Name	CWID

Final Exam

May 3rd, 2018
10:30-12:30

CS520 - Data Integration, Warehousing, and Provenance

Results

Please leave this empty!

1.1		1.2		1.3		Sum	

Instructions

- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**

Consider the following datawarehouse schema (star schema) and partial example instance. There is a single fact table (`warehouse`) about items stored in a warehouse. Each row in this fact table stores the quantity of a certain product (e.g., 3 Samson Galaxy phones) stored at a particular location and time. For each such set of products we also record which supplier did supply the product. 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)
- **Supplier** with two levels (type, sname)
- **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.

warehouse

TID	LID	SID	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

locationDim

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

suppliedDim

SID	type	sname
1	electronics	Nokiso
2	houseware	Saemens
...	...	...

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

Part 1.1 Data Warehousing (Total: 40 Points)

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

Question 1.1.1 (6 Points)

Write an SQL query that returns the 3 pairs of year and state with the highest total number of products in the warehouses in that state during that year. Note that we are dealing with snapshot facts here, since the fact table records for every TID a snapshot of the various warehouses recorded in the database and an item may contribute to multiple snapshots.

Solution

```
SELECT max(stateItems) AS ttlProc , state , year
FROM
     (SELECT sum(numItems) stateItems , TID, state , year
     FROM warehouse w NATURAL JOIN timeDim t NATURAL JOIN locationDim l
     GROUP BY state , TID, year)
GROUP BY year , state
ORDER BY ttlProd DESC
LIMIT 3
```

Question 1.1.2 (6 Points)

Write an SQL query that returns the total value (the number of items multiplied by the price) of products per supplier for all Apple products stored in warehouses at January 1st of 2018.

Solution

```
SELECT sum(numItems * price), sname
FROM warehouse f
     NATURAL JOIN
     (SELECT PID, price FROM productDim WHERE brand = 'Apple') c
     NATURAL JOIN
     (SELECT TID FROM timeDime WHERE year = 2018 AND month = 1 AND day = 1) t
     NATURAL JOIN
     (SELECT LID FROM locationDim) l
     NATURAL JOIN
     (SELECT SID, sname FROM supplierDim o) l
GROUP BY sname
```

Question 1.1.3 (7 Points)

Write an SQL query that returns the number of products (numItems) in total, per state, per city, and per zip code stored in warehouses at January 1st of 2018.

Solution

```
SELECT sum(numItems), state, city, zip,
       GROUPING(state) AS gstate, GROUPING(city) AS gcity, GROUPING(zip) AS gzip
FROM warehouse f
     NATURAL JOIN
     (SELECT TID FROM timeDime WHERE year = 2018 AND month = 1 AND day = 1) t
     NATURAL JOIN
     (SELECT LID, state, city, zip FROM locationDim) l
GROUP BY ROLLUP (state, city, zip);
```

Question 1.1.4 (7 Points)

Write an SQL query that returns for each year the change in the average of the total number of items in all warehouses. Return the year, the difference in average to the previous year, the average for this year, and the average for the following year.

Solution

```
WITH ttl AS (
    SELECT sum(numItems) AS ttlItems, year, TID
    FROM warehouse f
         NATURAL JOIN
         (SELECT TID, year FROM timeDime) t
         GROUP BY TID, year
),
yavg AS (
    SELECT avg(ttlItems) AS avgTotal, year
    FROM ttl
    GROUP BY year
)
sibAvg AS (
    SELECT
        first_value(avgTotal) OVER (ORDER BY year
                                    ROWS BETWEEN 1 PRECEDING
                                    AND 1 FOLLOWING) AS prevYear,
        avgTotal AS curYear,
        last_value(avgTotal) OVER (ORDER BY year ROWS
                                   BETWEEN 1 PRECEDING
                                   AND 1 FOLLOWING) AS nextYear,
        year
    FROM yavg
)
SELECT year, curYear - prevYear AS prevDiff, curYear, nextYear - curYear AS nextDiff
FROM sibAvg;
```

for the last part we would also accept nextYear instead of nextYear − curYear because the question was worded misleadingly

Question 1.1.5 (7 Points)

Write an SQL query that returns the number of months during which the average of the total products in warehouses is more than 100,000.

Solution

```
WITH ttl AS (
    SELECT sum(numItems) AS ttlItems, year, TID
    FROM warehouse f
         NATURAL JOIN
         (SELECT TID, year, month FROM timeDime) t
         GROUP BY TID, year, month
),
avgttl AS (
    SELECT avg(ttlItems), year, month
    FROM ttl
    GROUP BY year, month
    HAVING avg(ttlItems) > 100000
)
SELECT count(*) FROM avgttl;
```

Question 1.1.6 (7 Points)

Write an SQL query that returns the three cities with the highest average of the maximum of the total number of items per year.

Solution

```
WITH ttl AS (
    SELECT sum(numItems) AS ttlItems, year, TID, city
    FROM warehouse f
         NATURAL JOIN
         (SELECT TID, year FROM timeDime) t
         NATURAL JOIN
         (SELECT LID, city FROM locationDim) l
   GROUP BY TID, year, city
),
yavg AS (
    SELECT city, avg(maxTotal) AS avgttl
    FROM
        (SELECT max(ttlItems) AS maxTotal, year, city
        FROM ttl
        GROUP BY year, city) ymax
    GROUP BY city
)
SELECT *
FROM yavg
ORDER BY avgttl DESC
LIMIT 3;
```

Part 1.2 Big Data (Total: 30 Points)

Question 1.2.1 (12 Points)

Consider a dataset of key-value pairs `(ssn,state)` recording SSN of taxpayers and the state they live in. Describe a MapReduce workflow that computes the number of tax payers per state. First explain the workflow and then provide pseudocode for the map and reduce functions of your workflow.

Solution

The workflow consists of a single map reduce job. The map function $m : (ssn, state) \mapsto (state, 1)$ and the reduce function $r : \underbrace{[(state, 1), \ldots, (state, 1)]}_{n} \mapsto (state, n)$

Question 1.2.2 (5 Points)

Explain how group-by aggregation can be implemented using the MapReduce programming model.

Solution

Consider an aggregation ${}_G\gamma_{f(a)}(R)$ where G is a list of group-by attributes, f is an aggregation function, and a is an attribute from the input relation R.

- We use one map-reduce phase for this.
- The map function takes inputs (k, t) where k is a key and t is a full tuple and outputs $(t.G, a)$.
- The reduce function $r : \underbrace{[(g, a_1), \ldots, (g, a_n)]}_{n} \mapsto (g, f(a_1, \ldots, a_n))$

Question 1.2.3 Fault Tolerance (4 Points)

Explain in a few sentences why load balancing is critical for distributed systems to scale.

Solution

Many distributed computation require all nodes involved in a computation to finish processing before the output is returned. In this case the slowest node will determine performance. With a large number of nodes, the effect of random events that delay the progress of a node are more pronounced, because even if the probability of a single node experiencing such an event is low, for a large number of nodes it is likely that at least one node will experience a delay and, thus, slow down the computation. A similar argument holds for imbalances in distributing the work across nodes. Load balancing refer to techniques that aim at distributing the load evenly across nodes to reduce the chance of stragglers (nodes that are much slower than the rest of the cluster). Based on the above argument this will improve scalability by reducing the effect random differences in work assigned to nodes have on overall system performance.

Question 1.2.4 Distributed file systems (5 Points)

- ■ HDFS automatically detects when a data node is down
- ■ Writing of files in HDFS is append-only
- ❑ HDFS does not rely on replication to achieve fault tolerance
- ❑ HDFS scales well to large number of files
- ■ In HDFS, clients communicate both with the name node as well as with data nodes

Question 1.2.5 MapReduce and Hadoop (4 Points)

- ■ The map function in MapReduce is applied to single key-value pairs from the input.
- ■ The map phase in MapReduce does not require any communication among workers
- ■ Hadoop MapReduce uses an external merge sort algorithm to sort the input of reducers on their keys
- ❑ A shuffle only requires network communication, but not disk I/O

Part 1.3 Provenance (Total: 30 Points)

For the following the queries over the schema shown below, compute the provenance according to the following provenance models for all their result tuples.

- Why-Provenance
- Minimal Why-Provenance
- Provenance Polynomials

Before presenting provenance, show the results for each query first and label the result tuples t_1, t_2, $\ldots$, t_n.

Consider the following database schema and instance:

location

lName	city	owner	sizeSf	
Windsor Castle	Windsor	Queen	40,000	l_1
Big Ben	London	Public	3,500	l_2
Stonehedge	Amesbury	Public	14,000	l_3

account

witness	suspect	crimeId	
Bob	Peter	1	a_1
Peter	Bob	1	a_2
Queen	Bob	2	a_3

crime

crimeId	lName	time	type	victim	
1	Big Ben	10:30	murder	Alice	c_1
2	Windsor Castle	11:00	theft	Queen	c_2

Question 1.3.1 (5 Points)

$$\pi_{city}(\sigma_{sizeSf>10{,}000}(location))$$

Solution

Result relation:

city	
Windsor	t_1
Amesbury	t_2

Why provenance:

city	
Windsor	$\{\{l_1\}\}$
Amesbury	$\{\{l_3\}\}$

Minimal Why provenance:

city	
Windsor	$\{\{l_1\}\}$
Amesbury	$\{\{l_3\}\}$

Provenance Polynomials:

city	
Windsor	l_1
Amesbury	l_3

Question 1.3.2 (8 Points)

$$q_1 \overset{def}{=} (location \bowtie account \bowtie crime)$$
$$q \overset{def}{=} \pi_{type,time,city}(q_1)$$

Solution

Result relation:

type	time	city	
murder	10:30	London	t_1
theft	11:00	Windsor	t_2

Why provenance:

type	time	city	
murder	10:30	London	$\{\{c_1, l_2, a_1\}, \{c_1, l_2, a_2\}\}$
theft	11:00	Windsor	$\{\{c_2, l_1, a_3\}\}$

Minimal Why provenance:

type	time	city	
murder	10:30	London	$\{\{c_1, l_2, a_1\}, \{c_1, l_2, a_2\}\}$
theft	11:00	Windsor	$\{\{c_2, l_1, a_3\}\}$

Provenance Polynomials:

type	time	city	
murder	10:30	London	$(c_1 \cdot l_2 \cdot a_1) + (c_1 \cdot l_2 \cdot a_2)$
theft	11:00	Windsor	$c_2 \cdot l_1 \cdot a_3$

Question 1.3.3 (8 Points)

$$\begin{aligned}
q_1 &\overset{def}{=} \rho_{s1 \leftarrow suspect, w1 \leftarrow witness}(account) \\
q_2 &\overset{def}{=} \rho_{s2 \leftarrow suspect, w2 \leftarrow witness}(account) \\
q &\overset{def}{=} \pi_{crimId, w1, w2}(\sigma_{w1 \neq w2}(crime \bowtie q_1 \bowtie q_2))
\end{aligned}$$

Solution

Result relation:

crimeId	w1	w2	
1	Bob	Peter	t_1
1	Peter	Bob	t_2

Why provenance:

crimeId	w1	w2	
1	Bob	Peter	$\{\{c_1, a_1, a_2\}\}$
1	Peter	Bob	$\{\{c_1, a_1, a_2\}\}$

Minimal Why provenance:

crimeId	w1	w2	
1	Bob	Peter	$\{\{c_1, a_1, a_2\}\}$
1	Peter	Bob	$\{\{c_1, a_1, a_2\}\}$

Provenance Polynomials:

crimeId	w1	w2	
1	Bob	Peter	$c_1 \cdot a_1 \cdot a_2$
1	Peter	Bob	$c_1 \cdot a_1 \cdot a_2$

Question 1.3.4 (9 Points)

$$q \overset{def}{=} \pi_{city}(\sigma_{type='murder'}(location \bowtie crime)) \cup \pi_{city}(\sigma_{victim='Queen'}(location \bowtie crime))$$

Solution

Result relation:

city	
London	t_1
Windsor	t_2

Why provenance:

city	
London	$\{\{c_1, l_2\}\}$
Windsor	$\{\{c_2, l_1\}\}$

Minimal Why provenance:

city	
London	$\{\{c_1, l_2\}\}$
Windsor	$\{\{c_2, l_1\}\}$

Provenance Polynomials:

city	
London	$c_1 \cdot l_2$
Windsor	$c_2 \cdot l_1$