Database Systems - Homework 1

Yosef Goren

May 15, 2023

Contents

1 ERD

1.1

The tables are:

1. Policy($\underline{\text{Number}}$, $\underline{\text{Company}}$, Expiration)
2. Person($\underline{\text{ID}}$, Name)
3. Person.Record($\underline{\text{ID}}^F K$, Record)

4. Vehicle(PlateNumber, Manufacturer)
5. Car(PlateNumber, Type)
6. Bus(PlateNumber, Operator)
7. Insurance(PlateNumber^{FK}, Number^{FK}, Company^{FK}, Agent)
8. Drive(ID^{FK}, PlateNumber^{FK})
9. Accident(front.PlateNumber^{FK}, back.PlateNumber^{FK}, Location, Data)

1.2

1. Possible. It would require that the Insurance table would not contain any entiries with a PlateNumber key that equals to that of the car.
2. Possible. While a Person relates to precisely one Vechicle - a Vechicle (Bus) might not be related to any person.
3. Possible. There is no constraint on how many different Persons drive the same Car, only the other way around.

1.3

The phrasing on the question suggests exactly one officer arrives at each Accident so I will assume that is the case.
Additionally - no information was specified as for how different officers are identified, hence I will assume each officer can be fully and uniquely described with a key called 'OfficerID'.

Under these assumptions, the required information can be added by appending two attributes to the Accident relation: 'OfficerID' and 'OfficerArrivalTime'. After these changes, the table represeting accidents will look like:
Accident(front.PlateNumber^{FK}, back.PlateNumber^{FK}, Location, Data, OfficerID, OfficerArrivalTime)

1.4

Alice is correct. Since in both the solution of Bob and that of Charlie there is a unique connection going from the front accident participant to the back accident participant; This means that any Vechicle which participates in an accident as the front participant must participate in accidents as the front participant which **exactly one** other car [1]. The problem with that is the fact that this means accidents in which multiple cars crash to the back of a single car (such as in the example before) cannot be described by the scheme.

[1] which will be the back participant in that context

2 RA

To prove that the π operator is independent of the other operators σ, ρ, $\setminus$, $\times$, and $\cup$, we need to show that applying any combination of these operators in any order produces the same result as applying the π operator first and then applying the remaining operators.

Let R be a relation, and let A be a subset of the attributes of R. Then the π operator selects only those tuples of R that have the specified attributes in A, and discards all other attributes. Now, let's consider the other operators:

- σ selects only those tuples that satisfy a specified condition.
- ρ renames a relation or its attributes.
- $\setminus$ returns tuples that are in the first relation but not in the second relation.
- $\times$ returns the Cartesian product of two relations.
- $\cup$ returns all distinct tuples that are in either of two relations.

We can see that the π operator only operates on the attributes of a relation, and does not consider the tuples themselves. Therefore, applying any combination of the other operators in any order will not affect the attributes selected by the π operator.

In other words, the π operator can be applied before or after any combination of the σ, ρ, $\setminus$, $\times$, and $\cup$ operators without affecting the result. Hence, the π operator is independent of these operators.

3 RA,RC

3.1

1. $$\pi_{Name}(\sigma_{Company=BIG}(Person \bowtie Drive \bowtie Insurance))$$

   Note how $Person \bowtie Drive$ Is uniquely identified by ID since a person cannot have more than one car, hence ID is uniquely related to some PlateNumber so far.

   In addition an entity of type Vehicle must relate to at most one entity of Policy, so a PlateNumber is uniquely related to some Number (from Policy).

   Thus in $Person \bowtie Drive \bowtie Insurance$ - each ID is uniquely related to a Number.

2. $$\{n_{Name} \mid \exists (i_{ID}, p_{PlateNumber}, n_{Number}, a_{Agent}) :$$
$$Person(i_{ID}, n_{Name}) \wedge Drive(i_{ID}, p_{PlateNumber}) \wedge Insurance(p_{PlateNumber}, n_{Number}, BIG, a_{Agent})\}$$

3.2

3.3

1. For readablility define:

$$AccidentDates = \pi_{Time}(Accident)$$

   And the answer is:

$$(AccidentDates \times AccidentDates) \setminus AccidentDates$$

3.4

1. The query returns the set of agenets that work with insurance companies which insure every Manufacturer in the database.
2. That would not be possible since this query can be reduced to the '$\div$' which can be further reduced to the $\setminus$ operator (which we know is independent).

3.5

4 Datalog

4.1

```
CinemasShowingTarantinoMovie(Theater, Address) ←
    Movies(Title, "Tarantino", _),
    Cinemas(Theater, Address, _),
    Schedule(Theater, Title, _).
```

4.2

```
ActorPlayingMovie(Title, Actor) ← Movies(Title, _, Actor).

TarantinoMovieNotActing(Title) ←
    Movies(Title, "Tarantino", _),
    not ActorPlayingMovie(Title, "Tarantino").
```

4.3

```
PlayedTarantinoMovie(Actor) ← Movies(_, "Tarantino", Actor).

NeverPlayedForTarantino(Actor) ←
    Movies(_, _, Actor),
    not PlayedTarantinoMovie(Actor).

AnyoneNeverPlayedTarantino(Title) ←
    Movies(Title, _, Actor),
    NeverPlayedForTarantino(Actor).

AllTarantinoActors(Title) ←
    Movies(Title, _, _),
    not AnyoneNeverPlayedTarantino(Title).
```

4.4

```
NotDirectorOfAllActors(Director) ←
    Movies(_, _, Actor)
    not Movies(_, Director, Actor).

DirectorOfAllActors(Director) ←
    Movies(_, Director, _),
    not NotDirectorOfAllActors(Director).
```

4.5

```
Actor1PlayedWithout2(Actor1, Actor2, Title) ←
    Movies(Title, _, Actor1),
    not Movies(Title, _, Actor2).

Actor1NeverPlayedWithout2(Actor1, Actor2) ← not Actor1PlayedWithout2(Actor1, Actor2, Title)

ActorsInSameMovies(Actor1, Actor2) ←
    Actor1NeverPlayedWithout2(Actor1, Actor2),
    Actor1NeverPlayedWithout2(Actor2, Actor1).
```

4.6

```
Related(Actor1, Actor2) ←
    Movies(Title, _, Actor1),
    Movies(Title, _, Actor2).
Related(Actor1, Actor2) ←
    Related(Actor1, Actor),
    Related(Actor, Actor2).
```

```
NotRelated(Actor1, Actor2) ←
    Movies(_, _, Actor1),
    Movies(_, _, Actor2),
    not Related(Actor1, Actor2).
```