Name

CWID

Final Exam

May 4th, 2022
8:00-10:00

CS520 - Data Integration, Warehousing, and Provenance

Results

Please leave this empty!

1.1		1.2		1.3		Sum	

Part 1.1 Provenance (Total: 30 Points)

For each of the queries shown in the following compute the provenance of all of their result tuples produced over the database shown below. Calculate provenance for these provenance models:

- Minimal Why-Provenance
- Provenance Polynomials

List all query result tuples and show the provenance on the right for each query result tuple.

Consider the following database schema and instance:

city

name	population	state	
Chicago	3200000	IL	c_1
Schaumburg	70000	IL	c_2
Evanston	120000	IL	c_3
Seattle	800000	WA	c_4
Austin	2000000	TX	c_5

connection

from	to	connectiontype	price	miles	
Chicago	Seattle	flight	540	2000	n_1
Chicago	Austin	flight	430	1500	n_2
Chicago	Austin	bus	80	1500	n_3
Chicago	Schaumburg	bus	5	2000	n_4
Chicago	Seattle	train	250	2000	n_5
Schaumburg	Evanston	bus	15	2000	n_6
Austin	Seattle	flight	890	2000	n_7

Question 1.1.1 (9 Points)

$$\pi_{to}(\sigma_{price<100 \wedge from=Chicago}(connection))$$

Solution

Minimal Why:

to	
Austin	$\{\{n_3\}\}$
Schaumburg	$\{\{n_4\}\}$

Provenance Polynomials:

to	
Austin	n_3
Schaumburg	n_4

Question 1.1.2 (10 Points)

$$q_1 \stackrel{def}{=} \pi_{name}(\sigma_{state=IL}(city))$$
$$q \stackrel{def}{=} \pi_{from,to}(q_1 \bowtie_{name=from} connection \bowtie_{to=name} q_1)$$

Solution

Minimal Why provenance:

from	to	
Chicago	Schaumburg	$\{\{c_1, n_4, c_2\}\}$
Schaumburg	Evanston	$\{\{c_2, n_6, c_3\}\}$

Provenance Polynomials:

from	to	
Chicago	Schaumburg	$c_1 \cdot n_4 \cdot c_2$
Schaumburg	Evanston	$c_2 \cdot n_6 \cdot c_3$

Question 1.1.3 (11 Points)

$$q_1 \stackrel{def}{=} \pi_{from,to}(\sigma_{to=Seattle}(connection))$$

$$q_2 \stackrel{def}{=} \pi_{from,to}(\sigma_{middle=Austin}(\rho_{middle \leftarrow to}(connection) \bowtie \rho_{middle \leftarrow from}(connection)))$$

$$q \stackrel{def}{=} q_1 \cup q_2$$

Solution

Minimal Why provenance:

from	to	
Chicago	Seattle	$\{\{n_1\}, \{n_5\}, \{n_2, n_7\}, \{n_3, n_7\}\}$
Austin	Seattle	$\{\{n_7\}\}$

Provenance Polynomials:

from	to	
Chicago	Seattle	$n_1 + n_5 + n_2 \cdot n_7 + n_3 \cdot n_7$
Austin	Seattle	n_7

Part 1.2 Data Warehousing (Total: 35 Points)

Recall that you should write all queries according to the schema and not according to the example instance.

Consider the following datawarehouse schema (star schema) and partial example instance. There is a single fact table (`sales`) about sales of items. Each row in this fact table stores the quantity of a certain product (e.g., 3 Samson Galaxy phones) sold at a particular location and time to a particular customer. There are four dimension tables corresponding to the following dimensions:

- **Time** with three levels (year, month, day)
- **Location** with four levels (state, city, zip, street)
- **Customer** with one level (name).
- **Product** with three levels (category, brand, pname, price) where pname is the finest granularity and brand and category are not comparable (some brands can have products from multiple categories and categories obviously can contain have products from different brands). The same holds for price and brand and price and category.

sales

TID	LID	CID	PID	numItems
1	4	1	1	15
2	1	5	2	10
100	1	76	4	22
...	...	...	...	...

timeDim

TID	year	month	day
1	2010	1	1
2	2010	1	2
...	...	...	...
...	2018	5	1

customerDim

CID	cname
1	Noekig
2	Prokig
...	...

locationDim

LID	state	city	zip	street
1	Illinois	Chicago	60616	10 W 31st
2	Illinois	Chicago	60615	900 Cottage Grove
3	Lousiana	New Orleans	42345	12 Mark street
...	...	...	...	...

productDim

PID	category	brand	pname	price
1	computers	Apple	MacBook	1300
2	computers	Dell	Inspire	1000
3	smartphones	Samsung	Galaxy 1	600
...	...	...	...	...

Hints:

- Attributes with black background form the primary key of a relation (e.g., `PID` for relation `productDim`)
- Attributes `LID`, `TID`, `PID`, and `CID` in the fact table are foreign keys to the dimension tables

Question 1.2.1 (8 Points)

Write an SQL query that returns for each year a breakdown of the total revenue for each level of the location dimension. The revenue of a sale is the number of items (`numItems`) multiplied by product's price (`price`).

Solution

```
SELECT sum(numItems * price) AS revenue,
       year,
       state,
       city,
       zip,
       street,
       GROUPING(state),
       GROUPING(city),
       GROUPING(zip),
       GROUPING(street)
  FROM sales s, locationDim l, productDim p, timeDim t
 WHERE s.LID = l.LID AND s.PID = p.PID AND s.TID = t.TID
GROUP BY year, ROLLUP(state,city,zip,street);
```

Question 1.2.2 (9 Points)

Write an SQL query that returns the top-3 states with the highest average of the yearly total number of products sold in this state.

Solution

```
SELECT avg(totalitems) AS avgyearly, state
FROM (SELECT sum(numItems) AS totalitems, state, year
       FROM sales s, locationDim l, timeDim t
      WHERE s.LID = l.LID AND s.PID = p.PID AND s.TID = t.TID
      GROUP BY state, year)
ORDER BY avgyearly DESC
LIMIT 3;
```

Question 1.2.3 (9 Points)

Write an SQL query that returns cities with at least 3 times the number of items sold in this city than the average number of items sold in cities in the same state.

Solution

```
WITH cityitems AS (
  SELECT sum(numItems) as totalitems, city, state
    FROM sales s, locationDim l
   WHERE s.LID = l.LID
   GROUP BY city, state),

  avgstateitems AS (
    SELECT avg(totalitems) avgitems, state
      FROM cityitems
     GROUP BY state)

SELECT city, totalitems
  FROM cityitems c,
       avgstateitems a
 WHERE c.state = a.state
       AND totalitems >= 3 * avgitems;
```

Question 1.2.4 (9 Points)

Write an SQL query that returns all cities. For each city report its rank in terms of the total revenue (number of items (`numItems`) multiplied by the product `price`) produced by products sold in the city in 2022 compared to all other cities in the same state in 2022. Order the results by the city's revenue in decreasing order. For the ordering the state should be ignored.

Solution

```
SELECT city,
       rank() OVER (PARTITION BY state ORDER BY revenue DESC) AS rank,
       revenue
  FROM (SELECT sum(numItems * price) AS revenue, city, state
          FROM sales s, locationDim l, timeDim t
         WHERE s.TID = t.TID
           AND s.LID = l.LID
           AND t.year = 2022
         GROUP BY city, state) cityrevs
ORDER BY revenue DESC;
```

Part 1.3 Virtual Data Integration (Total: 35 Points)

Consider the following global schema and LAV views defining the content of local sources.

- **Artist**(name,age)
- **Song**(title,length,writtenby)
- **AlbumSong**(songtitle,albumtitle,nr)
- **Album**(title,price,genre,recordedby)

```
v1(Name,Age,Genre) :- artist(Name,Age), album(AT,P,Genre,Name), Age < 50.

v2(AlbumTitle,Performer) :- album(AlbumTitle,P,G,Performer), G = jazz.

v3(AlbumTitle) :- album(AlbumTitle,P,G,Performer), P > 30.

v4(SongTitle,AlbumTitle,WrittenBy) :- albumsong(SongTitle,AlbumTitle,N),
                                      song(SongTitle,L,WrittenBy),
                                      artist(WrittenBy,A).
```

Question 1.3.1 (35 Points)

Rewrite the following query using the bucket algorithm with the views given above. First write down the content of the buckets, then write down every candidate rewriting based on the buckets and demonstrate whether it is a contained rewriting or not, and then write down the maximally contained UCQ rewriting for the query.

```
q(Writer,Performer,Age,Title) :- artist(Performer,Age),
                                 album(T,P,G,Performer),
                                 albumsong(Title,T,N),
                                 song(Title,L,Writer).
```

Solution

First we need to create a bucket for every goal and put in each bucket the views which have the goal's relation in their body and return the head variables of the query coming from this goal.

artist(Performer,Age)	album(T,P,G,Performer)	albumsong(Title,T,N)	song(Title,L,Writer)
v1(Performer,Age,X1)	v1(Performer,X2,X3) v2(X4,Performer)	v4(Title,X6,X7)	v4(Title,X8,Writer)

There are three possible combinations of views to cover the goals according to the buckets.

Option 1:

We would have to add an additional equality predicate for the repeated variable `T` (album title). However, the view we use to cover goal `album(T,P,G,Performer)`, does not return the album title, so this is not possible. Thus, option 1 can not be extended into a contained rewriting, no matter which additional equality constraints we add.

```
q1(Writer,Performer,Age,Title) :- v1(Performer,Age,X1), v1(Performer,X2,X3),
                                  v4(Title,X6,X7), v4(Title,X8,Writer).
```

Option 2:

Note that we have to add the equality predicate `X4=X6` to simulate the repeated variable `T`. Optionally, we can equate `X7` with `Writer`, but that is not required for containment.

```
q2a(Writer,Performer,Age,Title) :- v1(Performer,Age,X1), v2(X4,Performer),
                                   v4(Title,X6,X7), v4(Title,X8,Writer), X4=X6.
```

or equivalently

```
q2(Writer,Performer,Age,Title) :- v1(Performer,Age,X1), v2(X4,Performer),
                                  v4(Title,X4,X7), v4(Title,X8,Writer).
```

To prove containment, we have to replace the views with their definition (renaming existentially quantified variables) and then find a containment mapping from `q` to the expanded view.

```
q2(Writer,Performer,Age,Title) :- artist(Performer,Age), album(Y1,Y2,X1,Performer), Age < 50,
                                  album(X4,Y3,Y4,Performer), Y4 = jazz,
                                  albumsong(Title,X4,Y5),
                                  song(Title,Y6,X7),
                                  artist(X7,Y7),
                                  albumsong(Title,X8,N),
                                  song(Title,Y8,Writer),
                                  artist(Writer,A).
```

Since there exists a containment mapping Ψ as shown below, `q2` is a contained rewriting. And because this is the only contained rewriting, it is also a maximally contained rewriting.

$$\begin{array}{ccc} \Psi(Writer) \to Writer & \Psi(Performer) \to Performer & \Psi(Age) \to Age \\ \Psi(Title) \to Title & \Psi(T) \to X4 & \Psi(P) \to Y3 \\ \Psi(G) \to Y4 & \Psi(N) \to Y5 & \Psi(L) \to Y8 \end{array}$$

Applying this containment mapping to the goals of `q` we get:

$$\begin{aligned} \Psi(artist(Performer, Age)) &= artist(Performer, Age) \\ \Psi(album(T, P, G, Performer)) &= album(X4, Y3, Y4, Performer) \\ \Psi(albumsong(Title, T, N)) &= albumsong(Title, X4, Y5) \\ \Psi(song(Title, L, Writer)) &= song(Title, Y8, Writer) \end{aligned}$$