CS 460

Lu Ye

Homework4

Question1: 5.7 Explain how a relational calculus expression can be unsafe. Illustrate your answer with an example. Discuss how to ensure that a relational calculus expression is safe.

Answer:

A example of an unsafe relational calculus expression would be {S|~Staff(S)} which means the set of all tuples that are not in the Staff relation. This expression is said to be unsafe because it generates an infinite set and there is no restriction of domain. More, it will typically include tuples from outside the Staff relation and so outside the domain of the expression. In order to make sure a relational calculus expression is safe, all values that appear in the result are values from the domain of the expression. For the example I pointed out above, we have to add a restriction that again, all values that appear in the result are values from the domain of the expression. Which means the domain of the expression is the set of all values appearing in the Staff relation.

Question 2: 5.12 Generate the relational algebra and domain relational calculus expressions for the following queries:

(b) List all single rooms with a price below £20 per night.

DRC:

$$\{< roomNo, hotelNo, type, price > | (\exists type, price)$$
$$(< roomNo, hotelNo, type, price > \in Room$$
$$\wedge type =' single' \wedge price < 20)\}$$

RA:

$$\sigma_{type='single',\ price<20}(Room)$$

(f) List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

DRC：

$$\{< roomNo, hotelNo, type, price, guestName > |$$

$$(\exists hotelNo, roomNo)(< \text{roomNo, hotelNo, type, price} > \in Room$$

$$\wedge (\exists hotelNo, roomNo)\left(\begin{array}{c}(\exists hotelNo)(< \text{hotelNo, hotelName, city} > \in Hotel) \\ \wedge (\exists guestNo, hotelNo, dateFrom, dateTo) \\ (< \text{hotelNo, guestNo, dateFrom, dateTo, roomNo} > \in \text{Booking}) \\ \wedge (\exists guestNo)(< \text{guestNo, guestName, guestAddress} > \in Guest) \\ \wedge Guest.guestNo = Booking.guestNo \wedge Booking.hotelNo = Hotel.hotelNo \\ \wedge Booking.dateFrom \leq CurrentDate \wedge Booking.dateTo \geq CurrentDate\end{array}\right.$$

$$\wedge Room.hotelNo = hotelNo \wedge Room.roomNo = roomNo)\}$$

RA:

$$\pi_{roomNo,hotelNo,type,price,guestName}$$

$$(Room \bowtie_{\substack{Room.hotelNo=hotelNo,\\ Room.roomNo=roomNo}} \left(\pi_{\substack{guestName,\\ hotelNo,\\ roomNo}} (\sigma_{\substack{Guest.guestNo=Booking.guestNo,\\ Booking.hotelNo=Hote.hotelNo,\\ Hotel.hotelName='Grosvenor',\\ Booking.dateFrom\leq CurrentDate,\\ Booking.dateTo\geq CurrentDate}} Guest, Booking, Hotel) \right))$$

Question 3: 5.16 List the names and addresses of all employees who are managers.

DRC:

$$\{< fName, IName, address > | (\exists empNo)$$

$$(< empNo, fName, lName, address, DOB, sex, position, deptNo >$$

$$\in Employee \wedge (\exists mgrEmpNo)(< deptNo, deptName, mgrEmpNo$$

$$> \in Department \wedge Emplpyee.empNo$$

$$= Department.mgrEmpNo))\}$$

RA:

$$\pi_{fName,IName,address}(Employe \bowtie_{Employee.empNo=Department.mgrEmpNo} Department)$$

Question 4: 6.11 List the bookings for which no dateTo has been specified.

Answer: select * from Booking where dataTo is null;

Question 5:6.13 What is the average price of a room?

Answer: select AVG(price) from Room;

Question 6:6.21 What is the lost income from unoccupied rooms at the Grosvenor Hotel?

Answer: select SUM(price) from Room
where roomNo not in
(select roomNo from Booking, Hotel
where (dateFrom<=CURRENTDATE and dateTo>=CURRENTDATE) and
Room.hotelNo=Booking.hotelNo and hotelName= 'Grosvenor');

Question 7:6.23 List the number of rooms in each hotel in London.

Answer: select hotelNo, COUNT(roomNo)
from Room, Hotel
where Room.hotelNo=Hotel.hotelNo and city= 'London'
group by hotelNo;

Question 8:7.5 Describe how the process of view resolution works.

Answer:

For the process of view resolution, it is actually the process to look at how a query on a view is handled. To illustrate the process of view resolution, we can base the example from book, which is asking the following query "which counts the number of properties managed by each member of staff at branch office B003". To answer this question, it is clearly to merge the query below.

SELECT staffNo, cnt
FROM StaffPropCnt
WHERE branchNo = 'B003'
ORDER BY staffNo;

Importantly, the based query is defining the corresponding column names in the SELECT list, which gives SELECT s.staffNo AS staffNo, COUNT(*) AS cnt and view names in the FROM clause are replaced with the corresponding FROM. The WHERE clause from the user query is combined with the WHERE clause of the defining query using the logical operator AND. The GROUP BY and HAVING clauses are copied from the defining query. In the last, the ORDER BY clause is copied from the user query with the view column name translated into the defining

query column name. Thus, to combine the query we can get the final merged query which is :

```
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo AND branchNo = 'B003'
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

Again, the process of combining a query on a view with its stored definition and translating it into a query on the view's underlying tables is called view resolution.

Question 9:7.7 What is a materialized view and what are the advantages of a maintaining a materialized view rather than using the view resolution process?

Answer:

Based on the definition from textbook, materialized view is a temporary table in the database when the view is first queried. Comparing to the view resolution process, maintaining a materialized view can be much faster than re-computing the view each time. Materialized views are also useful in new applications for integrity constraint checking and query optimization.

Question 10: 7.11 Now create the Room, Booking tables using the integrity enhancement features of SQL with the

Following constraints:

(a) type must be one of Single, Double, or Family.

(c) roomNo must be between 1 and 100.

(e) The same room cannot be double-booked.

```
    CREATE DOMAIN roomType AS VARCHAR2(15)
            CHECK(VALUE IN ('Single', 'Family', 'Double'));
    CREATE DOMAIN roomNumber AS INTEGER
            CHECK(VALUE BETWEEN 1 AND 100);
CREATE TABLE Room (
    roomNo roomNumber,
    hotelNo integer,
```

```
    type roomType,
    price integer,
    primary key(roomNo));

CREATE TABLE Booking(
    hotelNo integer,
    guestNo integer,
    dateFrom varchar2(15),
    dateTo varchar2(15),
    roomNo roomNumber,
    constraint bookroom
        check(not exists(select * from Booking b
                         where b.dateTo >= Booking.dateFrom
                         and b.dateFrom<=Booking.dateTo
                         and b.hotelNo = Booking.hotelNo
                         and b.roomNo = Booking.roomNo)),
    primary key(hotelNo, guestNo, dateFrom));
```

Question 11: 7.14 Create a view containing the account for each guest at the Grosvenor Hotel.

Answer:

```
    CREATE VIEW acctForGuest
    AS SELECT
    g.guestNo,g.guestName,g.guestAddress,
    h.price,b.dateFrom,b.dateTo,r.roomNo,r.type
    FROM Guest g, Hotel h, Booking.b, Room r
    WHERE g.guestNo = b.guestNo AND r.roomNo = b.roomNo
    AND b.hotelNo=h.hotelNo AND h.hotelName= 'Grosvenor';
```

Question 12: 8. 6 What are database triggers and what could they be used for?

Answer:

A trigger defines an action that the database should take when some event occurs in the application. Triggers are based on the Event–Condition–Action (ECA) model

which including the content:

Event - causes trigger activation, which can be an INSERT, UPDATE, or DELETE statement on a specified table (or possibly view).

Condition – if true, action will be performed.

Action - SQL statements to be executed.

There are two types of triggers; one is row-level triggers (FOR EACH ROW) that execute for each row of the table that is affected by the triggering event, and statement-level triggers (FOR EACH STATEMENT) that execute only once even if multiple rows are affected by the triggering event. Triggers are a more advanced form of constraint enforcement, it mainly be used to log activities on key tables.

Question 13: 8.11 Create a database trigger for the following situations:

(a) The price of all double rooms must be greater than £100.

Answer:

```
create trigger drPrice
after insert on Room
for each row
when (new.type = 'double' and new.price > 100)
    begin
    raise_application_error (-20000, 'price must be greater than 100');
    end drPrice;
```

Question 14: Consider this schema: R(A, B, C, D, E, F, G). Also consider the FDs B → ACDE and E → F G. Compute both B+ and G+.

FD	B+	temp
-	{B}	{B}
B→ACDE	{B,A,C,D,E}	{B,A,C,D,E}
E→FG	{B,A,C,D,E,F,G}	{B,A,C,D,E,F,G}

Thus, B+ is {B,A,C,D,E,F,G}.

FD	G+	Temp
-	{G}	{G}

G+={G}

Question 15: Given the set of FDs F = {A → D, AC → B, C → D, A → BC}, find a minimal cover of F.

1. Put FDs in standard form

   Decomposition A→BC into A→B and A→C

Now F={A→ D, AC→ B, C→ D, A→ B, A→ C}

2. Minimize the LHSes of the FDs

   Consider AC→B

   (1) A→B [Given]

   (2) AC→B [1, Aug(b)]

Because A→B can become AC→B, we can remove AC→B.

3. Remove redundant FDs.

   Consider A→D via transitivity A→C and C→D can form A→D

   Such that A→D can be removed.

Therefore, the final result should be F = {C→D,A→B,A→C}.