Chapter 5: The Relational Data Model and Relational Database Constraints

CHAPTER 5: THE RELATIONAL DATA MODEL AND RELATIONAL DATABASE CONSTRAINTS

Answers to Selected Exercises

5.10 Suppose each of the following update operations is applied directly to the database of Figure 5.6. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints:

(a) Insert < 'Robert', 'F', 'Scott', '943775543', '21-JUN-42', '2365 Newcastle Rd, Bellaire, TX', M, 58000, '888665555', 1 > into EMPLOYEE.

(b) Insert < 'ProductA', 4, 'Bellaire', 2 > into PROJECT.

(c) Insert < 'Production', 4, '943775543', '01-OCT-88' > into DEPARTMENT.

(d) Insert < '677678989', null, '40.0' > into WORKS_ON.

(e) Insert < '453453453', 'John', M, '12-DEC-60', 'SPOUSE' > into DEPENDENT.

(f) Delete the WORKS_ON tuples with ESSN= '333445555'.

(g) Delete the EMPLOYEE tuple with SSN= '987654321'.

(h) Delete the PROJECT tuple with PNAME= 'ProductX'.

(i) Modify the MGRSSN and MGRSTARTDATE of the DEPARTMENT tuple with DNUMBER=5 to '123456789' and '01-OCT-88', respectively.

(j) Modify the SUPERSSN attribute of the EMPLOYEE tuple with SSN= '999887777' to '943775543'.

(k) Modify the HOURS attribute of the WORKS_ON tuple with ESSN= '999887777' and PNO= 10 to '5.0'.

Answers:

(a) No constraint violations.

(b) Violates referential integrity because DNUM=2 and there is no tuple in the DEPARTMENT relation with DNUMBER=2. We may enforce the constraint by: (i) rejecting the insertion of the new PROJECT tuple, (ii) changing the value of DNUM in the new PROJECT tuple to an existing DNUMBER value in the DEPARTMENT relation, or (iii) inserting a new DEPARTMENT tuple with DNUMBER=2.

(c) Violates both the key constraint and referential integrity. Violates the key constraint because there already exists a DEPARTMENT tuple with DNUMBER=4. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of DNUMBER in the new DEPARTMENT tuple to a value that does not violate the key constraint. Violates referential integrity because MGRSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of MGRSSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='943775543'.
(d) Violates both the entity integrity and referential integrity. Violates entity integrity because PNO, which is part of the primary key of WORKS_ON, is null. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of PNO in the new WORKS_ON tuple to a value of PNUMBER that exists in the PROJECT relation. Violates referential integrity because ESSN='677678989' and there is no tuple in the EMPLOYEE relation with SSN='677678989'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of ESSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='677678989'.

(e) No constraint violations.

(f) No constraint violations.

(g) Violates referential integrity because several tuples exist in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations that reference the tuple being deleted from EMPLOYEE. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting all tuples in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations whose values for ESSN, ESSN, MGRSSN, and SUPERSSN, respectively, is equal to'987654321'.

(h) Violates referential integrity because two tuples exist in the WORKS_ON relations that reference the tuple being deleted from PROJECT. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting the tuples in the WORKS_ON relation whose value for PNO=1 (the value for the primary key PNUMBER for the tuple being deleted from PROJECT).

(i) No constraint violations.

(j) Violates referential integrity because the new value of SUPERSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the deletion, or (ii) inserting a new EMPLOYEE tuple with SSN='943775543'.

(k) No constraint violations.

5.11 Consider the AIRLINE relational database schema shown in Figure 5.8, which describes a database for airline flight information. Each FLIGHT is identified by a flight NUMBER, and consists of one or more FLIGHT_LEGs with LEG_NUMBERs 1, 2, 3, etc. Each leg has scheduled arrival and departure times and airports, and has many LEG_INSTANCEs—one for each DATE on which the flight travels. FARES are kept for each flight. For each leg instance, SEAT_RESERVATIONs are kept, as is the AIRPLANE used in the leg, and the actual arrival and departure times and airports. An AIRPLANE is identified by an AIRPLANE_ID, and is of a particular AIRPLANE_TYPE. CAN_LAND relates AIRPLANE_TYPES to the AIRPORTs in which they can land. An AIRPORT is identified by an AIRPORT_CODE. Consider an update for the AIRLINE database to enter a reservation on a particular flight or flight leg on a given date.

(a) Give the operations for this update.

(b) What types of constraints would you expect to check?

(c) Which of these constraints are key, entity integrity, and referential integrity constraints and which are not?
(d) Specify all the referential integrity constraints on Figure 5.8.

Answers:

(a) One possible set of operations for the following update is the following:

```
INSERT <FNO,LNO,DT,SEAT_NO,CUST_NAME,CUST_PHONE> into
SEAT_RESERVATION; MODIFY the LEG_INSTANCE tuple with the condition:
( FLIGHT_NUMBER=FNO AND LEG_NUMBER=LNO AND DATE=DT) by setting
NUMBER_OF_AVAILABLE_SEATS = NUMBER_OF_AVAILABLE_SEATS - 1; These
operations should be repeated for each LEG of the flight on which a reservation is
made. This assumes that the reservation has only one seat. More complex operations will
be needed for a more realistic reservation that may reserve several seats at once.
```

(b) We would check that NUMBER_OF_AVAILABLE_SEATS on each LEG_INSTANCE of
the flight is greater than 1 before doing any reservation (unless overbooking is permitted),
and that the SEAT_NUMBER being reserved in SEAT_RESERVATION is available.

(c) The INSERT operation into SEAT_RESERVATION will check all the key, entity integrity,
and referential integrity constraints for the relation. The check that
NUMBER_OF_AVAILABLE_SEATS on each LEG_INSTANCE of the flight is greater than 1
does not fall into any of the above types of constraints (it is a general semantic integrity
constraint).

(d) We will write a referential integrity constraint as R.A --> S (or R.(X) --> T)
whenever attribute A (or the set of attributes X) of relation R form a foreign key that
references the primary key of relation S (or T). FLIGHT_LEG.FLIGHT_NUMBER --> FLIGHT
FLIGHT_LEG.DEPARTURE_AIRPORT_CODE --> AIRPORT
FLIGHT_LEG.ARRIVAL_AIRPORT_CODE --> AIRPORT
LEG_INSTANCE.(FLIGHT_NUMBER,LEG_NUMBER) --> FLIGHT_LEG
LEG_INSTANCE.DEPARTURE_AIRPORT_CODE --> AIRPORT
LEG_INSTANCE.ARRIVAL_AIRPORT_CODE --> AIRPORT
LEG_INSTANCE.AIRPLANE_ID --> AIRPLANE
FARES.FLIGHT_NUMBER --> FLIGHT
CAN_LAND.AIRPLANE_TYPE_NAME --> AIRPLANE_TYPE
CAN_LAND.AIRPORT_CODE --> AIRPORT
AIRPLANE.AIRPLANE_TYPE --> AIRPLANE_TYPE
SEAT_RESERVATION.(FLIGHT_NUMBER,LEG_NUMBER,DATE) --> LEG_INSTANCE

5.12 Consider the relation CLASS(Course#, Univ_Section#, InstructorName, Semester,
BuildingCode, Room#, TimePeriod, Weekdays, CreditHours). This represents classes taught
in a university with unique Univ_Section#. Give what you think should be various candidate
keys and write in your own words under what constraints each candidate key would be valid.

Answer:

Possible candidate keys include the following (Note: We assume that the values of the
Semester attribute include the year; for example "Spring/94" or "Fall/93" could be
values for Semester):
1. {Semester, BuildingCode, Room#, TimePeriod, Weekdays} if the same room cannot be
used at the same time by more than one course during a particular semester.
2. {Univ_Section#} if it is unique across all semesters.
3. {InstructorName, Semester} if an instructor can teach at most one course during each
4. If Univ_Section# is not unique, which is the case in many universities, we have to examine the rules that the university uses for section numbering. For example, if the sections of a particular course during a particular semester are numbered 1, 2, 3, ..., then a candidate key would be \{Course#, Univ_Section#, Semester\}. If, on the other hand, all sections (of any course) have unique numbers during a particular semester only, then the candidate key would be \{Univ_Section#, Semester\}.

5.13 Consider the following six relations for an order-processing database application in a company:

CUSTOMER (Cust#, Cname, City)
ORDER (Order#, Odate, Cust#, Ord_Amt)
ORDER_ITEM (Order#, Item#, Qty)
ITEM (Item#, Unit_price)
SHIPMENT (Order#, Warehouse#, Ship_date)
WAREHOUSE (Warehouse#, City)

Here, Ord_Amt refers to total dollar amount of an order; Odate is the date the order was placed; Ship_date is the date an order is shipped from the warehouse. Assume that an order can be shipped from several warehouses. Specify foreign keys for this schema, stating any assumptions you make.

Answer:

Strictly speaking, a foreign key is a set of attributes, but when that set contains only one attribute, then that attribute itself is often informally called a foreign key. The schema of this question has the following five foreign keys:
1. the attribute Cust# of relation ORDER that references relation CUSTOMER,
2. the attribute Order# of relation ORDER_ITEM that references relation ORDER,
3. the attribute Item# of relation ORDER_ITEM that references relation ITEM,
4. the attribute Order# of relation SHIPMENT that references relation ORDER, and
5. the attribute Warehouse# of relation SHIPMENT that references relation WAREHOUSE.

We now give the queries in relational algebra:
Chapter 5: The Relational Data Model and Relational Database Constraints

5.14 Consider the following relations for a database that keeps track of business trips of salespersons in a sales office:

SALESPERSON (SSN, Name, Start_Year, Dept_No)
TRIP (SSN, From_City, To_City, Departure_Date, Return_Date, Trip_ID)
EXPENSE (Trip_ID, Account#, Amount)

Specify the foreign keys for this schema, stating any assumptions you make.

Answer:

The schema of this question has the following two foreign keys:
1. the attribute SSN of relation TRIP that references relation SALESPERSON, and
2. the attribute Trip_ID of relation EXPENSE that references relation TRIP.

In addition, the attributes Dept_No of relation SALESPERSON and Account# of relation EXPENSE are probably also foreign keys referencing other relations of the database not
5.15 Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT (SSN, Name, Major, Bdate)
COURSE (Course#, Quarter, Grade)
ENROLL (SSN, Course#, Quarter, Grade)
BOOK_ADOPTION (Course#, Quarter, Book_ISBN)
TEXT (Book_ISBN, Book_Title, Publisher, Author)

Specify the foreign keys for this schema, stating any assumptions you make.

Answer:

The schema of this question has the following four foreign keys:
3. the attribute SSN of relation ENROLL that references relation STUDENT,
4. the attribute Course# in relation ENROLL that references relation COURSE,
5. the attribute Course# in relation BOOK_ADOPTION that references relation COURSE, and
6. the attribute Book_ISBN of relation BOOK_ADOPTION that references relation TEXT.

We now give the queries in relational algebra:

a. \[
\text{COSTLY\_TRIPS} = \sum_{\text{Trip}\_id} \text{SUM\_Amount > 2000 (Trip\_id \ F SUM\_Amount (EXPENSE))}
\]
RESULT \(\pi_{\text{Trip}\_id} (\text{COSTLY\_TRIPS})\)

b. \[
\text{RESULT} \pi_{\text{SSN}} (\text{To\_City = 'Honolulu' (TRIP)})
\]

c. \[
\text{RESULT} \ F \text{SUM\_Amount} (\text{ssn = '234-56-7890' (TRIP) } \ast \text{ EXPENSE})
\]
d.  COURSES_JS_W99 — Name = 'John Smith' (STUDENT) * Quarter = 'W99' (ENROLL)
     RESULT — F COUNT Course (COURSES_JS_W99)

e.  CS_ADOPTIONS — π Course, ISBN (— Dept = 'CS' (COURSE) * BOOK_ADOPTION)
     BOOK_COUNT — Course, F COUNT ISBN (CS_ADOPTIONS)
     COURSES_NEEDED — π Course (— COUNT ISBN = 2 (BOOK_COUNT))
     RESULT — π Course, ISBN, Book Title (COURSES_NEEDED * BOOK_ADOPTION * TEXT)

f.  DEPT_PUBS — π Dept, Publisher (COURSE * BOOK_ADOPTION * TEXT)
     RESULT — π Dept (COURSE) — π Dept (— Publisher = 'BC Publishing' (DEPT_PUBS))

     It is interesting to observe that, contrary to intuition, the above expression does not
     employ the DIVISION operation, despite the fact that the query involves the word all.
     If, however, the query were to list any department that has adopted all books
     published by 'BC Publishing', then the solution would be the following expression
     involving DIVISION:

     DEPT_BOOKS — π Dept, ISBN (COURSE * BOOK_ADOPTION)
     BOOKS_BY_BC — π ISBN (— Publisher = 'BC Publishing' (TEXT))
     RESULT — DEPT_BOOKS, BOOKS_BY_BC