名称

新值/修改后的值
旧值/修改前的值

执行程序的条件

```
Create Trigger NetWorthTrigger
After Update Of netWorth On MovieExec
referencing
  Old As OldTuple,New As NewTuple
When(OldTuple.netWorth > NewTuple.netWorth)
  Update MovieExec
  Set netWorth = OldTuple.netWorth
  Where cert# = NewTuple.cert#
For Each Row
```

每修改一个元组便激活

小结

- 数据库保护
  - 为了保证数据的安全可靠和正确有效,DBMS必须提供统一的数据保护功能.(数据控制)

- 机制
  - 安全性
  - 完整性
  - 并发控制
  - 数据库恢复