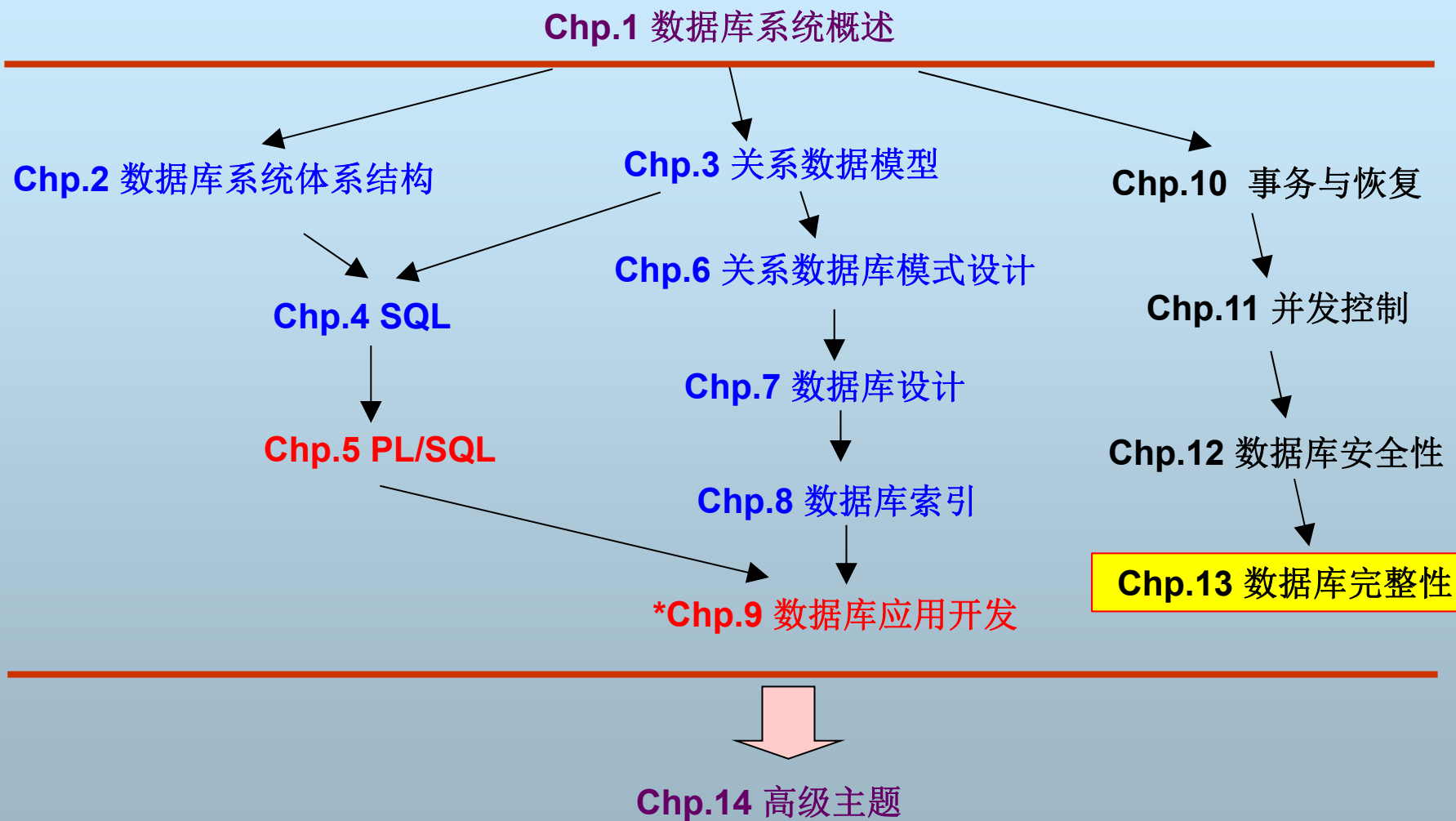


第13章 数据库完整性



课程知识结构



Databases Protection

- 数据库保护：排除和防止各种对数据库的干扰破坏，确保数据安全可靠，以及在数据库遭到破坏后尽快地恢复

- 数据库保护通过四个方面来实现

- ① 完整性控制技术
 - ◆ Enable constraints
- ② 安全性控制技术
 - ◆ Authorization and authentication
- ③ 数据库的恢复技术
 - ◆ Deal with failure
- ④ 并发控制技术
 - ◆ Deal with data sharing

数据语义
实体完整性：为了和现实一致（正确性、相容性）
外键约束
Trigger

①②④：事先
③：事后

主要内容

- 数据库完整性概念
- 完整性约束类型
- 完整性实施途径

一、数据库完整性概念

- 数据库完整性防止**合法用户**使用数据库时向数据库加入**不符合语义**的数据。防止错误的数据库数据进入数据库。

(如连续数据等)
无法保证和真实世界完全一致, 只能尽可能

- 数据库的完整性是指保护数据库中数据的约束

- **正确性**: 数据的合法性。如年龄由数字组成
- **有效性**: 数据是否在有效范围内。如月份取**1—12**
- **相容性**: 指表示同一个事实的两个数据应该一致。
。如一个人的性别只有一个

性格可能在多个表中冗余出现, 可用 trigger, 在添加时进行检查

不同 DBMS 提供了不同的机制

1、完整性控制功能

完整性子系统

■ 完整性控制机制应具有的三个功能

设计数据库时可考虑

- **定义功能**：提供定义完整性约束条件的机制
- **检查功能**：检查用户发出的操作请求是否违背了约束条件。
 - ◆ **立即执行约束**（一条语句执行完成后立即检查）
 - ◆ **延迟执行约束**（整个事务执行完毕后再检查）
- **如果发现用户操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性。**

↓ 一般会被拒绝

↓ 如 roll back

2、完整性规则定义

即约束。

- **DBA向DBMS提出的一组完整性规则**，来检查数据库中的数据是否满足语义约束，主要包括三部分：

- **触发条件**：系统什么时候使用规则来检查数据
- **约束条件**：系统检查用户发出的错误操作违背了什么完整性约束条件

- **违约响应**：违约时要做的事情 rollback | 拒绝 | 允许改，但会被改回去

自动

- 完整性约束规则是由**DBMS提供的语句**来描述，存储在数据字典，但**违约响应由系统来处理**

2、完整性规则定义

■ 一条完整性规则是一个五元组 (D,O,A,C,P)

- **D(Data)**: 约束作用的数据对象
- **O(Operation)**: 触发完整性检查的数据库操作。即当用户发出什么操作请求时需要检查该完整性规则，是立即检查还是延迟检查。
- **A(Assertion)**: 数据对象要满足的断言或语义规则
- **C(Condition)**: 受A作用的数据对象值的谓词
- **P(Procedure)**: 违反完整性规则时触发的过程

完整性规则
作用于哪些对象

一般是 insert, delete, update.

也可监视 delete, 不过一般是为了安全性功能而非完整性, 如审计

也可拓展分析 (如推理分析) 记录一些日志来分析

	学号不能为空	教授工资不得低于 1000 元
D	约束的对象为 SNO 属性	约束的对象为 Sal 属性
O	插入和修改 STUDENT 元组时	插入和修改职工元组时
A	SNO 不能为空	Sal 不能小于 1000
C	无 (作用于所有记录的 SNO 属性)	职称='教授' (作用于职称='教授'的记录)
P	拒绝执行	拒绝执行

update 语句本身不能指定, 但可通过在 trigger 加条件来实现.

二、完整性约束分类

■ 按约束的粒度

- 表级约束、列级约束、元组级约束

■ 按约束对象的状态

- 静态约束、动态约束

数据集发生变化时应满足什么条件。
→ 如选课时必须先修课程

■ 按约束作用类型分

- 域完整性、实体完整性、参照完整性

针对↓列
如类型约束,
学号为str

1、按约束粒度分类

■ 按约束的粒度分

- **表级约束：**若干元组间、关系上以及关系之间联系的约束；
- **列级约束：**针对列的类型、取值范围、精度等而制定的约束条件。
- **元组级约束：**单个元组中字段间联系的约束；

1、按约束粒度分类

[例] 建立部门表DEPT，要求部门名称Dname列为字符串型且不可为空，部门编号Deptno列为主码

```
CREATE TABLE DEPT
(
    Deptno  NUMBER,
    Dname   CHAR(9) NOT NULL,
    Location CHAR(10),
    PRIMARY KEY (Deptno)
);
```

列级约束

表级约束

1、按约束粒度分类

[例] 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,
sex      CHAR(2),
age      SMALLINT,
PRIMARY KEY (Sno),
CHECK (sex='女' OR Sname NOT LIKE 'Ms.%')
);
```

在SQL中也可认为其是表约束
元组级约束

- 定义了元组中Sname和 sex两个属性值之间的约束条件
- 性别是‘女’的元组都能通过该项检查，因为sex=‘女’成立；当性别是男性时，要检查Sname不能以Ms.开头

2、按约束对象的状态分类

■ 按约束对象的状态分

- **静态约束**：数据库每一确定状态时的数据对象所应满足的约束条件；

- ◆ 例如：学生关系中年龄不能大于**100**

- **动态约束**：数据库从一种状态转变为另一种状态时，新、旧值之间所应满足的约束条件

- ◆ 例如调整工资时须满足：

现有工资 $>$ 原有工资 + 工龄 * **100** → 一般使用 trigger 实现

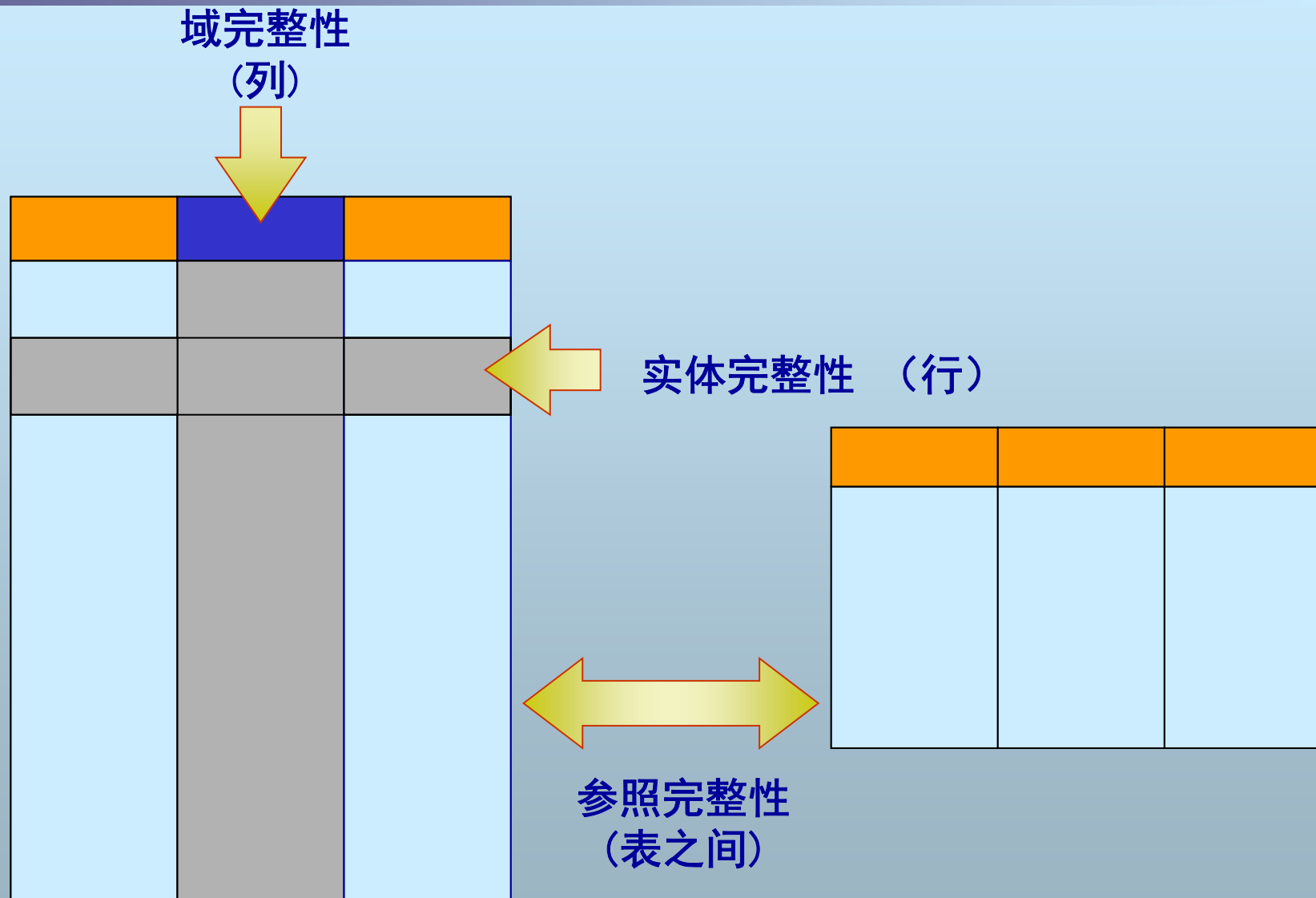
2、按约束作用类型分类

■ 按约束作用类型分

如 → 类型、范围^等，如 check 约束

- **域完整性**：域完整性为为列或列组指定一个有效的数据集，并确定该列是否允许为空
- **实体完整性**：实体完整性要求表中所有的元组都应该有一个惟一的标识符，这个标识符就是平常所说的主码
- **参照完整性**：参照完整性维护参照表中的外码与被参照表中候选码之间的相容关系。如果在被参照表中某一元组被外码参照，那么这一行既不能被删除，也不能更改

2、按约束作用类型分类



域完整性

域 比类型要“小”一些

具有相同类型的值的集合。

- 域是一组具有相同类型的值的集合 如 score [0,100] int. 域 类型
- SQL3支持域的概念，并可以用**CREATE DOMAIN***语句建立一个域以及该域应满足的完整性约束条件

仅针对数据库内部的对象（而int是通用的）。

[例] 建立一个性别域GenderDomain，并对其中的限制命名

一般会被多个表共享，使用起来较方便（类似C中的define）

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

```
CREATE TABLE student
( Sno CHAR(10),
  sex GenderDomain,
  Sname CHAR(20)
)
```

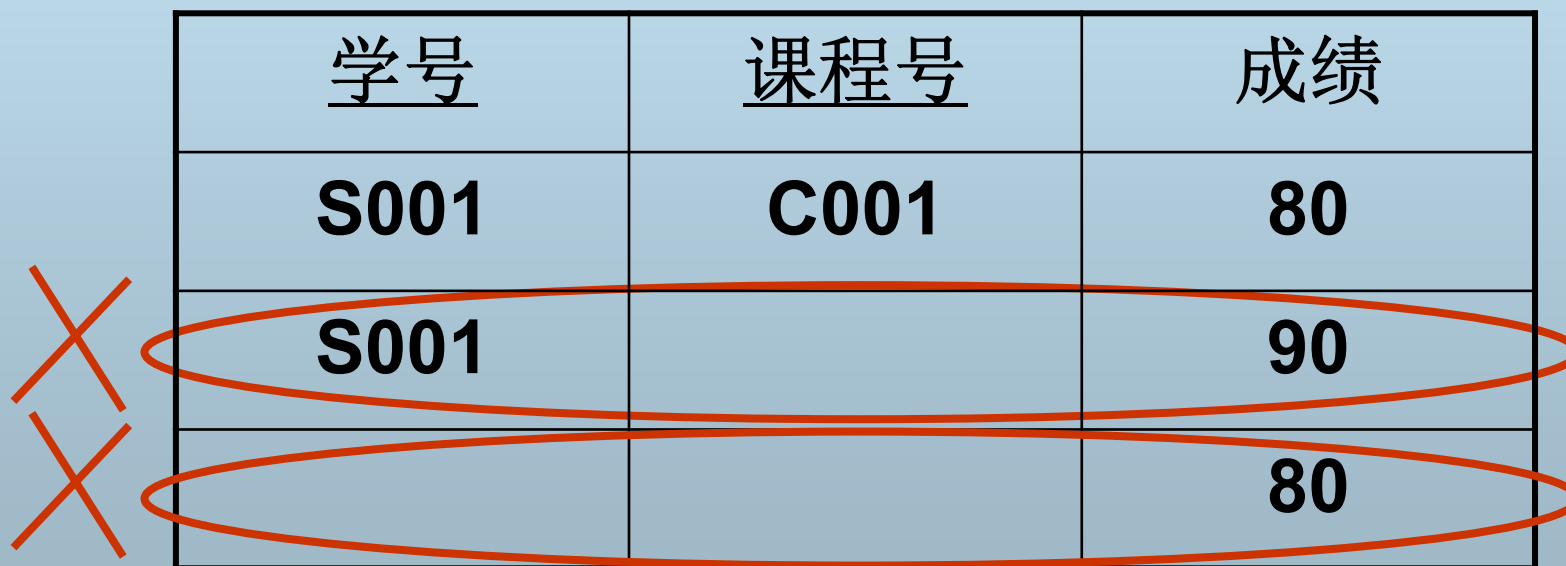
同 sex CHAR(2) + check约束

也是一种域完整性，用类型代替了域

* Oracle不支持Create Domain语句，但可用Create Type语句实现类似功能

实体完整性

- 关系模式R的主码不可为空
 - 指组成主码的所有属性均不可取空值



The diagram shows a table with three columns: 学号 (Student ID), 课程号 (Course ID), and 成绩 (Score). The first row contains the values S001, C001, and 80. The second row contains S001, an empty cell, and 90. The third row contains an empty cell, an empty cell, and 80. To the left of the table, there are two large red 'X' marks. Two red ovals are drawn around the second and third rows, highlighting the empty cells in the primary key attributes.

<u>学号</u>	<u>课程号</u>	成绩
S001	C001	80
S001		90
		80

参照完整性

表示现实中的某种联系

- 参照关系R的任一个外码值必须
 - 等于被参照关系S中所参照的候选码的某个值
 - 或者为空

R
选课关系

学号	课程号	成绩
001	002	80

002

S
学生关系

学号	姓名	专业号
001	John	

专业号	专业名	学科类别
0020	PHY	1

三、完整性实施途径

- 约束 (**Constraint**)
- 触发器 (**Trigger**)
- 规则 (**Rule**)
- 断言 (**Assertion**)

} MySQL不支持

} 不同的DBMS可能
能在功能上支持,
但实施细节上有差距

1、约束

SQL中的 constraint
DBMS 基本都支持

- 约束的用途是限制输入到表中的值的范围。
约束是实施数据完整性的首选方法。

- SQL中的约束

- 实体 ~ {
 - 主键约束 (Primary Key)
 - 唯一键约束 (Unique) 对应候选码
- 参照 ~ {
 - 外键约束 (Foreign Key)
- 域完整性 {
 - 检查约束 (Check)
 - 默认值约束 (Default) (有时也可认为是一种 Check)

1、约束

■ 数据库完整性的约束实现

完整性类型	约束类型	完整性功能描述
域完整性	DEFAULT	插入数据时，如果没有明确提供列值，则用缺省值作为该列的值
	CHECK	指定某个列或列组可以接受值的范围，或指定数据应满足的条件
实体完整性	PRIMARY KEY	指定主码，确保主码值不重复，并不允许主码为空值
	UNIQUE ↓ 候选码	指出数据应具有惟一值，防止出现冗余
参照完整性	FOREIGN KEY	定义外码、被参照表和其候选码

2、触发器

- 与特定表关联的存储过程。当在该表上执行DML操作时，可以自动触发该存储过程执行相应的操作

- 触发操作：Update、Insert、Delete，为了保证完整性约束有效（∴无select）
- 通过触发器可以实现复杂的约束。

如动态约束：失修课程

也可使用存储过程，但前端调用procedne比单纯的SQL复杂得多。故一般用Trigger

3、规则

- 规则是一组用过程化SQL(如T-SQL*)书写的条件语句
 - Where子句中合法的语句一般都可以用作规则
 - ◆ 算术运算符
 - ◆ 关系运算符
 - ◆ IN、LIKE、BETWEEN等关键字
- 规则可以和列或用户定义类型捆绑在一起，检查数据完整性
- 多个列可以共用一个规则

* T-SQL是Microsoft SQL Server的过程化SQL语言

3、规则

■ 创建规则 **【T-SQL】** SQL Server 支持.

- **CREATE RULE** *rule_name*
AS *condition_expression*

- **参数**

- ◆ **rule_name:** 是新规则的名称。规则名称必须符合标识符规则。
- ◆ **condition_expression:** 是定义规则的条件。规则可以是 **WHERE** 子句中任何有效的表达式。

3、规则

■ 绑定规则【T-SQL】系统内置的 procedure，即 sp 开头。

- 存储过程 **sp_bindrule**: 将规则绑定到列或用户自定义数据类型。

- 语法

- ◆ `sp_bindrule [@rulename =] 'rule' ,
[@objname =] 'object_name'`

- ◆ 参数

- **rulename**: 规则名
 - **objname**: 列名或用户自定义类型名

3、规则

- 例：在数据库中创建一个**Email**的规则对象，其值为包含@的字符串。

有多个email字段，希望该rule被整个db共享(类/domain)

- 创建规则(T-SQL)

```
CREATE RULE rl_email
```

```
AS
```

```
@val LIKE '%@%'
```

- 绑定规则(绑定到学生表的**Email**字段)

```
sp_bindrule 'rl_email', 'student.Email'
```

4、断言

- **Create Assertion***创建一个断言，对断言涉及的数据进行操作时会触发断言；断言为假时操作将被拒绝
- 例：限制每一门课程选修人数不超过**60**人。

- **T-SQL**

```
CREATE ASSERTION asser1 CHECK (60 >= ALL  
    (SELECT count(*) FROM SC GROUP BY c#));
```

* *Oracle不支持Create Assertion语句*

本章小结

- 数据库完整性概念
- 完整性约束类型
- 完整性实施途径