



数据库系统原理

李瑞轩

华中科技大学计算机学院



第五章 数据库完整性

Exact and careful model building should embody constraints that the final answer had in any case to satisfy.



- 学习内容

- 5.0 数据库完整性概述

- 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 完整性约束命名字句

- *5.5 域中的完整性限制

- 5.6 触发器

- 5.7 小结

5.0 数据库完整性概述

■ 数据库的完整性

- 数据的正确性和相容性
- 防止错误的数据进入数据库，造成无效操作和错误结果
- DBMS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

数据库完整性概述

■ 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

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

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，结果才能提交。

完整性控制

为维护数据库的完整性，DBMS必须：

- 1.提供定义完整性约束条件的机制
- 2.提供完整性检查的方法
- 3.违约处理

5.1 实体完整性

- 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

[例] 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*/
```

```
);
```

实体完整性定义

■ 实体完整性

- 主码的属性不能取Null、也不能有重复值
- 候选码的属性不能有重复值，可以取Null
- 主码和候选码

■ Primary Key

■ Unique

- 一般在Primary Key上自动加上Index
- 在Unique上的Index需另行声明

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：
 - 1. 检查主码值是否唯一，如果不唯一则拒绝插入或修改
 - 2. 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

实体完整性检查和违约处理(续)

- 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

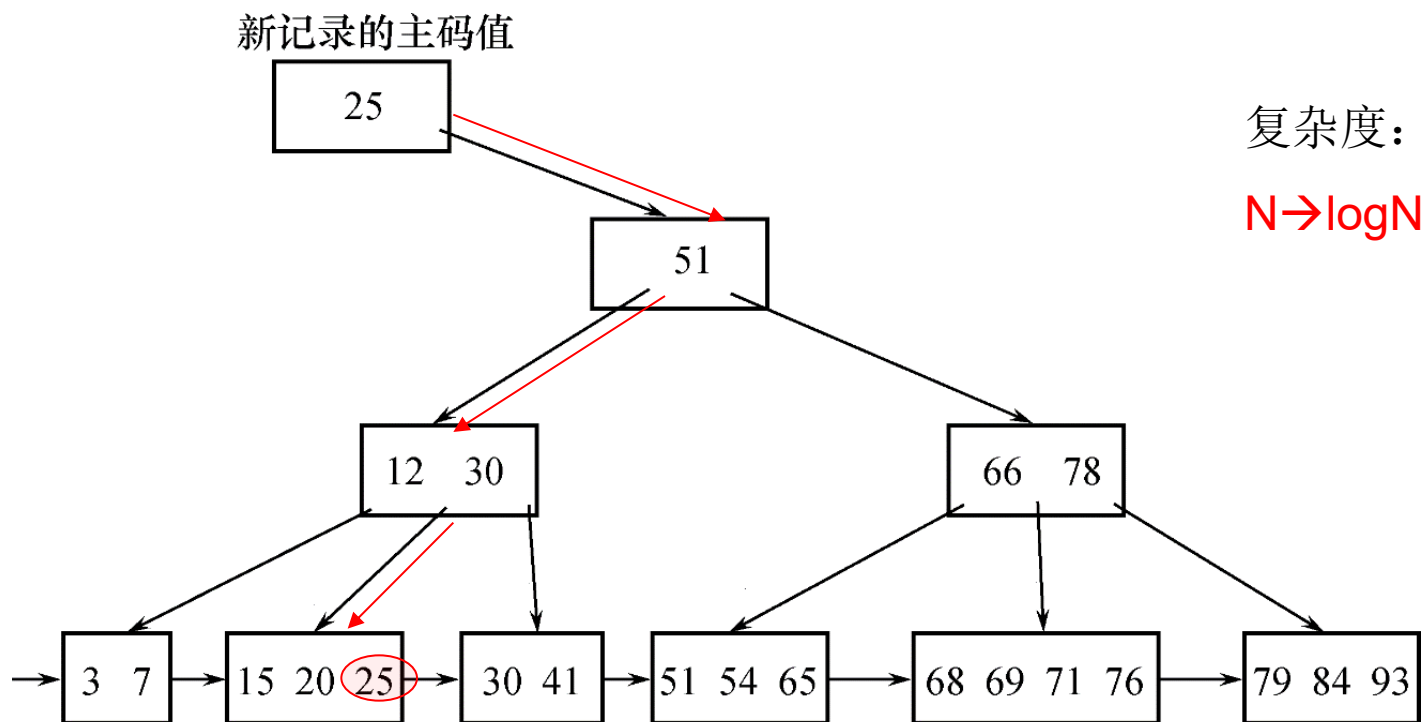
Key _i	F2 _i	F3 _i	F4 _i	F5 _i
------------------	-----------------	-----------------	-----------------	-----------------

基本表

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

实体完整性检查和违约处理(续)

- 使用索引的方法进行查找（如B树或B+树索引）



5.2 参照完整性

■ 参照完整性和外键

- 在CREATE TABLE中用FOREIGN KEY定义外码，用REFERENCES指明这些外码参照哪些表的主码
- 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	
<u>sno</u>	<u>char(10)</u>
sname	varchar(20)
sage	smallint
ssex	char
sdept	char(2)

sno = sno

SC	
<u>sno</u>	<u>char(10)</u>
<u>cno</u>	<u>char(10)</u>
grade	smallint

cno = cno

Course	
<u>cno</u>	<u>char(10)</u>
cname	varchar(20)
credit	smallint

参照完整性

■ 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

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/级联删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级联修改/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

■ 默认策略

□ 2. 级联(CASCADE)操作

□ 3. 设置为空值 (SET-NULL)

- 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

违约处理(续)

[例] 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

■ 属性上的约束条件

□ NOT NULL

- Primary Key约束隐含Not Null
- 不加约束隐含允许Null

□ UNIQUE

- 列值唯一

□ CHECK

- 检查列值是否满足一个布尔表达式

用户定义的完整性

■ 元组上的约束条件

- 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**
- 元组级的限制可以设置不同属性间取值的相互约束条件

```
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*/
);
```

5.4 完整性约束命名子句

■ CONSTRAINT 约束

CONSTRAINT <完整性约束条件名>

[PRIMARY KEY 短语

| FOREIGN KEY 短语

| CHECK 短语]

完整性约束命名子句(续)

[例] 建立学生登记表Student, 要求学号在90000~99999之间, 姓名不能取空值, 年龄小于30, 性别只能是“男”或“女”。

```
CREATE TABLE Student  
(Sno NUMERIC(6)  
  CONSTRAINT C1 CHECK (Sno BETWEEN 1000 AND 9999),  
  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)  
);
```

- ✓ 在Student表上建立了5个约束条件, 包括主码约束(命名为StudentKey)以及C1、C2、C3、C4四个列级约束。

完整性约束命名子句(续)

■ 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

```
ALTER TABLE Student
```

```
    DROP CONSTRAINT C1;
```

```
ALTER TABLE Student
```

```
    ADD CONSTRAINT C1 CHECK (Sno BETWEEN 1000 AND  
    9999),
```

```
ALTER TABLE Student
```

```
    DROP CONSTRAINT C3;
```

```
ALTER TABLE Student
```

```
    ADD CONSTRAINT C3 CHECK (Sage < 40);
```

断言 (Assertions)

- These are database-schema elements, like relations or views.
- Defined by:

CREATE ASSERTION <name>

CHECK (<condition>);


- Condition may refer to any relation or attribute in the database schema.

Example: Assertion

- In **Sells(bar, beer, price)**, no bar may charge an average of more than \$5.

```
CREATE ASSERTION NoRipoffBars  
CHECK (  
  NOT EXISTS (  
    SELECT bar FROM Sells  
    GROUP BY bar  
    HAVING 5.00 < AVG(price)  
  );
```

Bars with an
average price
above \$5



Example: Assertion

- In `Drinkers(name, addr, phone)` and `Bars(name, addr, license)`, there cannot be more bars than drinkers.

```
CREATE ASSERTION FewBar CHECK (  
    (SELECT COUNT(*) FROM Bars) <=  
    (SELECT COUNT(*) FROM Drinkers)  
);
```

Timing of Assertion Checks

- In principle, we must check every assertion after every modification to any relation of the database.
- A clever system can observe that only certain changes could cause a given assertion to be violated.
 - **Example:** No change to Beers can affect FewBar. Neither can an insertion to Drinkers.

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] 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'));
```

域中的完整性限制(续)

[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' ) );
```

✓ 将性别的取值范围由('男', '女')改为 ('1', '0')

5.6 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
- 触发器
 - Trigger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

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>

激活的时机

事件

名称

新值/修改后的值
旧值/修改前的值

Create Trigger NetWorthTrigger
After Update Of netWorth On MovieExec
Referencing

Old As OldTuple, New As NewTuple

For Each Row

每修改一个元组便激活

When (*OldTuple.netWorth > NewTuple.netWorth*)

Update MovieExec

Set netWorth = OldTuple.netWorth

Where cert# = NewTuple.cert#

执行程序的条件

INSTEAD OF触发器

- INSTEAD OF触发器可以实现将对视图的更新转换为对多个基本表的更新
- CREATE TRIGGER语法格式

CREATE TRIGGER <触发器名>

INSTEAD OF [INSERT | DELETE | UPDATE | UPDATE
OF 列名称 [,列名称,...]] ON 视图名称

FOR EACH ROW

<触发动作体>

触发器类型

事件

触发器名称

Create Trigger View_Del
Instead Of Delete On MovieView
Referencing

Old As OldTuple

For Each Row

Delete From Movie

Where m_id = OldTuple.m_id

And exec_director = 'Tomas';

新值/修改后的值
旧值/修改前的值

只能使用行级触发器

一般无须指明条件

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） 执行激活触发器的**SQL**语句；
 - （3） 执行该表上的**AFTER**触发器。

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] 删除教师表Teacher上的触发器Insert_Sal

DROP TRIGGER Insert_Sal ON Teacher;

5.7 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
- **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