Fall 2022 B461 Assignment 4
Joins, Semi-joins and Relational Algebra

Srinivas Kini, Keerthana Sugasi, Muazzam Siddiqui

Released: October 9th 2022
Due: October 27th 2022

1 Introduction

The goals for this assignment are to

1. Formulate queries in Relational Algebra (RA) SQL.
2. Formulate RA expressions from statements.

To turn in your assignment, you will need to upload the following files:

- `assignment4.sql`
- `assignment4.txt`
- `assignment4.pdf`

The `assignment4.sql` contains the necessary SQL statements that solve the problems in this assignment. The `assignment4.sql` file must be such that the AI's can run it in their PostgreSQL environment.

The `assignment4.txt` file contains the results of running your queries.

The `assignment4.pdf` file contains the results of the RA expressions in standard notation.

Grading Rubric (100 pts total)

1. `10 pts` if the query returns expected results.
2. `0 - 9 pts` for incorrect results, the deduction of points will be gauged on how logically sound the query is.
3. Out of 10 questions, 3 will be randomly selected and full credit be will awarded for attempting them. **Note: the attempt should be relevant to the question and non-trivial.**

For the problems in this assignment we will use the following database schema:[1]

```
Westerosi(wid, wname, wlocation)
House(hname, kingdom)
Skill(skill)
OfHouse(wid, hname, wages)
HouseAllyRegion(hname, region)
WesterosiSkill(wid, skill)
Predecessor(succid, predid)
Knows(wid1, wid2)
```

In this database[2] we maintain a set of Westerosis[3] (`Westerosi`), a set of Houses (`House`), and a set of skills (`Skill`). The `wname` attribute in `Westerosi` is the name of the resident of Westeros.

The `wlocation` attribute in `Westerosi` specifies the area in which the person is currently stationed. The `hname` attribute in `House` is the name of a House in Westeros.

The `kingdom` attribute in `House` is the name of the location wherein the lord of the house resides. The `skill` attribute in `Skill` is the name of a skill possessed by Westerosis.

A Westerosi can be of at most one House. This information is maintained in the `OfHouse` relation. (We permit that a Westerosi does not belong to any House.) The `wages` attribute in `OfHouse` specifies the wages made by the Westerosi.

The `region` attribute in `HouseAllyRegion` indicates a region in which the house has allies. (Houses may have allies in multiple regions.)

A Westerosi can have multiple skills. This information is maintained in the `WesterosiSkill` relation. A skill can be the skill of multiple Westerosis. (A Westerosi may not have any skills, and a skill may have no Westerosis with that skill.)

A pair (s, p) in `Predecessor` indicates that a Westerosi (successor) s has a Westerosi p as one of his or her predecessors. We permit that a successor has multiple predecessors and that a predecessor may be succeeded by multiple successors. (It is possible that a Westerosi has no predecessor and that a Westerosi is not a predecessor.) We further require that a Westerosi and his or her predecessors must belong to the same House.

The relation `Knows` maintains a set of pairs (w_1, w_2) where w_1 and w_2 are wids of Westerosis. The pair (w_1, w_2) indicates that the person with wid w_1

[1] The primary key, which may consist of one or more attributes, of each of these relations is underlined.

[2] The values of the database are inspired by a popular series - Game of Thrones just to make the course a little fun. We in no way bear responsibility for any spoilers or faults in the storyline/theories based on these values. So kindly humor us and have just as fun with making the queries as we do in asking for them!

[3] Residents of Westeros

knows the person with wid w_2. We do not assume that the relation `Knows` is symmetric: it is possible that (w_1, w_2) is in the relation but that (w_2, w_1) is not.

The domain for the attributes `wid`, `wages`, `succid`, and `predid` is `integer`. The domain for all other attributes is `text`.

We assume the following foreign key constraints:

- `wid` is a foreign key in `OfHouse` referencing the primary key `wid` in `Westerosi`;
- `hname` is a foreign key in `OfHouse` referencing the primary key `hname` in `House`;
- `hname` is a foreign key in `HouseAllyRegion` referencing the primary key `hname` in `House`;
- `wid` is a foreign key in `WesterosiSkill` referencing the primary key `wid` in `Westerosi`;
- `skill` is a foreign key in `WesterosiSkill` referencing the primary key `skill` in `Skill`;
- `succid` is a foreign key in `Predecessor` referencing the primary key `wid` in `Westerosi`; and
- `predid` is a foreign key in `Predecessor` referencing the primary key `wid` in `Westerosi`;
- `wid1` is a foreign key in `Knows` referencing the primary key `wid` in `Westerosi`; and
- `wid2` is a foreign key in `Knows` referencing the primary key `wid` in `Westerosi`

The file `a4data.sql` contains the data supplied for this assignment.

2 Instructions

1. Each query must be formulated in Relational Algebra (RA) SQL (explained in detail in the points below).
2. You must **NOT** use the set predicates `IN, NOT IN, EXISTS, NOT EXISTS, SOME & ALL`. Furthermore, you also cannot use window functions like `RANK, CUBE` etc. in any query.
3. You must **NOT** use aggregate functions.
4. You may make use of set operators like `UNION, INTERSECT & EXCEPT`.
5. You must make use of `JOINS` in each query.
6. You may make use of functions, views and parameterized views as long as the queries they embed adhere to the points mentioned above.
7. The RA expressions should be constructed properly, with the use of appropriate symbols and operators ($\sigma, \pi, \bowtie_C, \bowtie, \ltimes, \cup, \cap, and-$)
.

8. Use the following table of (indexed) letters to denote the relation names in RA expressions.

`Westerosi`	$w, w_1, w_2...$
`House`	$h, h_1, h_2...$
`Skill`	$s, s_1, s_2...$
`OfHouse`	$oh, oh_1, oh_2...$
`HouseAllyRegion`	$hr, hr_1, hr_2...$
`WesterosiSkill`	$ws, ws_1, ws_2...$
`Predecessor`	$p, p_1, p_2...$
`Knows`	$k, k_1, k_2...$

RA SQL Example: *Return the wid and wlocation of each Westerosi that has at least 1 skill, or is known by at least 1 Westerosi.*

```
select w.wid, w.wlocation from Westerosi w
join WesterosiSkill ws on (ws.wid = w.wid)
union
select w.wid, wlocation from Westerosi w
join Knows k on (w.wid = k.wid2)
```

Notice that the query above obeys points 2 through 6 mentioned in the instructions. The RA Expression of this query is given by:

E = $\pi_{W.wid,W.location}(W \bowtie_{W.wid=wS.wid} wS)$ (Westerosis with 1 or more skills)
F = $\pi_{W_1.wid,W_1.location}(W_1 \bowtie_{W_1.wid=K.wid2} K)$ (Westerosis known by at least 1 other Westerosi)
RA Expression : $E \cup F$ (The union of E and F)

3 Queries in RA

Formulate each query below in RA SQL, and write them in the `assignment4.sql` file. Write the outputs to the queries in `assignment4.txt`.

1. Formulate a query in RA SQL that returns each `hname` such that no Westerosi belonging to that house has the 'Archery' skill.

2. Formulate a query in RA SQL that returns the `wid` and `region` of each Westerosi that:

   - Knows at least 2 people OR,
   - Has no successor

3. Formulate a query in RA SQL that returns each skill that is a skill of some `predid`, such that each `succid` associated with that `predid` does not have any of those skills.

4. Consider the following query in Pure SQL:

   ```
   select w.wid, exists (select 1
                         from Predecessor P1, Predecessor P2
                         where P1.predid = w.wid and P2.predid = w.wid and
                         P1.succid <> P2.succid)
   from Westerosi w;
   ```

   This query returns a pair (w, t) if w is the wid of a Westerosi who has at least two predecessors and returns the pair (w, f) otherwise [4].

   Formulate the query above in RA SQL.

5. Formulate a query in RA SQL that finds the `wid` and `wname` of each Westerosi who belongs to a house allied in Winterfell but does not know any Westerosi that lives in BlackwaterBay.

[4] t represents the boolean value `true` and f represents the boolean value `false`. Note that you can represent these values as constants in RA SQL.

4 RA Expressions

It is also possible to write constraints using RA expressions. Let E, F, and G denote RA expressions. Then we can write RA expression comparisons that express constraints:

$E \neq \emptyset$	which is true if E evaluates to an non-empty relation
$E = \emptyset$	which is true if E evaluates to the empty relation
$F \subseteq G$	which is true if F evaluates to a relation that is a subset of the relation obtained from G
$F \not\subseteq G$	which is true if F evaluates to a relation that is not a subset of the relation obtained from G

Here are some examples of writing constraints in this manner.
Example for RA constraint: "*Some Westerosi belongs to Lannister.*" This constraint can be written as follows:

$$\pi_{wid}(\sigma_{hname=\textbf{Lannister}}(oh)) \neq \emptyset$$

The above RA expression computes set of all the wid of westerosi who belong to house Lannister. If it is not a emptyset then there are people who belong to Lannister.

Incidentally, the constraint "*No one belongs to Lannister*" can be written as follows

$$\pi_{wid}(\sigma_{hname=\textbf{Lannister}}(oh)) = \emptyset$$

Express the queries and constraints given below in standard RA notation and submit the responses in `assignment4.pdf`.

6. Write an RA expression for the following constraint: *The attribute* `wid` *is the primary key of the relation* `Westerosi`.

7. Write an RA expression for the following query: *Return the* `wid` *of each Westerosi such that at least 1 Westerosi known by him/her knows all his/her successors.*

8. Write an RA expression for the following constraint: *Some* `skill` *is not a skill of all Westerosis*.

9. Write an RA expression for the query in Question 4 of Section 3. The RA Expression should match the query.

10. Write an RA expression for the constraint: *Some person has fewer than 2 skills.*