Database systems, winter 2015-2016

Dry homework#1

Submission until November 23rd at 13:30, pairs only, in the course’s cell on the first floor.

Teaching assistant in charge: Hadar Frenkel.

For questions please contact hfrenkel@cs.technion.ac.il with the subject HW1.

Students’ details:

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please return to cell:________

Final grade:_______________
Question 1: ERD, 40 points

Given the following requirements for a libraries database:

- Every library has an opening hours and a unique name.
- The library can be either a mobile library or a stationary library.
- For stationary libraries we need to remember the address of the library.
- Every book has a title, an author, a copy number and an ISBN number.
- Every book belongs to one library only.
- There may be several copies of the same book at one library. There cannot be two books with the same ISBN and copy number within the same library. There may be two books with the same ISBN and copy number in different libraries.
- Every reader has a name and a unique ID.
- In addition, there are readers that are members of the "Book Club" of a certain library. For these readers we need to remember also a unique club member number.
- There are books that are only for club members, and regular readers cannot borrow them.
- We want to keep information about current borrowings (no need to keep information about the past). At any given moment a specific copy of a book can be borrowed by at most one reader.
a. 18 point - Draw an ERD that describes the database in the best way and follows the given requirements. Make sure that you mark the keys in the diagram.
b. 9 points - Translate the ERD from section a into tables (for entities, relationships...). Remember to mention keys.

```
book(vol, ISBN, lib_name, author, title)
belongs(ISBN, vol, Lib_name)
library(lib_name, hours)
stationary(lib_name, address)
mobile(lib_name)
loan(ISBN, vol, ID)
special_loan(ID, ISBN, vol, lib_name, club_num)
reader(id, name)
member(ID, lib_name, club_num)
```
c. 6 points – according to the diagram you drew in section a, can a book stand on its own (not as a book in the library)? If so, explain how to change the diagram so that this cannot happen (you can explain in words, but we prefer it if you drew). If not, explain why.

NO, since it is a weak entity, under “library”.

d. 7 points - Explain how to modify the diagram in section a, so that we can keep information about borrowings that happened in the past - that is, for anyone who borrowed a book, we want to know the dates of the borrowing.

We will add to “loan”, “special loan” relations a multivalued attribute for dates of past loans.
Question 2: RA, 40 points
In this question you need to use the tables derived from the translation of the ERD of question 1.

a. 10 points – write an RA query that finds all readers so that all their borrowed books are from the same library.

\[
\text{readerInLib} = \pi_{\text{libNAME,readerID}}((\text{Belongs} \bowtie \text{loans}) \\
\quad \cup (\text{Belongs} \bowtie \text{specialLoans})) \\
\text{moreThanOneLib} = \pi_{\text{readerID}}(\sigma_{\text{libName1} \neq \text{libName2}} \sigma_{\text{readerID1}=\text{readerID2}}(\text{readerInLib} \\
\quad \times \text{readerInLib})) \\
\text{ans} = \pi_{\text{readerID,readers}} \setminus \text{moreThanOneLib}
\]

b. 10 points – write an RA query that finds all books in the “book club” that were read by all of the readers in system, and were written by “Meir Shalev”.

\[
\pi_{\text{bookName}}((\sigma_{\text{author}="Meir Shalev" \text{Books}} \bowtie \text{specialLoan}) \\
\quad \div \pi_{\text{readerID,readers}})
\]
c. 7 points - Prove or refute: for every relation schemas R and S and a group X of attributes it holds that: let \( X_1 = R \cap X, X_2 = S \cap X \) then
\[
\pi_X(R \bowtie S) = \pi_{X_1}R \bowtie \pi_{X_2}S
\]

This does not hold:
\[
R=(ABC), S=(CDE), X=AE
\]
\[
\begin{align*}
r &= \{ (a_1, b_1, c_1), (a_2, b_2, c_2) \} \\
s &= \{ (c_1, d_1, e_1), (c_2, d_2, e_2) \}
\end{align*}
\]
\[
\begin{align*}
r \bowtie s &= \{ (a_1, b_1, c_1, d_1, e_1), (a_2, b_2, c_2, d_2, e_2) \} \\
\pi_X(r \bowtie s) &= \{ (a_1, e_1), (a_2, e_2) \} \\
\pi_{X_1}r &= \{ a_1, a_2 \}, \pi_{X_2}s = \{ e_1, e_2 \}
\end{align*}
\]
\[
\begin{align*}
\pi_{X_1}r \bowtie \pi_{X_2}s &= \{ (a_1, e_1), (a_1, e_2), (a_2, e_1), (a_2, e_2) \}
\end{align*}
\]

d. In the lecture you have learned 6 primitive operators: projection, selection, renaming, union, difference and Cartesian product. Prove formally the following independencies:
1. 5 points – the Cartesian product operator, \( \times \), is independent of all other operators.

We will show by induction that \( \times \) can add attributes to the schema while other operators can’t.

fix schema \( S=R(A), S(A) \), and instance \( I=\{ R(0), S(1) \} \)

\( P=\#\text{attributes}<2 \)

Base: \( R, S \) both have \( \#\text{att}<2 \).

Induction: assume \( \varphi \ (I) \) have \( \#\text{att}<2 \), then:
- \( \sigma_c(\varphi \ (I)) \) have \( \#\text{att}<2 \) since \( \sigma \) chooses rows but doesn’t concern the attributes.
- \( \pi_c(\varphi \ (I)) \) can only lessen the number of attributes, therefore have \( \#\text{att}<2 \)
- \( \rho(\varphi \ (I)) \) have \( \#\text{att}<2 \) since \( \rho \) doesn’t concern the number of attributes.
- Union and difference: act on relations with the same schema, change only the number of rows.

Cartesian product: \( R \times S = \{ (0,1) \} \) a table with 2 attributes, and therefore \( P \) does not hold.
2. 8 points – the difference operator, \( \setminus \), is independent of all other operators. 
(hint: prove that without difference, RA is monotonic).

Fix schema \( S = R(A), S(A) \), and instance \( I = \{ R(0), R(1), S(1) \} \)

\( P = \text{monotony: adding row to the base relation will add row to the result.} \)

Base: adding rows to \( R \) and \( S \) will obviously add rows and not reduce them.

Induction:

- \( \sigma \): if we add rows to \( \varphi \ (R) \) and get \( \varphi' \ (R) \), then \( \sigma' \varphi \ (R) \) will be at least the size of \( \sigma \varphi \ (R) \), since there are more rows that can be chosen to the result.

- \( \pi \): affects only the number of attributes. Adding rows to \( \varphi \ (R) \) will obviously add rows to the projection result.

- \( \rho \varphi \ (R) \) doesn’t change the number of rows, hence, if we got \( \varphi' \ (R) \) by adding row to \( \varphi \ (R) \) then \( \rho \varphi' \ (R) \) will have more rows as well.

- Union: if we got \( \varphi'_1 \ (R) \) by adding rows to \( \varphi_1 \ (R) \) and \( \varphi'_2 \ (S) \) by adding rows to \( \varphi_2 \ (S) \) than \( \varphi' \ (R) \cup \varphi' \ (S) \) will have no less rows than \( \varphi \ (R) \cup \varphi \ (S) \), since the since the number of rows in the union is the sum of the distinct rows of the two relations.

- Cartesian product: the same as union, since the number of rows is the product of the number of rows of the two relation.

Difference: \( R \setminus S = \{(0)\} \)

If we add to \( R \): \( R(2) \) and get \( R' \)

And we add to \( S \): \( S(0), S(2) \) and get \( S' \)

Then \( R' \setminus S' = \emptyset \)

Hence, adding rows to the original relations made the result smaller, not bigger, therefore difference is not monotonic.
Question 3: SQL, 20 points
Build a legal SQL script (a series of SQL commands that run in that order and not fail) that executes the following actions (in that order) – you might need more than one SQL command in order to implement one action:

- Create two tables:
  1. A table with residents that have a unique ID (a positive number), and at least two more fields as you choose.
  2. A table that keeps the history of places the residents lived in, with the following conditions:
     - Each resident can live in each place at most once.
     - You have to keep the date the resident moved to that address.

- Add history for two residents.
- Delete one of the residents from the system

```
CREATE TABLE Citizens
(Id INTEGER, Name varchar, work varchar, PRIMARY KEY (Id), CHECK (Id > 0))

CREATE TABLE History
(Id INTEGER, Address TEXT NOT NULL, Date DATE NOT NULL, UNIQUE (Id, Address),
FOREIGN KEY (Id) REFERENCES Citizens(Id) ON DELETE CASCADE)

INSERT INTO Citizens VALUES (123, 'Ayelet Shaked', 'minister')
INSERT INTO Citizens VALUES (101, 'BOB', 'builder')
INSERT INTO History VALUES (123, 'BBC', '20150317')
INSERT INTO History VALUES (123, 'Hamagenem 12, Haifa Israel', '19980101')
DELETE FROM Citizens WHERE Id = 123
```