第5章 数据库的安全和完整性约束

数据库是一种共享资源,可以为多个用户共享。但数据库中每部分数据分别属于不同的人,数据的所属者都非常关心他们数据的安全性是否会遭到破坏。而且随着 Web 数据库的应用越来越广泛,数据库的安全问题日益突出,如何保证和加强数据库的安全性显得尤为重要。

一般来讲,数据库的破坏来自于几个方面:系统故障、并发所引起的数据不一致,人为的破坏、输入或更新数据库的数据有误,更新事务未遵守保持数据一致性原则。对付系统故障、并发所引起的数据不一致的措施第 4 章中已有讨论。人为的破坏问题属于数据库安全问题。最后一种破坏问题属于完整性约束问题。本章主要讨论数据库的安全和完整性约束。

5.1 数据库的安全

数据库的安全是指保护数据库以防止非法用户的越权使用、窃取、更改或破坏数据。数据库的安全涉及很多层面,必须在以下几个层面做好安全措施。

① 物理层:重要的计算机系统必须在物理上受到保护,以防止入侵者强行进入或暗中潜入。

② 人员层:数据库系统的建立、应用和维护等工作,一定要由可靠的合法用户来进行。

③ 操作系统层:要进入数据库系统,首先要经过操作系统,如果操作系统的安全性差,数据库将面临着重大的威胁。

④ 网络层:由于几乎所有网络上的数据库系统都允许通过终端或网络进行远程访问,所以网络的安全和操作系统的安全一样重要,网络安全了,无疑对数据的安全提供了保障。

⑤ 数据库系统层:数据库系统应该有完善的访问控制机制,以防止非法用户的非法操作。

为了保证数据库的安全,必须在以上所有层面上进行安全性保护。如果物理层或人员层存在安全隐患,即使操作系统层、网络层以及数据库系统层访问控制很严格,数据库也可能是不安全的。本节主要讨论数据库系统层上的安全性措施。

5.1.1 用户标识和鉴别

用户标识和鉴别是系统提供的最外层的安全保护措施。该方法主要是由系统提供一定的方式让用户标识自己的名字和身份,每次用户要进入系统时,系统对用户身份进行核实,通过鉴别后才提供机器使用权。

用户标识和鉴别常用的方法如下。

① 用一个用户名或用户标识号标明用户身份。系统内部记录着所有合法用户的标识,当用户提供了用户名或用户标识号后,系统与内部记录的合法用户标识进行核实,若

是合法用户，则要求用户输入口令以进一步核实；否则，则不能使用计算机。

② 输入口令。为进一步核实用户，系统常常要求用户再输入口令，为保密起见，用户输入的口令不显示在终端屏幕上。系统通过核对口令鉴别用户身份。

③ 利用只有用户具有的物品鉴别用户。钥匙就是属于这种性质的鉴别物。在计算机系统中常用磁性卡片作为用户身份凭证，但系统必须有阅读磁卡的装置，而且磁卡也有丢失或被盗的危险。

④ 利用用户的个人特征鉴别用户。签名、指纹、声音等都是用户个人特征。利用这些用户个人特征来鉴别用户非常可靠，但需要昂贵的、特殊的鉴别装置，因而影响了它们的推广和使用。

5.1.2 存取控制

存取控制是指对用户访问数据库各种资源的权力的控制。这里资源是指基表、视图、各种目录以及实用程序等，而权利是指对表进行创建、撤销、查询、增、删、改等。数据库用户按其访问权力的大小，可分为一般数据库用户、具有支配部分数据库资源特权的数据库用户以及具有 DBA 特权的数据库用户。

一般数据库用户是具有 connect 特权的用户，可以查询或更新数据库中的数据；可以创建视图或定义数据的别名。

具有支配部分数据库资源特权的数据库用户是具有 resource 特权的用户，可以创建表、索引和簇集；可以授予或收回其他数据库用户对其所创建的数据对象所拥有的访问权；有权对其创建的数据对象跟踪审查。

具有 DBA 特权的数据库用户权利最大，有权访问数据库中任何数据；不但可以授予或收回数据库用户对数据对象的访问权，还可以批准或收回数据库用户；可以为 public 定义别名，public 是所有数据库用户的总称；可以对数据库进行调整、重组或重构；可以控制整个数据库的跟踪审查。

在 SQL 中，有两种授权方式：①由 DBA 授予某类数据库用户的特权；②由 DBA 或由数据对象的创建者授予对某些数据对象进行某些操作的特权。

授予特权的方式和特权收回的方式及例子，见 3.7 节内容。

在数据目录中，有一张授权表，记录了每个数据库的授权情况。在数据库中，许多用户的权限相同，如分别授权，十分烦琐。可以为每个用户定义一个角色，然后对角色授权，某用户承担某种角色就拥有该角色的权限，这样就简单了。当然，一个用户可以拥有多个角色和其他权限。

为了保证数据库的安全性，系统必须保证用户只能存取他有权存取的数据。定义一个用户的存取权限就是定义一个用户可以在哪些数据对象上进行哪些类型的操作。在数据库系统中，定义存取权限称为授权。

用户在数据对象上的存取权限有以下几种。

① **read** 权限。允许读取数据，但不允许修改数据。

② **insert** 权限。允许插入新数据,但不允许修改已经存在的数据。

③ **update** 权限。允许修改数据,但不允许删除数据。

④ **delete** 权限。允许删除数据。

⑤ **index** 权限。允许创建和删除索引。

⑥ **resource** 权限。允许创建新关系。

⑦ **alteration** 权限。允许添加或删除关系中的属性。

⑧ **drop** 权限。允许删除关系。

注意:delete 权限和 drop 权限的区别在于 delete 权限只允许对关系中的元组进行删除,即使删除了关系中所有的元组,关系仍然存在。drop 权限删除的是整个关系,删除后关系不再存在。

具有 resource 权限的用户在创建新关系后自动获得该关系上的所有权限。

一旦定义了用户的存取权限,DBMS 根据每个用户的存取权限进行合法性检查,若用户的操作请求超出了定义的权限,系统将拒绝执行该操作。常用的存取控制方法有两类。

① 自主存取控制(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也有不同的存取权限,用户还可以将其拥有的存取权限授予其他用户。因此,自主存取控制非常灵活。该方法详见 3.7 节中关于如何实现授权和回收权限问题的讨论。

② 强制存取控制(MAC)。每一个数据对象被标以一定的密级,每一个用户也被授予某一个级别的许可证。对任意一个对象,只有合法许可证的用户才可以存取。因此,强制存取控制相对比较严格。

5.1.3 视图定义和查询修改

利用视图定义也可以在一定程度上起到保护数据库的作用。例如,为不同的用户定义不同的视图,以限制各个用户的访问范围,详见 3.6.5 节的讨论。但是,查询修改只能处理一些比较简单的访问限制,不如视图灵活。例如,通过在多表上定义视图,可以利用其他表上的条件限制本表的查询范围,这种功能用查询修改就难以实现。而且,视图的作用除了提高数据库的安全性外,还提高了数据库的逻辑独立性。

5.1.4 数据加密

另一种安全性方法是使用数据加密。把数据用密码形式存储在磁盘上,这样那些企图通过不正常渠道存取数据的人就只能看到一些无法辨认的二进制数。数据在存入时加密了,在查询时就须解密,增加了系统开销,降低了数据库的性能。只有对那些保密要求特别高的数据,才值得采用此方法。关于密码问题,有专门课程论述,本书不再介绍。

5.1.5 审计跟踪

审计跟踪是一种监视措施，对某些保密数据，它跟踪记录有关这些数据的访问活动。一旦发现潜在的窃密企图，例如重复的、相似的查询，有些 DBMS 会自动发出警报；有些 DBMS 虽无自动报警功能，但可根据这些数据进行事后分析和调查。审计跟踪的结果记录在一个审计跟踪日志文件上。审计跟踪日志文件记录一般包括下列内容。

① 操作类型(例如修改、查询等)。

② 操作终端标识与操作者标识。

③ 操作日期和时间。

④ 所涉及的数据(如表、视图、记录、属性等)。

⑤ 数据的前像和后像。

审计跟踪在几个方面加强了安全性。例如，如果发现一个账户的余额不正确，银行也许会跟踪所有在这个账户上的更新来找到错误，同时也会找到执行这个更新的人。然后银行就可利用审计跟踪日志文件跟踪这个人所做的所有更新以找到其他错误。

审计跟踪可通过在关系更新操作上定义适当的触发器来实现，也可利用数据库系统提供的内置机制来实现。

5.2 数据库的完整性约束

数据库的完整性约束可保证授权用户对数据库进行修改时不会破坏数据的一致性。因此，完整性约束防止的是对数据库的意外破坏。

5.2.1 域完整性约束

域的完整性约束是指关系中每个属性的取值范围应该为指定范围内的值，也就是对属性取值的约束。域的完整性约束是最基本的完整性约束。当在关系中插入新的数据时，系统会根据域的约束条件自动进行检查。

例如，如果规定选课表 SC 中成绩属性 GRADE 的取值为 0～100，那么当往 SC 表中插入学生成绩数据时，如果插入的成绩不在 0～100 之内，系统便会报错，错误的数据也就不能插入 SC 表中了。

5.2.2 引用完整性约束

引用完整性约束是指不同关系之间或同一关系的不同元组间的约束。如果一个表中存在外键，则外键的值必须存在，或者为空。

例如，选课表中 SNO 属性和 CNO 属性都是外键，来自于学生表中 SNO 属性的值和

课程表中 CNO 属性的值,因此,它们的值必须在学生表和课程表中存在。

5.2.3 实体完整性约束

每一个关系都有一个用来唯一识别一个元组的主键。因此,它的值不能为空,也不能出现重复,否则无法区分和识别元组,这就是实体完整性约束。

例如,学生表中 SNO 是主键,因此往 SC 表中插入数据时 SNO 不允许为空,也不允许插入一个与已存在 SNO 相同的元组。

5.2.4 其他完整性约束

域完整性约束、实体完整性约束和引用完整性约束是对关系模式的约束,除此之外,还有其他一些完整性约束。例如,关系的属性应是原子的,即满足第一范式的约束。这是对数据模型的约束,在 DBMS 实现时已经考虑,不必特别说明;另外,数据库从一个状态变为另一个状态时也应遵守一定的约束,例如,更新职工表时,工资、工龄这些属性值一般只会增加,不会减少。这种约束一般是显式说明。

5.2.5 完整性约束的说明

1. 用 SQL 提供的 DDL 语句说明约束

域完整性约束、实体完整性约束和引用完整性约束可用 SQL 提供的 DDL 语句说明。例如,第 3 章中的例 3.3.3 是创建一个选课表 SC,它由学号 SNO、课程号 CNO、成绩 GRADE 三个属性组成。如果规定选课表 SC 中成绩属性 GRADE 的取值为 0～100。

利用 SQL 中 **CREATE TABLE** 语句定义如下:

```
CREATE TABLE SC
 (SNO CHAR(8)NOT NULL,                -- 域约束
 CNO CHAR(6) NOT NULL,                -- 域约束
 GRADE DEC (4,1) DEFAULT NULL,        -- 域约束
 PRIMARY KEY (SNO,CNO),               -- 定义主键,实体完整性约束
 FOREIGN KEY (SNO)                    -- 定义外键,引用完整性约束
    REFERENCES STUDENT,
    ON DELETE CASCADE,
/* 当主表中删除了某一主键时,基表中引用此主键的行也随之被删除 */
 FOREIGN KEY (CNO)                    -- 定义外键,引用完整性约束
    REFERENCES COURSE,
    ON DELETE RESTRICT,               -- 凡是被基表所引用的主键,不得被删除
 CHECK(BETWEEN 0 AND 100));           -- 定义值域,域约束
```

2. 用断言说明约束

断言(assertions)是指数据库状态必须满足的逻辑条件。实际上,数据库完整性约束可以看成一系列断言的集合,DBA 可以用断言的形式写出数据库的完整性约束,由系统编译成约束库。DBMS 中还有一个完整性控制子系统,对每个更新事务,用约束库中的断言对它进行检查。如果发现更新违反约束,就卷回该事务。

SQL 语言标准中增加了 ASSERT 语句,可以用来说明断言。例如,一个人的存款不能为负,可以用下面的语句表示:

```
ASSERT 余额约束 ON 储蓄账户: 余额>=0;
```

ASSERT,**ON** 是两个关键字。余额约束是约束的名称,储蓄账户是该约束所作用的关系名。冒号后面是断言,表示储蓄账户中的余额不能为负。

3. 用触发器说明约束

断言表示数据库状态应满足的条件,而触发器中表示的却是违反约束的条件。触发器(trigger)是一个软件机制,其功能相当于下面的语句:

```
WHENEVER <事件>
IF <条件> THEN <动作>;
```

其语义为:当某一个事件发生时,如果满足给定的条件,则执行相应的动作。

这种规则称为主动数据库规则(active database rules),又称为 ECA 规则(取事件、条件、动作英文名的首字母),也称为触发器。

利用触发器可以表示约束,以违反约束作为条件,以违反约束的处理作为动作。动作不限于卷回事务,也可以给用户一个消息或执行一个过程。在系统中定义一批触发器后,就会监视数据库状态。一旦出现违反约束的更新,就会引发相应的动作。

触发器可定义(按照 ECA 规则)为

```
触发器::=CREATE TRIGGER <触发器名>
          {BEFORE|AFTER} <触发事件>
           ON <表名>
          [REFERENCING <引用名>] /旧值和新值的别名/
          FOR EACH {ROW|STATEMENT}
          WHEN (<条件>)
          <动作>
<触发事件>::=INSERT|DELETE|UPDATE[OF <属性表>]
<引用名>::=OLD [ROW] [AS] <旧元组名>
            NEW [ROW] [AS] <新元组名>
            OLD TABLE [AS] <旧表名>
            NEW TABLE [AS] <新表名>
```

注意:有些规则要求在事件前触发,有些规则要求在事件后触发,这可以通过定义中

BEFORE 或 AFTER 来指定。

在 SQL 新标准中,增加了定义触发器的语句。

例 5.2.1 引用完整性约束的实现。以 3.3.1 节定义的 STUDENT、COURSE、SC 三个表为例,其中 SC 表中定义了两个外键 SNO 和 CNO 及其完整性约束,试写出实现此引用完整性约束的规则。

首先,我们分析一下有哪些操作会影响到本例的引用完整性约束?

对于 STUDENT 表来说,在其上进行删除操作和更新 UPDATE(SNO)操作时,由于选课表 SC 中引用了 STUDENT 中的 SNO,因此,这两个操作会影响引用完整性约束。

对于 COURSE 表来说,在其上进行删除操作和更新 UPDATE(CNO)操作时,由于选课表 SC 中引用了 COURSE 中的 CNO,因此,这两个操作会影响引用完整性约束。

对于 SC 来说,在其上进行插入和更新 UPDATE(SNO,CNO)操作时,由于 SC 表中引用了 STUDENT 表中的 SNO 和 COURSE 表中的 CNO,因此,这两个操作会影响引用完整性约束。

下面我们以 STUDENT 表上的删除操作以及 SC 表上的更新操作为例,写出实现此引用完整性约束的规则。其他的规则类似,读者可以自己试写。

```
CREATE TRIGGER STUDENT-DELETE
AFTER DELETE ON STUDENT
REFERENCING OLD TABLE AS O
FOR EACH ROW
WHEN(EXISTS(SELECT * FROM SC WHERE SNO = O.SNO))
DELETE FROM SC
WHERE SNO = O.SNO;
```

在 SC 表的定义中,外键 SNO 的定义中采用了 CASCADE 选项,所以,当在 STUDENT 表中删除一个元组时,则在 SC 表中删除引用该元组的主键作为外键的所有元组。

BEFORE 和 AFTER 分别表示事件前触发和事件后触发;REFERENCING 子句是为过渡值定义引用名,由于删除操作只有旧值,没有新值,所以只为旧值定义了引用名 O;ROW 表示监控的粒度为按行监控,STATEMENT 表示监控的粒度为按语句监控。

```
CREATE TRIGGER SC - FK - UPDATE
BEFORE UPDATE OF SNO,CNO ON SC
REFERENCING NEW AS N
FOR EACH ROW
WHEN(NOT(EXISTS(SELECT * FROM  STUDENT
                WHERE SNO = N.SNO)
     AND
        EXISTS(SELECT * FROM COURSE
                WHERE CNO = N.CNO)))
ROLLBACK;
```

如果想更新 SC 表中的外键，使 STUDENT 表和 COURSE 表中无相应的主键供其引用，则卷回更新此元组的操作。

触发器不但可以用来表示约束，也可用来监视数据库状态。例如，当仓库中某项材料的库存量低于一定水平时，触发器会主动通知采购人员订购该项材料。

例 5.2.2 库存量控制。设有两张表：

```
库存(零件号,库存量,库存下限,订购量)
在购订单(零件号,订购量,订购日期)
```

当库存量低于库存下限，而在购订单表中没有该零件的订单时，则将该零件的订单插入在购订单表中，订单的订购数量按库存表中的规定。

下面是控制库存量的规则：

```
CREATE TRIGGER 库存控制
AFTER UPDATE OF 库存量 ON 库存
REFERENCING NEW TABLE AS N
FOR EACH ROW
WHEN(N.库存量< N.库存下限 AND NOT EXISTS(SELECT * FROM 在购订单
                                          WHERE 零件号 = N.零件号))
INSERT INTO 在购订单 VALUES(N.零件号,N.订购量,SYSDATE);
```

4. 在基表定义中加 CHECK 子句

利用 **CHECK** 子句可以表示单表中的约束，尤其是属性值的约束。

例 5.2.3 在 STUDENT 的定义中增加一个检查：女同学的年龄应为 15～30 岁，男同学年龄应为 15～35 岁。

用 CHECK 子句表示的约束如下：

```
CHECK(AGE >= 15 AND ((SEX = 'M' AND AGE <= 35)OR(SEX = 'F' AND AGE <= 30)));
```

思考题

1. 在数据库系统中，数据的安全性和完整性有什么区别？系统如何实现数据的安全性和完整性控制？

2. 数据库的安全性和计算机系统安全性有什么联系？如何判断一个好的安全性措施？

重点内容与典型题目

重点内容

1. 数据库安全性实现的各种方法。
2. 数据库中完整性约束实现的各种方法。

典型题目

1. 假设某单位人事管理系统中有两个表：

职工(职工号,职工名,年龄,性别,职务,工资,部门号)

部门(部门号,部门名,经理名,地址,电话号码)

请用 SQL 的 GRANT、REVOKE 语句以及视图机制,完成以下授权定义或存取控制功能。

(1) 用户王云对两个表具有 SELECT 权限；

(2) 用户刘宁对两个表具有 INSERT 和 DELETE 权限；

(3) 用户张明对职工表具有 SELECT 权限,对工资字段具有更新权限；

(4) 用户周蕾具有对两个表的所有权限,并具有给其他用户授权的权限。

(5) 将用户刘宁对职工表的 INSERT 权限收回。

(6) 用户郑琪具有从每个部门职工中查询最高工资、最低工资、平均工资的权限,但他不能查看每个职工的工资。

2. 以你熟悉的一个数据库管理系统为例,简述该系统的完整性控制方法。在 DBMS 中,当发现违反这些完整性约束时,DBMS 将分别采取何种措施?

3. 写一个触发器,监视 STUDENT 表上的 UPDATE 操作,判断其更新后的元组是否有 1990 年 1 月 1 日后出生的学生,将这样的学生插入 YONGSTUDENT 表中。

习题

1. 什么是数据库的完整性? DBMS 的完整性子系统的功能是什么?

2. 引用完整性约束在 SQL 中可以用哪几种方式实现?

3. 数据库的完整性和一致性有何异同点?

4. 数据库的安全性和完整性有什么区别和联系?

5. 数据库完整性受到破坏的原因主要来自哪几个方面?

6. 鉴别用户的身份通常有哪几种方法?

7. 试述数据库权限的作用。

8. 试对 SQL 中 CHECK 子句和断言两种完整性约束进行比较,各说明什么对象,何时激活,能保证数据库的一致性么?

9. 有一选课关系 SC(SNO,CNO,GRADE),如果规定 0≤GRADE≤100,试用触发器 **TRIGGER** 说明该完整性约束。

10. 在教学数据库的关系 SC、STUDENT、COURSE 中,试用 SQL 的断言机制定义下列两个完整性约束。

(1) 学生必须在选修 MATHS 课后,才能选修其他课程。

(2) 每个男学生最多选修 20 门课程。