Databases - Exercise 6:
Relational Algebra

Sample solution

13 December 2019

1 Music domain

Assume that we have the following schema.

Artists(name)
Bands(name)
    name → Artists(name)
Persons(name, birthyear)
    name → Artists(name)
Albums(artist, title, year)
    artist → Artists(name)
Songs(artist, title, length)
    artist → Artists(name)
Tracks(album, song)
    album → Albums(title)
    song → Songs(title)
Members(person, band)
    person → Persons(name)
    band → Bands(name)

1.1 Finding length of certain album

Write a query, in SQL and in relational algebra, to find total length of Amy MacDonald's Album "This is the life". The total length means the total length of all songs on that album, assuming, for simplicity, that lengths are expressed in seconds as an integer). The query output is of the following format:

```
sum
-----
3528
```

Solution

In SQL:

```
SELECT SUM(length)
FROM Songs, Tracks
WHERE album = 'This is the life' AND artist = 'Amy MacDonald'
    AND Songs.title = Tracks.song;
```

In relational algebra:

$$\pi_{\mathrm{SUM(length)}}\gamma_{\mathrm{SUM(length)}}($$
$$\sigma_{\mathrm{album='Thisisthelife'\wedge artist='AmyMacDonald'\wedge Songs.title=Tracks.song}}($$
$$\mathrm{Songs} \times \mathrm{Tracks}))$$

1.2 Finding album by certain artist from certain year

Write an SQL query and a relational algebra expression to find all albums by Amy MacDonald from year 2016 or later, with their years of appearance. The format of the output is:

```
  title              | year
---------------------+------
  Under Stars        | 2017
  Woman of the world | 2018
```

Solution

In SQL:

```
SELECT title, year
FROM Albums
WHERE artist = 'Amy MacDonald' AND year >= 2016;
```

In relational algebra:

$$\pi_{\mathrm{title,year}}(\sigma_{\mathrm{artist='AmyMacDonald'\wedge year>=2016}})(\mathrm{Albums})$$

1.3 Finding all solo albums

Write an SQL query and a relational algebra expression to find all solo albums, i.e., albums whose artist is not a band but a person. The format of the output is:

```
  title
--------------------------
  Under Stars
  Woman of the world
```

Solution

In SQL:

```
SELECT title
FROM Albums
WHERE EXISTS (SELECT name FROM Persons WHERE name=artist);
```

In relational algebra:

$$\pi_{\text{title,artist,year,name}}\sigma_{\text{name=artist}}(\text{Albums} \times \text{Persons})$$

2 Fligths domain

Let's assume we have the following schema.

Airports(code, city)

FlightCodes(code, airlineName)

Flights(departureAirport, destinationAirport, departureTime, arrivalTime, code)

departureAirport → Airports.code
destinationAirport → Airports.code
code → FlightCodes.code

2.1 Translate an SQL query to a relational algebra expression

Below you can find an SQL query that finds all airports that have departures or arrivals (or both) of flights operated by Lufthansa or SAS (or both), based on the schema about flights. Express this query by a relational algebra expression.

```
SELECT DISTINCT served
FROM ( (SELECT destinationAirport AS served, airlineName
        FROM FlightCodes NATURAL JOIN Flights)
       UNION
       (SELECT departureAirport AS served, airlineName
        FROM FlightCodes NATURAL JOIN Flights)
     ) WHERE airlineName='Lufthansa' OR airlineName = 'SAS';
```

Solution

$\delta(\pi_{\mathrm{served}}(\sigma_{\mathrm{airlineName}='\mathrm{Lufthansa}'\vee\mathrm{airlineName}='\mathrm{SAS}'}($
$\rho_{(\mathrm{served},\mathrm{airlineName})}(\pi_{\mathrm{destinationAirport},\mathrm{airlineName}}(\mathrm{FlightCodes} \bowtie \mathrm{Flights}))$
$\cup$
$\rho_{(\mathrm{served},\mathrm{airlineName})}(\pi_{\mathrm{departureAirport},\mathrm{airlineName}}(\mathrm{FlightCodes} \bowtie \mathrm{Flights}))$
$)))$

2.2 Translate a relational algebra expression to an SQL query

Translate the following relational algebra expression to an SQL query.

$\pi_{\mathrm{First.departureTime},\mathrm{Second.arrivalTime}}$
$((\rho_{\mathrm{First}}(\mathrm{Flights})) \bowtie_{\mathrm{First.destinationAirport}=\mathrm{Second.departureAirport}} (\rho_{\mathrm{Second}}(\mathrm{Flights})))$

Solution

In SQL:

```
SELECT First.departureTime, Second.arrivalTime
FROM Flights AS First JOIN Flights AS Second
ON First.destinationAirport = Second.departureAirport
```