CS–A1150 Databases
Exam Sep 3rd, 2020

1. a) (8 p) Based on the description below, construct a UML diagram for an insurance company to model its customers, insurance policies it has sold, damages, and compensation claims. Use the notation used in the course text book and mark the key attributes like they are marked in the course text book (and in the lecture slides, too). Your diagram does not include such parts of the database which have not been described below.

   Each customer has a unique ID, a name, an address and e-mail address. One customer may have several insurance policies and the same insurance policy may have several owners (for example, a couple may own together their home insurance). Each insurance policy has a unique ID, the type of the insurance (car or home, for example) and a yearly payment.

   In addition, the database includes information about damages and compensation claims connected to them. Each damage is connected with exactly one insurance policy. In addition, the company wants to store a unique ID, the date, the type of the damage, and the compensation claims connected to the damage. Several compensation claims may be connected to the same damage (for example an illness may cause different types of costs during several years). For each compensation claim, the information stored includes also the date of the claim, the sum the customer asks to be paid, and the decision (the sum the company will pay).

   b) (2 p) Convert the UML diagram from part (a) into relations. Write the relation schemas and underline the names of the key attributes.

2. Consider the following database schema, which contains information about customers, hotels, their rooms and reservations of a hotel chain. For the sake of simplicity, we assume that each reservation contains only one room and the room is always assigned immediately when the reservation is made. We also assume that the price of a certain room does not vary, i.e. the same room has always the same price.

   The relation `Hotels` contains information about the hotels belonging to the chain. The hotels are identified by the attribute `ID`. Relation `Customers` contains information about the customers of the hotel chain. The customers are identified by unique customer number (attribute `CID`). The relation may also contain customers who have made no reservation yet, for example customers who have taken part in some advertising campaign.

   The relation `Rooms` contains information about the rooms in the hotels of the chain. Attribute `number` is the number of the room, `hotelID` is the ID of the hotel, `beds` is the number of beds in the room and `price` is the price of the room for one night.

   The relation `Reservations` contains information about the reservations. The reservations are identified by unique `RID`. The attributes of the relation also tell the customer number of the customer who has made the reservation, the room number and the ID of the hotel, the date of check-in and the number of the consecutive nights included in the reservation. Each reservation contains always only one room.

   The number of the room, the number of the beds and the number of the nights included in the reservation are integers and the price is a decimal number. The values of all other attributes are strings. You may assume that the attributes of the tuples do not have NULL values.

   Database schema:

```
Hotels(ID, name, city, address, country)
Customers(CID, name, email, phone)
Rooms(number, hotelID, beds, price)
Reservations(RID, customerID, hotelID, roomno, date, numberofnights)
```

   Write the following SQL queries:

   a) (2 p) The IDs and names of the hotels which have at least one room with price under 60 euros.

   b) (2 p) The customer numbers and names of those customers who have made at least one reservation which has the total price (the number of nights times the price of the room) of over 1000 euros.

   c) (2 p) The IDs and names of those hotels which have at least one room with 3 beds, but no rooms with 4 or more beds.

   d) (2 p) We want to find the hotels in Finland which can accommodate at least 100 guests (consider the total number of beds in the rooms of the hotel). For each such hotel, the query must produce the hotel ID, the name and the average of the prices of all rooms in the hotel. Note that the average price and the total number of the beds are calculated separately for each hotel.

Explain which query the following expressions of Relational Algebra produce the answer (for example: "The expression produces names and IDs of hotels which are located in Finland".)

e) (2 p)

$$\pi_{CID,name}(\texttt{Customers}) - \pi_{CID,name}(\texttt{Customers} \bowtie_{CID=customerID} (\texttt{Reservations}))$$

f) (2 p)

$$\pi_{R1.hotelID,R1.number}(\rho_{R1}(\texttt{Rooms}) \bowtie_{R1.price>R2.price \texttt{ AND } R1.hotelID=R2.hotelID} (\rho_{R2}(\texttt{Rooms})))$$

3. (4 p) Consider a database which is too large to fit in the main memory. Assume that this database contains Table (relation) R(A, B, C, D) which occupies hundreds of disk pages. The primary key of R is A. However, the queries which search for tuples (rows) where C has a value given by a user are also quite common. The value of C to be searched for is not the same every time the query is executed, but varies.

   a) The course material tells that in this case it is profitable to make an index for Table R on attribute C, if relatively few tuples have the same value for attribute C. If very many tuples have the same value for attribute C, the index is usually not profitable. Why is it so? Explain this briefly by using a small example. Exact expressions or calculations are not needed.

   b) There is one exception which makes the index profitable even in the case where many tuples have the same value for attribute C. What is it and why is the index then profitable?

4. Consider a relation R with schema $R(A, B, C, D, E)$ and functional dependencies $A \rightarrow B$, $A\ C \rightarrow D$, and $E \rightarrow B$.

   a) (1 p) Explain why this relation is not in Boyce-Codd normal form (BCNF).

   b) (6 p) Decompose the relation using the BCNF decomposition algorithm taught in this course and in the text book. Give a short justification for each new relation. Continue the decomposition until the final relations are in BCNF. Explain why the final relations are in BCNF.

5. Consider the relation RoomBooking(event, room, date, numberOfPeople). It is part of a larger database back-end for an organizer of company events, and tracks room reservations for a specific event. `event` is the event ID, `room` is the room ID, and `date` the date of the reservation (for simplicity let this be full day events only), while `numberOfPeople` is an integer value tracking the number of registered attendants for that particular booking.

   Assume that for the date '2020-09-03' there is only a single tuple present in the table:

   `('S-20', 'A2', '2020-09-03', 75)`

   At this stage there are two partially *concurrent transactions*:

   T1 with the following operations

   1. Update tuple with key `('S-20', 'A2', '2020-09-03')` setting `numberOfPeople` to 88,
   2. Calculate the total number of people for all rooms and events on date '2020-09-03',
   3. If total $>$ 200 then **rollback** (i.e. change the value of `numberOfPeople` of the updated tuple back to 75 and end the transaction) else **commit**.

   T2 with the following operations

   1. Insert a new tuple `('S-21', 'B5', '2020-09-03', 120)` into RoomBooking,
   2. Calculate the total number of people for all rooms and events on date '2020-09-03',
   3. If total $>$ 200 then **rollback** (i.e. remove the newly inserted tuple from RoomBooking and end the transaction) else **commit**.

   The operations within a particular transaction happens in the order given, and a single operation is considered atomic.

   It is not known in which order the transactions start or finish in relation to each other, only that they can be considered concurrent.

   No triggers or constraints which would affect the execution of **T1**, and **T2** are active on the database. There are no other active transactions which would update the data for date '2020-09-03'.

   Now, assume that transaction **T1** suffer an interruption between steps 2 and 3, while transaction **T2** completes all of its steps. Provide answers for the following three (independent) scenarios:

a) (3 p) Let the **ACID** constraints hold for the database transactions. Which of the following end results are possible for date '2020-09-03' after T1 and T2 are executed? Give the letter or letters corresponding to all options that apply as your answer.

- **X** Room A2 has 88 people; There is no entry for Room B5
- **Y** Room A2 has 88 people; Room B5 has 120 people
- **Z** Room A2 has 75 people; There is no entry for Room B5
- **W** Room A2 has 75 people; Room B5 has 120 people

b) (2 p) Let the **ACID** properties, *except for atomicity*, hold for the database transactions. Which of the following end results are possible for date '2020-09-03' after T1 and T2 are executed? Give the letter or letters corresponding to all options that apply as your answer.

- **X** Room A2 has 88 people; There is no entry for Room B5
- **Y** Room A2 has 88 people; Room B5 has 120 people
- **Z** Room A2 has 75 people; There is no entry for Room B5
- **W** Room A2 has 75 people; Room B5 has 120 people

c) (2 p) Let the **ACID** properties hold otherwise, *except that isolation is not guaranteed*, for the database transactions. Which of the following end results are possible for date '2020-09-03' after T1 and T2 are executed? Give the letter or letters corresponding to all options that apply as your answer.

- **X** Room A2 has 88 people; There is no entry for Room B5
- **Y** Room A2 has 88 people; Room B5 has 120 people
- **Z** Room A2 has 75 people; There is no entry for Room B5
- **W** Room A2 has 75 people; Room B5 has 120 people