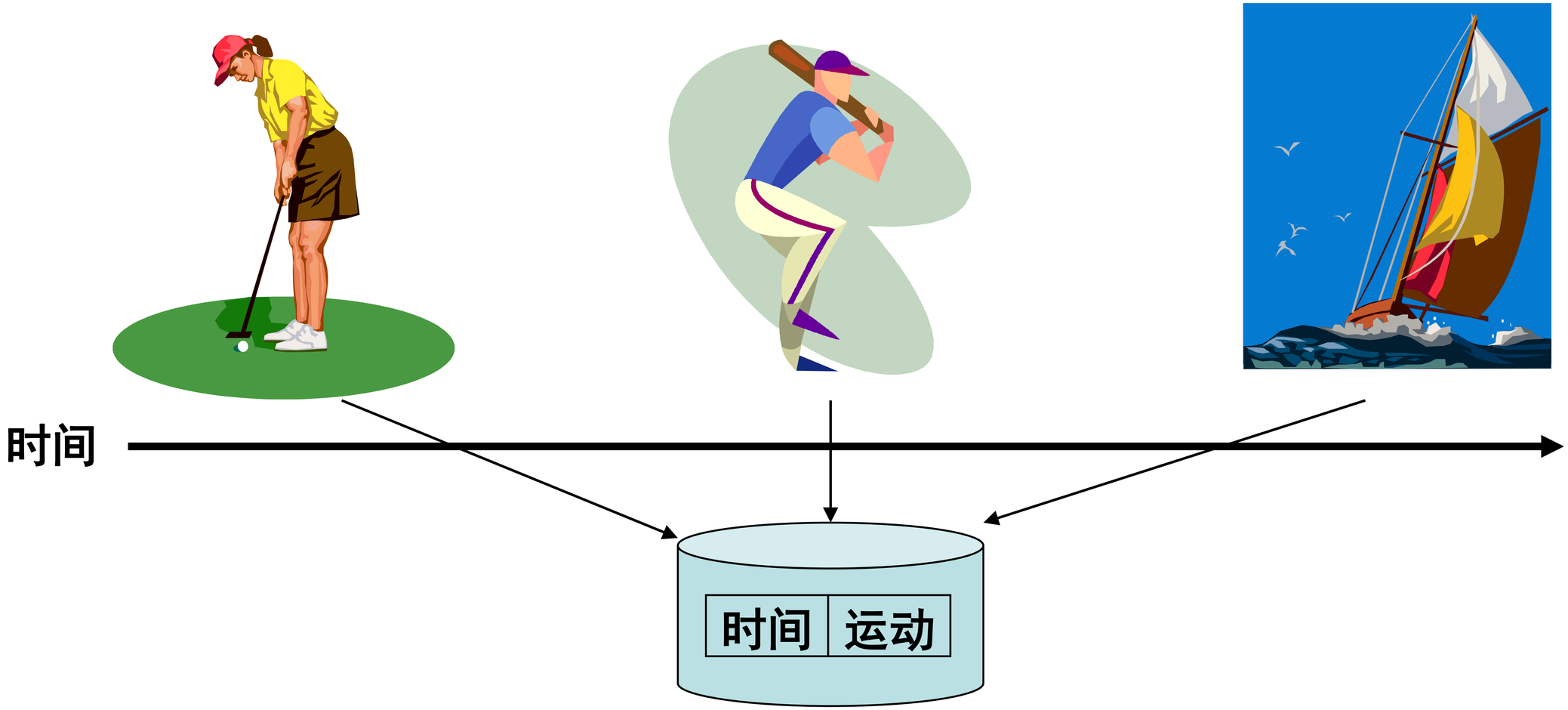
第五章 数据库完整性

从发现问题开始

- 数据库是现实世界状态的正确的反映.、
- 那么如何确保数据库能正确反映现实世界呢?



可能造成数据失真的因素

(1) 数据进入系统时的错误

– 与现实世界状态不符合的、不正确的数据

(2) 系统故障

– 丢失了数据

(3) 相互干扰

– 被他人覆写了

(4) 恶意破坏

– 非法入侵与修改

分析和观察

有哪些具体的违反现实世界规则的情况？
（以学生选课数据库为例进行讨论）

(1) 学生的性别只能取“男”或者“女”
(2) 成绩应在0到100分之间
(3) 学生学号应唯一
(4) 学生所选课程应是存在的课程
(5) 学生的名字不能取“赵C”
(6) ……

分析和观察

- 这些要求可以进一步分类
  - 关系模型要求的约束：实体完整性，参照完整性，数据类型等
  - 用户定义的约束：可以按照约束影响的范围分为：单一属性上的，单一元组上的，单一表上的，多个表之间的；范围越小，维护的代价越小

对问题进行抽象和定义

- 对于一个关系模式R，以及一组从现实世界中抽象出来的约束条件F
- 将F称为“完整性约束条件”，也称为“完整性规则”，是数据库中的数据必须满足的语义约束条件

- 对于任意的一个具体关系r ∈ R, 保证 F(r) 为真

如何进行完整性维护系统的设计?

三件事情:

1 如何表达完整性约束条件?

2 如何检查完整性约束条件?

3 违反条件的时候如何处理?

数据库完整性

1.提供定义完整性约束条件的机制

- SQL标准使用了一系列概念来描述完整性，包括关系模型的实体完整性、参照完整性和用户定义完整性
- 这些完整性一般由SQL的DDL语句来实现

数据库完整性(续)

2.提供完整性检查的方法

- 引起数据库状态改变的操作有哪些？INSERT、UPDATE、DELETE语句
- 执行上述操作后开始检查

3.违约处理

一致性状态 ⇨ 一致性状态

- DBMS若发现用户的操作违背了完整性约束条件，就采取一定的动作
  - 拒绝(NO ACTION)执行该操作
  - 级联(CASCADE)执行其他操作
  - 其他用户定义的操作

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名子句

5.5 域中的完整性限制

5.6 小结

5.1 实体完整性

5.1.1 实体完整性定义

5.1.2 实体完整性检查和违约处理

5.1.1 实体完整性定义

- 关系模型的实体完整性
  - CREATE TABLE中用PRIMARY KEY定义
- 单属性构成的码有两种说明方法
  - 定义为列级约束条件
  - 定义为表级约束条件
- 对多个属性构成的码只有一种说明方法
  - 定义为表级约束条件

实体完整性定义(续)

［例1］ 将Student表中的Sno属性定义为码

(1) 在列级定义主码

```
CREATE TABLE Student
(Sno  CHAR(9)  PRIMARY KEY,
Sname  CHAR(20) NOT NULL,
Ssex  CHAR(2) ,
Sage  SMALLINT,
Sdept  CHAR(20));
```

实体完整性定义(续)

(2)在表级定义主码

```
CREATE TABLE Student
    (Sno  CHAR(9),
     Sname  CHAR(20) NOT NULL,
     Ssex  CHAR(2) ,
     Sage  SMALLINT,
     Sdept  CHAR(20),
     PRIMARY KEY (Sno)
);
```

实体完整性定义(续)

［例2］将SC表中的Sno，Cno属性组定义为码

```
CREATE TABLE SC
 (Sno   CHAR(9)  NOT NULL,
  Cno  CHAR(4)  NOT NULL,
Grade    SMALLINT,
PRIMARY KEY (Sno，Cno)/*只能在表级定义主码*/
);
```

5.1.2 实体完整性检查和违约处理

- 插入或对主码列进行更新操作时，RDBMS按照实体完整性规则自动进行检查。包括：
  - 检查主码值是否唯一，如果不唯一则拒绝插入或修改
  - 检查主码的各个属性是否为空，只要有一个为空就拒绝插入或修改

- **检查记录中主码值是否唯一的一种方法是进行全表扫描**

待插入记录

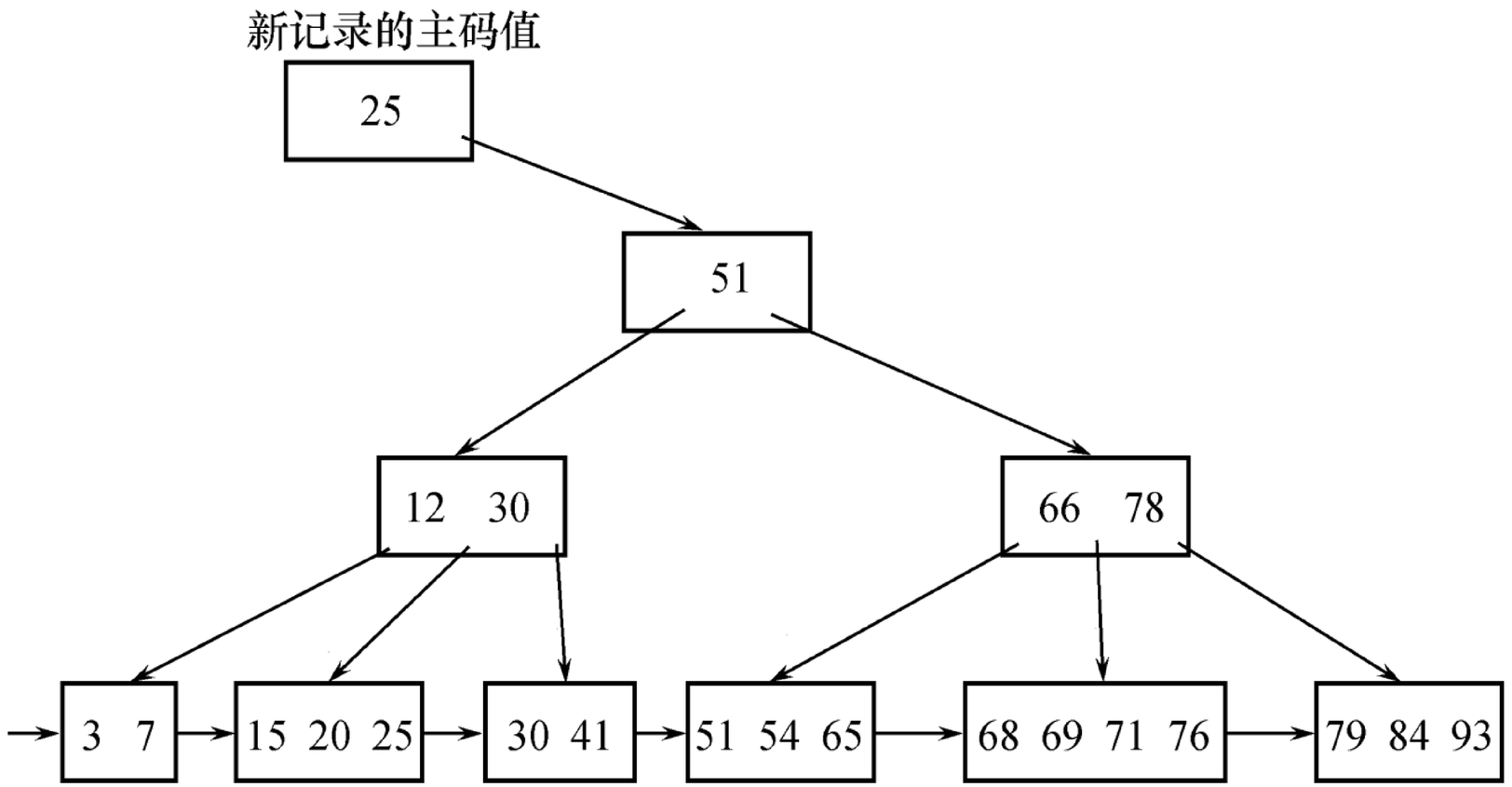
Keyi	F2i	F3i	F4i	F5i

基本表

Key1	F21	F31	F41	F51
Key2	F22	F32	F42	F52
Key3	F23	F33	F43	F53
⋮				

全表扫描缺点：十分耗时
为避免对基本表进行全表扫描，RDBMS一般都在主码上自动建立一个索引

通过B+树索引查找基本表中是否已经存在新的主码值，可以提高效率



如果新插入记录的主码值是25:

- 通过主码索引从B+树的根结点开始查找

- 读取3个结点：

- **根结点（51）**
- **中间结点（12 30）**
- **叶结点（15 20 25）**

- 该主码值已经存在，不能插入这条记录

5.2 参照完整性

5.2.1 参照完整性定义

5.2.2 参照完整性检查和违约处理

5.2.1 参照完整性定义

- 关系模型的参照完整性定义
  - 在CREATE TABLE中用FOREIGN KEY短语定义哪些列为外码
  - 用REFERENCES短语指明这些外码参照哪些表的主码

例如，关系SC中一个元组表示一个学生选修的某门课程的成绩，(Sno，Cno)是主码。Sno，Cno分别参照引用Student表的主码和Course表的主码

［例3］定义SC中的参照完整性

```
CREATE TABLE SC
  (Sno    CHAR(9)  NOT NULL,
   Cno    CHAR(4)  NOT NULL,
   Grade SMALLINT,
   PRIMARY KEY (Sno,   Cno),
   /*在表级定义实体完整性*/
   FOREIGN KEY (Sno) REFERENCES Student(Sno),
   /*在表级定义参照完整性*/
   FOREIGN KEY (Cno) REFERENCES Course(Cno)
   /*在表级定义参照完整性*/
);
```

对表SC和Student有四种可能破坏参照完整性的情况：

- SC表中增加一个元组，该元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等。
- 修改SC表中的一个元组，修改后该元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等。
- 从Student表中删除一个元组，造成SC表中某些元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等。
- 修改Student表中一个元组的Sno属性，造成SC表中某些元组的Sno属性的值在表Student中找不到一个元组，其Sno属性的值与之相等

参照完整性检查和违约处理

可能破坏参照完整性的情况及违约处理

被参照表 （例如Student）	参照表 （例如SC）	违约处理
可能破坏参照完整性	⇦ 插入元组	拒绝
可能破坏参照完整性	⇦ 修改外码值	拒绝
删除元组 ⇨	可能破坏参照完整性	拒绝/级连删除/设置为空值
修改主码值 ⇨	可能破坏参照完整性	拒绝/级连修改/设置为空值

违约处理

- **参照完整性违约处理**

  1. 拒绝(NO ACTION)执行

  不允许该操作执行。一般设置为默认策略。

  2. 级联(CASCADE)操作

  当删除或修改被参照表(Student)的一个元组造成了与参照表(SC)的不一致，则删除或修改参照表中的所有造成不一致的元组。例如删除Student表中的元组，Sno值为200215121，则从SC表中级连删除 SC.Sno='200215121'的所有元组

违约处理

- 参照完整性违约处理

3. 设置为空值（SET-NULL）

当删除或修改被参照表的一个元组时造成了不一致，则将参照表中的所有造成不一致的元组的对应属性设置为空值。

违约处理(续)

3. 设置为空值（SET-NULL）（续）

外码

例如，有下面2个关系

学生（学号，姓名，性别，专业号，年龄）

专业（专业号，专业名）

- 假设专业表中某个元组被删除，专业号为12
- 按照设置为空值的策略，就要把学生表中专业号=12的所有元组的专业号设置为空值。
- 对应语义：某个专业删除了，该专业的所有学生专业未定，等待重新分配专业

违约处理(续)

- 对于参照完整性，除了应该定义外码，还应定义外码列是否允许空值

（1）在学生表中，“专业号”是外码，可以取空值，表示这个学生的专业尚未确定

（2）学生－选课数据库中

- SC为参照关系，Sno为外码；同时Sno为SC的主属性，按照实体完整性Sno不能为空值
- 若SC的Sno为空值，则表明尚不存在的某个学生，或者某个不知学号的学生，选修了某门课程，其成绩记录在Grade列中，这与学校的应用环境是不相符的

违约处理(续)

- 一般地，当对参照表和被参照表的操作违反了参照完整性，系统选用默认策略，即拒绝执行

- 如果想让系统采用其他的策略则必须在创建表的时候显式地加以说明

［例4］ 显式说明参照完整性的违约处理示例

```
CREATE TABLE SC
 (Sno   CHAR(9)  NOT NULL,
  Cno   CHAR(4)  NOT NULL,
  Grade  SMALLINT,
  PRIMARY KEY（Sno，Cno），
  FOREIGN KEY (Sno) REFERENCES Student(Sno)
     ON DELETE CASCADE   /*级联删除SC表中相应的元组*/
     ON UPDATE CASCADE，  /*级联更新SC表中相应的元组*/
FOREIGN KEY (Cno) REFERENCES Course(Cno)
     ON DELETE NO ACTION
     /*当删除course表中的元组造成了与SC表不一致时拒绝删除*/
     ON UPDATE CASCADE
     /*当更新course表中的cno时，级联更新SC表中相应的元组*/
  );
```

5.3 用户定义的完整性

- 用户定义的完整性就是针对某一具体应用的数据必须满足的语义要求
- RDBMS提供，而不必由应用程序承担

5.3 用户定义的完整性

5.3.1 属性上的约束条件的定义

5.3.2 属性上的约束条件检查和违约处理

5.3.3 元组上的约束条件的定义

5.3.4 元组上的约束条件检查和违约处理

5.3.1 属性上的约束条件的定义

- CREATE TABLE时定义
  - 列值非空（NOT NULL）
  - 列值唯一（UNIQUE）
  - 检查列值是否满足一个布尔表达（CHECK）

1.不允许取空值

[例5] 在定义SC表时，要求Sno、Cno、Grade属性不允许取空值。

```
CREATE TABLE SC
(Sno  CHAR(9)  NOT NULL,
 Cno  CHAR(4)  NOT NULL,
 Grade  SMALLINT NOT NULL,
 PRIMARY KEY (Sno，  Cno));
```

/* 如果在表级定义实体完整性，隐含了Sno，Cno不允许取空值，则在列级不允许取空值的定义就不写了 * /

2.列值唯一

[例6] 建立部门表DEPT，要求部门名称Dname列取值唯一，部门编号Deptno列为主码

```
CREATE TABLE DEPT
    (Deptno  NUMERIC(2),
     Dname  CHAR(9)  UNIQUE,
     /*要求Dname列值唯一*/
     Location  CHAR(10),
     PRIMARY KEY (Deptno)
);
```

3. 用CHECK短语指定列值应该满足的条件

[例7] Student表的Ssex只允许取“男”或“女”

```
CREATE TABLE Student
  (Sno  CHAR(9) PRIMARY KEY,
   Sname CHAR(8) NOT NULL,
   Ssex  CHAR(2)  CHECK (Ssex IN (‘男’, ‘女’) ) ,
   Sage  SMALLINT,
   Sdept  CHAR(20)
  );
```

3. 用CHECK短语指定列值应该满足的条件

[例8] SC表的Grade的值应该在0和100之间

```
CREATE TABLE  SC
  (Sno    CHAR(9) ,
   Cno    CHAR(4) ,
   Grade SMALLINT CHECK(Grade>=0 AND Grade<=100),
   PRIMARY KEY (Sno，  Cno),
   FOREIGN KEY (Sno) REFERENCES Student(Sno),
   FOREIGN KEY (Cno) REFERENCES Course(Cno)
  );
```

5.3.2 属性上的约束条件检查和违约处理

- 插入元组或修改属性的值时，RDBMS检查属性上的约束条件是否被满足
- 如果不满足则操作被拒绝执行

5.3.3 元组上的约束条件的定义

- 在CREATE TABLE时可以用CHECK短语定义元组上的约束条件，即元组级的限制
- 同属性值限制相比，元组级的限制可以设置不同属性之间的取值的相互约束条件

［例9］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.%'));
```

- 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

5.3.4 元组上的约束条件检查和违约处理

- 插入元组或修改属性的值时，RDBMS检查元组上的约束条件是否被满足
- 如果不满足则操作被拒绝执行

5.4 完整性约束命名子句

SQL还在CREATE TABLE语句中提供了完整性约束命名子句CONSTRAINT，用来对完整性约束条件命名

CONSTRAINT 约束定义：

CONSTRAINT <完整性约束条件名>
［PRIMARY KEY短语|FOREIGN KEY短语
|CHECK短语］

［例10］建立学生登记表Student，要求学号是以s打头的5位字符，后4位只能是数字，姓名不能取空值，年龄小于30，性别只能是“男”或“女”。

```
CREATE TABLE Student
 (Sno  CHAR(5)
  CONSTRAINT C1 CHECK (Sno Like ‘s[0-9][0-9][0-9][0-9]'),
  Sname  CHAR(20)
  CONSTRAINT C2 NOT NULL,
  Sage  NUMERIC(3)
  CONSTRAINT C3 CHECK (Sage < 30),
  Ssex  CHAR(2)
  CONSTRAINT C4 CHECK (Ssex IN ( '男'，'女')),
  CONSTRAINT StudentKey PRIMARY KEY (Sno)
);
```

［例11］建立教师表TEACHER，要求每个教师的应发工资不低于3000元（应发工资实际上就是实发工资列Sal与扣除项Deduct之和）。

```
CREATE TABLE TEACHER
 ( Eno      NUMERIC(4)  PRIMARY KEY,
   Ename    CHAR(10),
   Job      CHAR(8),
   Sal      NUMERIC(7，2),
   Deduct   NUMERIC(7，2),
   Deptno   NUMERIC(2),
   CONSTRAINT EMPFKey FOREIGN KEY (Deptno)
   REFERENCES DEPT(Deptno),
   CONSTRAINT C1 CHECK (Sal + Deduct >= 3000)
);
```

2、修改表中的完整性限制

使用ALTER TABLE语句修改表中的完整性限制

［例13］修改表Student中的约束条件，要求学号改为在900000-999999之间，年龄由小于30改为小于40

■可以先删除原来的约束条件，再增加新的约束条件

- ALTER TABLE Student DROP CONSTRAINT C1;
- ALTER TABLE Student ADD CONSTRAINT C1 CHECK (Sno BETWEEN 900000 AND 999999),
- ALTER TABLE Student DROP CONSTRAINT C3;
- ALTER TABLE Student ADD CONSTRAINT C3 CHECK (Sage < 40)；

5.5 域中的完整性限制

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14］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15] 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') );
```

[例16] 删除域GenderDomain的限制条件GD

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

[例17] 在域GenderDomain上增加限制条件GDD

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

通过［例16］和［例17］，就把性别的取值范围由('男'，'女')改为 ('1'，'0')

5.6 小结

- 数据库的完整性是为了保证数据库中存储的数据是正确的
- 数据的完整性和安全性是两个不同概念
  - 数据的完整性
    - 防止数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据
    - 防范对象：不合语义的、不正确的数据
  - 数据的安全性
    - 保护数据库防止恶意的破坏和非法的存取
    - 防范对象：非法用户和非法操作

小结(续)

- RDBMS完整性实现的机制
  - 完整性约束定义机制
  - 完整性检查机制
  - 违背完整性约束条件时RDBMS应采取的动作
- 在关系系统中，最重要的完整性约束是
  - 实体完整性
  - 参照完整性
  - 用户定义的完整性

小结(续)

- 数据库完整性的定义
  - 一般由SQL的DDL语句实现
  - 作为数据库模式的一部分存入数据字典
  - 在数据库数据修改时，RDBMS的完整性检查机制就按照数据字典中定义的这些约束进行检查
- 完整性机制的实施会影响系统性能。随着硬件性能的提高，数据库技术的发展，目前的RDBMS都提供了定义和检查实体完整性、参照完整性和用户定义的完整性的功能。

本章作业

第六版：

P.166

4,6

第五版：

P.173

4,6