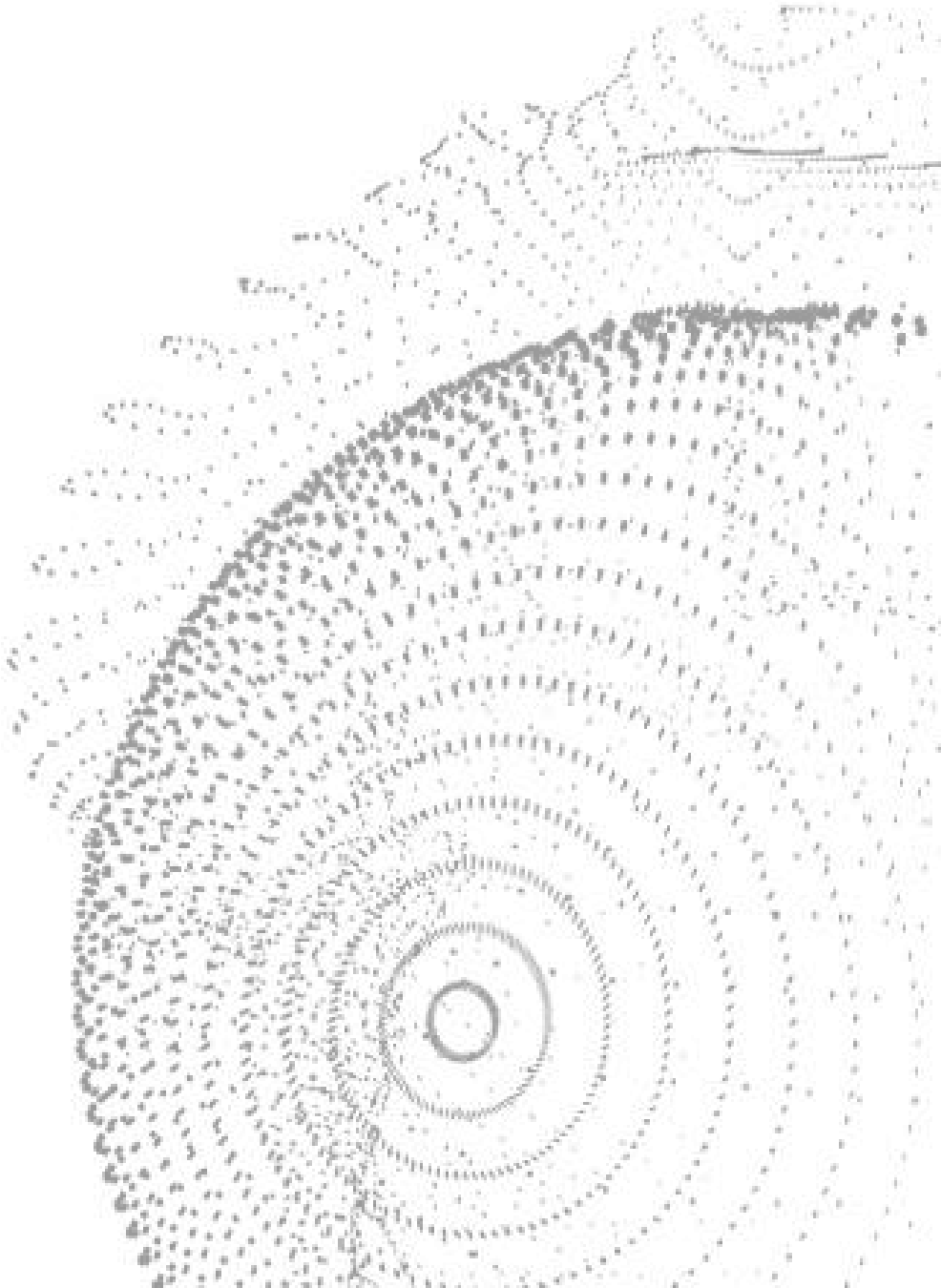
Chapter 04

数据库的安全性与完整性保护

The Protection of Database Security and Integrity



为什么要提供数据库的安全性保护和完整性保护功能?

计算机系统外部的

- 环境的保护
- 社会的保护
- 设备的保护

计算机系统内部的

- 计算机
- 操作系统
- 数据库

其他

- 网络
- 应用程序

在上述这些保护措施中，数据库系统中的数据保护是至关重要的。在本章中我们主要讨论***在数据库管理系统内部设置一些必要的软件以达到数据保护的目的，这就是数据库的保护***

数据库管理系统提供的保护措施主要有两类：

01 数据库的安全性（security）保护

02 数据库的完整性（integrity）保护

4.1 数据库的安全性

-01 数据库的安全与安全数据库

-02 数据库安全的基本概念与内容

-03 数据库的安全标准

-04 SQL对数据库安全的支持

4.1.1 数据库的安全与安全数据库

- **数据库的安全（database security）**

防止非法使用数据库。即要求数据库的用户

- 通过规定的访问途径
- 按照规定的访问规则（规范）

来访问数据库中的数据，并接受来自DBMS的各种检查

- **访问数据库的规范有多种，但是：**

- 不同的规范适用于不同的应用
- 可以根据应用的不同需求来选择不同的数据库访问规范，即选择具有不同安全级别的数据库
- 那些能适应网络环境下安全要求级别的数据库称为安全**数据库（secure database）**，或称为**可信数据库（trusted database）**

4.1.2 SQL对数据库安全的支持

在SQL’92中提供了C1级数据库安全的支持，它们是：

主体、客体及主/客体分离

身份标识与鉴别

数据完整性

自主访问控制与授权功能

4.1.2 SQL对数据库安全的支持

自主访问控制与授权功能

- SQL中的自主访问控制是通过（用户，操作对象，操作权限）这样的三元组来定义用户对于数据的访问权限的，并可通过授权（Grant）和回收（Revoke）语句来改变用户的访问权限

- 操作权限

SELECT权	***INSERT权***	***DELETE权***	***UPDATE权***
REFERENCE权	***EXECUTE权***	***USAGE权***	

4.1.2 SQL对数据库安全的支持

- **操作对象**

- 表
- 视图
- 属性 域（type），UDT（用户定义数据类型）
- 存储过程/函数，触发器

- **用户**

- 数据库用户

4.1.2 SQL对数据库安全的支持

授权语句

GRANT <操作权限列表> ON <操作对象> TO <用户名列表> [WITH GRANT OPTION]

-例:

```
grant SELECT, UPDATE on S to XULIN with grant
option
grant UPDATE(G) on SC to XULIN
```

4.1.2 SQL对数据库安全的支持

回收语句

REVOKE <操作权限列表> ON <操作对象> FROM <用户名列表> [RESTRICT | CASCADE]

- CASCADE：连锁回收
- RESTRICT：在不存在连锁回收问题时才能回收权限，否则拒绝回收

-例：

```
revoke UPDATE on S from XULIN cascade
```

4.2 数据库的完整性

-01 数据库完整性保护的功能

-02 完整性规则的三个内容

-03 完整性约束的设置、检查与处理

-04 触发器

4.2 数据库的完整性

- **指数据库中数据的正确性和一致性，包括：**

- **正确性：**数据的有效性、有意义
- **一致性：**在多用户（多程序）并发访问数据库的情况下，保证对数据的更新不会出现与实际不一致的情况

- **我们一方面要避免在数据库中出现错误的数据，即防止数据的完整性受到破坏。同时，如果因为某些不可抗拒的原因而导致数据库中的数据被破坏，也要能够及时发现并采取一定的措施将数据库中的数据恢复到正确的状态下去**

4.2 数据库的完整性

- **完整性保护**

对数据库中数据的正确性和一致性的维护，包括：

- 在执行更新操作时，检查是否违反完整性约束条件，并且在证明其无效后作出适当的反应
- 防止有存取权的合法用户的误操作

- **完整性保护的目的**

- 及时发现错误
- 能够采取措施防止错误的进一步蔓延
- 最终将数据库恢复到正确状态

- **完整性保护的实现措施**

- 完整性约束条件的定义及检查
- 触发器
- 并发控制技术

4.2.1 数据库完整性保护的功能

三个基本功能

设置功能

- 系统及用户对数据库完整性的基本要求

检查功能

- 有能力检查数据库中的数据是否有违反约束条件的现象出现

处理功能

- 出现违反约束条件时，有及时处理的能力

4.2.2 完整性规则的三个内容

- **在关系数据库系统中提供了下述三类数据完整性约束：**

01 实体完整性规则

- 在一个基表的主关键字（主码）中，其属性的取值不能为空值

02 参照完整性规则

03 用户定义的完整性规则

-由用户来定义的数据完整性要求

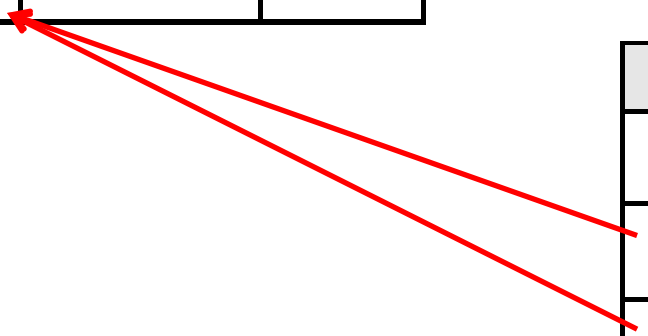
- **由数据库管理系统来提供数据完整性约束条件的定义和检查的优点？**

学 生 关 系

学　号	姓　名	系　别	年 龄
…	…	…	…
993501	刘 英	计 算 机	18
…	…	…	…

选 课 关 系

学　号	课 程 号	成 绩
…	…	…
993501	CS101	85
993501	EN103	90
…	…	…



职 工 关 系

工作证号	姓名	职务	年龄	工资	上级领导的工作证号
E012	李 强	车间主任	53	1200	E001
E025	王 英	技 工	27	800	E012
E001	张伟一	厂 长	49	3000	NULL

其中：‘工作证号’是主码，‘上级领导的工作证号’是外码

外键完整性约束示例

学 生 关 系

学　号	姓　名	系　别	年 龄
…	…	…	…
993501	刘 英	计 算 机	18
…	…	…	…

选 课 关 系

学　号	课 程 号	成 绩
…	…	…
993501	CS101	85
993501	EN103	90
…	…	…

对‘选课’关系中的外键‘学号’的引用完整性约束定义：

Foreign key(学号) References 学生 On Delete **Cascade**;
Foreign key(学号) References 学生 On Delete **Restrict**;
Foreign key(学号) References 学生 On Delete **Set Null**;
Foreign key(学号) References 学生 On Delete **No Action**;

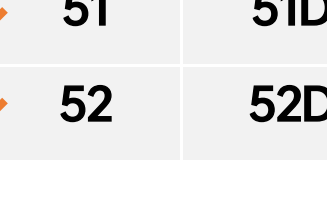
外键完整性约束

“restrict”和“no action”的区别

PAR

PID	NAME
1	PAR1
2	PAR2
3	PAR3

CID	CDESC	*PID*
51	51DESC	*1*
52	52DESC	*2*



UPDATE PAR SET PID=PID-1

- “restrict”报错
- “no action”可以执行

4.2.3 完整性约束的设置、检查与处理

一条完整性约束规则一般有三个组成部分

- **完整性约束条件的设置**
- **完整性约束条件的检查**

  在DBMS内部设置专门的软件检查模块
- **完整性约束条件的处理**

  在用户的操作会破坏数据的完整性（即违反完整性约束条件的要求）时，系统将：
  - 拒绝执行，并报警或报错；
  - 调用相应的函数（例程）进行处理，如：
    - 在外键定义子句中给出的处理方法
    - 在触发器中给出的处理过程

4.2.3 完整性约束的设置、检查与处理

完整性约束条件设置

- **属性级的约束（域约束）**
  数据类型的约束，非空值约束，取值范围的约束
- **元组级的约束（表约束）**
  主码定义，候选码（唯一键）定义
  外码定义
  基于元组的检查子句：属性间关系的定义
- **全局约束（断言assertion）**
  单个关系中涉及到统计操作的约束条件
  多个关系之间复杂的约束条件

对约束命名

CONSTRAINT <约束名> <完整性约束定义子句>

4.2.3 SQL语言对完整性约束规则的支持

创建基表命令 （CREATE TABLE）

需要定义的内容

- 模式名 & 表名
- 属性的定义
  - 属性名 & 数据类型
  - 属性的缺省值定义 **DEFAULT { default_constant | NULL }**
  - 属性级的数据约束定义
- 表级的数据约束定义

4.2.3 SQL语言对完整性约束规则的支持

属性级的约束

数据类型的约束，取值范围的约束，非空值约束

```
{    NOT NULL    |
     [ CONSTRAINT constraint_name ]
           UNIQUE
         | PRIMARY KEY
         | CHECK ( search_condition )
         | REFERENCES table_name [ ( column_name ) ]
         [ ON DELETE CASCADE | RESTRICT | SET NULL ]
         [ ON UPDATE CASCADE | RESTRICT | SET NULL ]    }
```

基于单个属性的取值约束

4.2.3 SQL语言对完整性约束规则的支持

```
[ CONSTRAINT constraint_name ]
    {    UNIQUE ( colname { , colname ... } )
         | PRIMARY KEY ( colname { , colname ... } )
         | CHECK ( search_condition )
         | FOREIGN KEY ( colname { , colname ... } )
         REFERENCES table_name [ ( colname { , colname ... } ) ]
                  [ ON DELETE CASCADE | RESTRICT | SET NULL ]
                  [ ON UPDATE CASCADE | RESTRICT | SET NULL ]    }
```

基于单个属性的取值约束

4.2.3 SQL语言对完整性约束规则的支持

在定义属性级的数据约束时需要考虑的内容

- **NOT NULL vs. DEFAULT NULL**
- **Constraint name**

  对某个数据约束条件进行命名（可选项）

  以利于以后使用ALTER TABLE命令来修改表中的数据约束定义
- **UNIQUE vs. NOT NULL**

  UNIQUE属性可以取空值

  候选键（candidate key）：**UNIQUE + NOT NULL**

4.2.3 SQL语言对完整性约束规则的支持

在定义属性级的数据约束时需要考虑的内容（cont.）

- **PRIMARY KEY vs. NOT NULL**
- **REFERENCES**
  - FOREIGN KEY（外键） vs. PRIMARY KEY（主键）
  - 外键上的取值约束及其一致性的保证措施

    CASCADE | RESTRICT | SET NULL
- **CHECK**

4.2.3 SQL语言对完整性约束规则的支持

元组级的约束

- **主码定义：PRIMARY KEY(<column-list>)**
- **唯一键定义：UNIQUE(<column-list>)**
- **外码定义：**

```
FOREIGN KEY ( <fk-column-list> )
REFERENCES <table-name> ( <pk-column-list> )
[ ON UPDATE [ RESTRICT | CASCADE | SET NULL ] ]
[ ON DELETE [ RESTRICT | CASCADE | SET NULL ] ]
```

- **属性间关系的定义：CHECK(<condition>)**

4.2.3 SQL语言对完整性约束规则的支持

定义属性级数据约束的例子

```
CREATE  TABLE  Student (
    sno  NUMBER(5)
        CONSTRAINT C1 CHECK (sno  BETWEEN  90000 AND  99999),
    sname  VARCHAR (20)
        CONSTRAINT  C2  NOT  NULL,
    sage  NUMBER(3)
        CONSTRAINT  C3  CHECK (sage<29) );
```

4.2.3 SQL语言对完整性约束规则的支持

例子

```
CREATE  TABLE  EMP  (
        Empno   NUMBER (4),
        Ename   VARCHAR (10),
        PersonId    VARCHAR(15),
        Job         VARCHAR (9),
        Mgr         NUMBER (4),
        Sal         NUMBER (7, 2),
        Deptno    NUMBER (2),

        CONSTRAINT  pk  PRIMARY  KEY( Empno ),

        CONSTRAINT  uni_name  UNIQUE ( PersonId ),
```

4.2.3 SQL语言对完整性约束规则的支持

```
CONSTRAINT  fk_mgr
FOREIGN  KEY( Mgr )
REFERENCES  EMP( Empno )
                    ON  UPDATE  CASCADE
                    ON  DELETE  SET  NULL,
```

4.2.3 SQL语言对完整性约束规则的支持

```
CONSTRAINT  fk_mgr
FOREIGN  KEY( Mgr )
REFERENCES  EMP( Empno )
                    ON  UPDATE  CASCADE
                    ON  DELETE  SET  NULL,
```

4.2.3 SQL语言对完整性约束规则的支持

```
CONSTRAINT  chk_1  CHECK (
(Job='总经理' AND Sal>15000)  OR
(Job='部门经理'  AND  Sal between 5000 and 10000)  OR
(Job<>'总经理' AND Job<>'部门经理'  AND
                                    Sal between 1000 and 5000)
)
);
```

4.2.3 SQL语言对完整性约束规则的支持

例如：不允许‘智科院’（IS）的学生选修‘PASCAL程序设计’（PAS）课程，则在创建‘学习’关系时可以定义一个完整性约束：

```
CHECK ( NOT (
(S#  IN  (SELECT  S#
      FROM  S
      WHERE  sd='IS') )
AND
(C#  IN  (SELECT  C#
      FROM  C
      WHERE  cn='PAS') )
) );
```

？问题：

上述完整性约束只对定义它的‘学习’关系起作用，而对‘学生’关系则不起作用。如果一个正在选修PASCAL课程的学生转到计算机系（将‘系别’属性的值修改为‘计算机’），这无疑会破坏‘学习’关系的数据完整性

4.2.3 SQL语言对完整性约束规则的支持

全局约束：断言

- **定义断言**

  CREATE ASSERTION <name> CHECK(<condition>)

- **CHECK**

  DROP ASSERTION <assertion-name-list>

4.2.3 SQL语言对完整性约束规则的支持

例：每门课程的选修人数不少于20人

```
CREATE ASSERTION ass_1 CHECK (
        20 <= ALL (SELECT COUNT(*)
                   FROM SC
                   GROUP BY C# )
);
```

4.2.3 SQL语言对完整性约束规则的支持

例：学生在选修‘数据结构’（DS）课程之前必需先选修过‘PASCAL程序设计’（PAS）课

```
CREATE ASSERTION ass_2 CHECK (
    NOT EXISTS (
        SELECT * FROM SC WHERE
            C# IN (SELECT C#
            FROM C
            WHERE cn = 'DS')
            AND
            S# NOT IN (SELECT SCX.S#
                FROM SC SCX, C
                WHERE SCX.C# = C.C#
                AND C.cn='PAS')
) );
```

4.2.4 触发器

- **在数据库系统中，一个事件的发生会导致另外一些事件的发生，这样的功能被称为触发器**

- **触发器的功能：某个事件的发生会导致另外一些事件的执行，以消除前一个事件对数据完整性所起的影响**

- **触发器最初是用于数据的完整性保护，但现在已经远远超出了此范围，也被应用于其它的目的，如：**
  - 数据的安全性保护
  - 用户的应用逻辑处理
  - 数据库系统的主动功能

4.2.4 触发器

- **触发器的组成**

- **触发事件（由用户定义）**
  - 通常为某个完整性约束条件的否定或某种数据操纵事件
  - 如：用户登录，数据的增、删、改等
- **结果事件（由用户定义）**
  - 当触发事件发生时，用以消除触发事件所引起的负面影响的程序
  - 通常是一组由用户书写的SQL命令
- **触发过程**
  - 当DBMS检测到触发事件的发生时，自动调用并执行结果事件的过程

4.2.4 触发器

触发器的定义命令

```
CREATE TRIGGER trigger_name { BEFORE | AFTER }
      { INSERT | DELETE
        | UPDATE [ OF colname { , colname ... } ] }
          ON table_name
      [ REFERENCING corr_name_def { , ..... } ]
      [ FOR EACH ROW | FOR EACH STATEMENT ]
      [ WHEN ( search_condition ) ]
            { statement
            | BEGIN ATOMIC statement; { statement; ... } END
```

4.2.4 触发器

The corr_name_def that defines a correlation name follows:

```
{      OLD [ ROW ] [ AS ] old_row_corr_name
| NEW [ ROW ] [ AS ] new_row_corr_name

| OLD TABLE [ AS ] old_table_corr_name
| NEW TABLE [ AS ] new_table_corr_name    }
```

4.2.4 触发器

触发器的删除命令

```
DROP TRIGGER trigger_name ;
```

4.2.4 触发器

例4.12

触发事件：修改或增添教师的工资和职称
结果事件：在插入新的教师元组或教师的职称晋升为教授时，若教授工资低于1000元，则将其自动转为1000元

```
CREATE  TRIGGER  update_sal
BEFORE  INSERT  or  UPDATE(Sal, Pos)  ON Teach
FOR  EACH  ROW
WHEN  (:new.Pos ='教授') /*某教师的职称为教授*/
BEGIN  IF        :new.sal < 1000
          THEN  :new.sal := 1000;
END IF;
END;
```