# B561 Advanced Database Concepts
# Assignment 2
# Fall 2022

Dirk Van Gucht

This assignment relies on the lectures

- SQL Part 1 and SQL Part 2 (Pure SQL);
- Tuple Relational Calculus;
- Views;
- Relational Algebra (RA); and
- Joins and semijoins.

To turn in your assignment, you will need to upload to Canvas 3 files:

1. A file with name `assignment2.sql` which contains the necessary SQL statements that solve the problems in this assignment. The `assignment2.sql` file must be so that the AI's can run it in their PostgreSQL environment. You should use the `Assignment2Script.sql` file to construct the `assignment2.sql` file. (Note that the data to be used for this assignment is included in this file.)
2. A file with name `assignment2.txt` file that contains the results of running the SQL code in the `assignment2.sql` code.
3. A file with name `assignment2.pdf` that contains the solutions to the problems that require it. You are strongly advised to use Latex to construct this file.

The problems that need to be included in the `assignment2.sql` are marked with a blue bullet •. The problems that need to be included in the `assignment2.pdf` are marked with a green bullet •. Problems that are marked with a red bullet • are practice problems and they should not be included in the 3 files you need to submit.

### Database schema and instances

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

```
Student(sid, sname, birthYear)
Book(bno, title, price)
Buys(sid, bno)
Major(major)
hasMajor(sid, major)
Cites(bno1, bno2)
```

In this database we maintain a set of students (`Student`), a set of books (`Book`), and a set of majors (`Major`). The `sname` attribute in `Student` is the name of the student. The `birthYear` attribute in `Student` specifies the birth year of the student. The `bno` attribute in `Book` is the book number of the book.[2] The `title` attribute in `Book` is the title of the book. The `price` attribute in `Book` is the price for the book. The `major` attribute in `Major` is the name of a major.

A student can buy books. This information is maintained in the `Buys` relation. A triple $(s, b)$ indicates that the student with sid $s$ buys the book with bno $b$. We permit that a student buys multiple books and that a book is bought by multiple students. It is possible that a student buys no books and that a book is bought by no students.

A student can have multiple majors. This information is maintained in the `hasMajor` relation. A student can have multiple majors and a major can have multiple students. It is possible that a student has no major and that a major has no students.

A book can cite other books. This information is maintained in the `Cites` relation. A pair $(b_1, b_2)$ in `Cites` indicates that the book with bno $b_1$ cites the book with bno $b_2$. We permit that a book cites multiple books and that a book is cited by multiple books. It is possible that a book cites no other books and that a book is not cited by any book. The domain for the attributes `sid`, `bno`, `bno1`, `bno2`, `birthYear`, and `price` is `integer`. The domain for all other attributes is `text`.

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

[2] You should think of `bno` as the `ISBN` number of a book. Notice that different physical books may have the same ISBN number.

The primary keys in the relations are the respective underlined attributes.

We assume the following foreign key constraints:

- `sid` is a foreign key in `Buys` referencing the primary key `sid` in `Student`;
- `bno` is a foreign key in `Buys` referencing the primary key `bno` in `Book`;
- `sid` is a foreign key in `hasMajor` referencing the primary key `sid` in `Student`;
- `major` is a foreign key in `hasMajor` referencing the primary key `major` in `Major`;
- `bno1` is a foreign key in `Cites` referencing the primary key `bno` in `Book`; and
- `bno2` is a foreign key in `Cites` referencing the primary key `bno` in `Book`.

We assume that a book can not recursively cite itself. In other words, the `Cites` relation is an acylic directed graph. I.e., there are no cycles in the `Cites graph`.

### Pure SQL and RA SQL

In this assignemt, we distinguish between Pure SQL[3] and RA SQL. Below we list the **only** features that are allowed in Pure SQL and in RA SQL.

In particular notice that

- '`join`', '`natural join`', and '`cross join`' are **not** allowed in Pure SQL.
- The subquery expressions '`some`', '`all`', ['`not`'] "`exists`", ['`not`] '`exists`' are **not** allowed in RA SQL.

**The only features allowed in Pure SQL**

`select ... from ... where`
`with ...`
`union, intersect, except` operations
'`some`' and '`all`' subquery expressions
'`exists`' and '`not exists`' subquery expressions
'`in`' and `not in` subquery expressions
'`view`'s that can only use the above Pure SQL features

**The only features allowed in RA SQL**

`select ... from ... where`
`with ...`
`union, intersect, except` operations
`join ... on ...`, `natural join`, and `cross join` operations
'`view`'s that can only use the above RA SQL features
commas in the `from` clause are **not** allowed

[3]Pure SQL is the fragment of SQL that was covered in the lectures up to but not including the lectures about relational algebra.

# 1 Translating SQL queries with query predicates and subquery expressions into safe TRC

We have learned that safe TRC is at the core of the design of SQL. For the problems in the section, you are asked to translate SQL queries into equivalent safe TRC queries. To illustrate what you need to do, consider the following example:

**Example 1** *Consider the query* 'Find each $(s, b)$ pair where $s$ is the sid of a student who bought book $b$ and such that each other book, with price above \$30, that was bought by $s$ is a book cited by $b$.'

*This query can be expressed in SQL as*

```
select t.sid, t.bno
from   Buys t
where  true = all (select (t.bno, b.bno) in (select c.bno1, c.bno2
                                             from   Cites c)
                   from   Book b
                   where  b.bno != t.bno and
                          b.price >= 30 and
                          (t.sid, b.bno) in (select t1.sid, t1.bno
                                             from   Buys t1));
```

*Starting from this SQL query, we obtain the safe TRC query*

$$\{(t.sid, t.bno) \mid Buys(t) \wedge \\ \forall b\,((Book(b) \wedge b.bno \neq t.bno \wedge b.price \geq 35 \wedge Buys(t.sid, b.bno)) \rightarrow Cites(t.bno, b.bno))\}$$

*Using logical equivalences, this query can also be expressed as the SQL query*

```
select t.sid, t.bno
from   Buys t
where  not exists (select 1
                   from   Book b
                   where  b.bno != t.bno and
                          b.price >= 30 and
                          (t.sid, b.bno) in (select t1.sid, t1.bno
                                             from   Buys t1) and
                          (t.bno, b.bno) not in (select c.bno1, c.bno2
                                                 from   Cites c));
```

*Starting from this SQL query, we obtain the safe TRC query*

$$\{(t.sid, t.bno) \mid Buys(t) \wedge \\ \neg\exists b \in Book\,(b.bno \neq t.bno \wedge b.price \geq 35 \wedge Buys(t.sid, b.bno) \wedge \neg Cites(t.bno, b.bno))\}$$

We now turn to the problems for this section.

1. • Consider the query '*Find the bno and title of each book that was bought by exactly one student.*' This query can be expressed as the SQL query

```
select b.bno, b.title
from   Book b
where  true = some  (select (s.sid, b.bno) in (select t.sid, t.bno
                                               from   Buys t)

                     from   Student s) and
       true = all (select s1 = s2
                   from   Student s1, Student s2
                   where  (s1.sid, b.bno) in (select t.sid, t.bno from Buys t) and
                          (s2.sid, b.bno) in (select t.sid, t.bno from Buys t));
```

   Starting from this SQL query, express the query in safe TRC.

2. • Consider the query '*Find each pair* $(m, b)$ *where* $m$ *is a major and* $b$ *is the bno of a book bought by a student who has major* $m$ *and such that the price of* $b$ *is the lowest among the set of books bought by students with major* $m$.'

```
select m.major, b.bno
from   Major m, Book b
where  b.bno in (select t.bno
                 from   Buys t
                 where  t.sid in (select hm.sid
                                  from   hasMajor hm
                                  where  hm.major = m.major)) and
       not exists (select 1
                   from   Buys t, Book b1
                   where  t.bno = b1.bno and
                          true = some (select (t.sid,m.major) in (select hm.sid, hm.major
                                                                  from   hasMajor hm)) and
                          b1.price < b.price);
```

   Starting from this SQL query, express the query in safe TRC.

## 2 Expressing queries in (Extended) Safe TRC and Pure SQL with and without subquery expressions

An important consideration in expressing queries is knowing how they can be written in different, but equivalent, ways. In fact, this is an aspect of programming in general and, as can expected, is also true for database programming languages such as SQL. A learning outcome of this course is to acquire skills for writing queries in different ways. One of the main motivations for this is to learn that different formulations of the same query can dramatically impact performance, sometimes by orders of magnitude.

For the problems in this section, you will need to express queries in Pure SQL with and without subquery expressions. You can use the SQL operators `INTERSECT`, `UNION`, and `EXCEPT`, unless otherwise specified. You are allowed to use views including temporary and user-defined views.

To illustrate what you need to do for the problems in this section, consider the following example.

**Example 2** *Consider the query* 'Find the bno and title of each book that cites a book and that was bought by a student who majors in CS or in Math.'

**(a)** *Express this query in Safe SQL (i.e, with quantifiers '∃' or '∀').*

$$\{(b.bno, b.title) \mid Book(b) \wedge \exists c(Cites(c) \wedge c.bno1 = b.bno) \wedge \\ \exists t(Buys(t) \wedge t.bno = b.bno \wedge \\ \exists hm(hasMajor(hm) \wedge hm.sid = t.sid \wedge (hm.major = \mathbf{CS} \vee hm.major = \mathbf{Math})))\}$$

**(b)** *Express this query in Extended Safe SQL (i.e, with subquery expressions).*

$$\{(b.bno, b.title) \mid Book(b) \wedge \texttt{some}(\{c.bno1 = b.bno \mid Cites(c)\}) \wedge \\ \texttt{some}(\{t.bno = b.bno \mid Buys(t) \wedge \\ \texttt{some}(\{hm.sid = t.sid \wedge (hm.major = \mathbf{CS} \vee hm.major = \mathbf{Math}) \mid hasMajor(hm)\})\})\}$$

*Or, alternatively*

$$\{(b.bno, b.title) \mid Book(b) \wedge \texttt{some}(\{c.bno1 = b.bno \mid Cites(c)\}) \wedge \\ \texttt{some}(\{t.bno = b.bno \wedge \texttt{some}(\{hm.sid = t.sid \wedge \\ (hm.major = \mathbf{CS} \vee hm.major = \mathbf{Math}) \mid hasMajor(hm)\}) \mid Buys(t)\})\}$$

**(c)** *Express this query in Pure SQL by only using the '*`true = some`*' or '*`true = all`*' subquery expressions.*
*A possible solution is*

```
select b.bno, b.title
from   Book b
where  true = some (select c.bno1 = b.bno
                    from   Cites c) and
       true = some (select t.bno = b.bno and
                    from   Buys t
                    where  true = some (select hm.sid = t.sid and
                                               (hm.major = 'CS' or hm.major = 'Math')
                                        from   hasMajor hm);
```

*Or, alternatively*

```
select b.bno, b.title
from   Book b
where  true = some (select c.bno1 = b.bno
                    from   Cites c) and
       true = some (select t.bno = b.bno and
                         true = some (select hm.sid = t.sid and
                                             (hm.major = 'CS' or hm.major = 'Math')
                                      from   hasMajor hm)
                    from   Buys t);
```

**(d)** *Express this query in Pure SQL by only using the '*`exists`*' or '*`not exists`*' subquery expressions.*
*A possible solution is*

```
select b.bno, b.title
from   Book b
where  exists (select 1
               from   Cites c
               where  c.bno1 = b.bno) and
       exists (select 1
               from   Buys t
               where  t.bno = b.bno and
                      exists (select 1
                              from   hasMajor hm
                              where  hm.sid = t.sid and
                                     (hm.major = 'CS' or hm.major = 'Math')));
```

**(e)** *Express this query in Pure SQL by only using the '*`in`*' or '*`not in`*' subquery expressions. You can not use the set operations* `INTERSECT`, `UNION`, *and* `EXCEPT`.
*A possible solution is*

```
select b.bno, b.title
from   Book b
where  b.bno in (select c.bno1
                 from   Cites c) and
       b.bno in (select t.bno
                 from   Buys t
                 where  t.sid in (select hm.sid
                                  from   hasMajor hm
                                  where  hm.major = 'CS' or hm.major = 'Math'));
```

**(f)** *Express this query in Pure SQL without using subquery expressions.*

*A possible solution is*

```
select b.bno, b.title
from   Book b, Cites c
where  b.bno = c.bno1
intersect
select b.bno, b.title
from   Book b, Buys t, hasMajor hm
where  b.bno = t.bno and
       t.sid = hm.sid and
      (hm.major = 'CS' or hm.major = 'Math');
```

We now turn to the problems for this section.

3. Consider the query '*Find the bno and title of each book that is bought by a student who is (strictly) younger than each student who majors in Chemistry and who also bought that book.*'

   (a) • Express this query in Safe TRC (i.e, with quantifiers '$\exists$' or '$\forall$').

   (b) • Express this query in Extended Safe TRC (i.e, with subquery expressions).

   (c) • Express this query in Pure SQL by only using the '`true = some`' or '`true = all`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (d) • Express this query in Pure SQL by only using the '`exists`' or '`not exists`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (e) • Express this query in Pure SQL by only using the '`in`' or '`not in`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (f) • Express this query in Pure SQL without using subquery expressions. Now you can use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

4. Consider the query '*Find each student-book pair* $(s, b)$ *where* $s$ *is the sid of a student who majors in CS and who bought each book that costs no more than book* $b$.'

   (a) • Express this query in Extended Safe TRC (i.e, with subquery expressions).

   (b) • Express this query in Pure SQL by only using the '`true = some`' or '`true = all`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (c) • Express this query in Pure SQL by only using the '`exists`' or '`not exists`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (d) • Express this query in Pure SQL by only using the '`in`' or '`not in`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

(e) • Express this query in Pure SQL without using subquery expressions. Now you can use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

5. Consider the query '*Find the sid and name of each student who bought all-but-one book that cost strictly more than $30.*'

   (a) • Express this query in Safe TRC (i.e, with quantifiers '$\exists$' or '$\forall$').

   (b) • Express this query in Pure SQL by only using the '`true = some`' or '`true = all`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (c) • Express this query in Pure SQL by only using the '`exists`' or '`not exists`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (d) • Express this query in Pure SQL by only using the '`in`' or '`not in`' subquery expressions. You can not use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

   (e) • Express this query in Pure SQL without using subquery expressions. Now you can use the set operations `INTERSECT`, `UNION`, and `EXCEPT`.

Repeat the sub problems of the type 'a' through 'f' for the following queries

6. • Consider the query '*Find the sid and sname of each student who majors in CS and who is (strictly) younger than some other student who also majors in CS.*"

7. • Consider the query '*Find each $(s, b)$ pair where $s$ is the sid of a student who bought book $b$ and such that each other book bought by $s$ is a book cited by $b$.*"

8. • Consider the query '*Find each major that is not a major of any person who bought a book with title 'Databases' or 'Philosophy'.*"

9. • Consider the query *'Find the bno and title of each book that is bought by a student who majors in CS and who is, among all students who major in CS, the next-to-oldest.*'

# 3 Expressing queries in Relational Algebra and RA SQL

Reconsider the queries in Section 1 and Section 2. The objective is to express these queries in Relational Algebra in standard notation and in RA SQL.

There are some restrictions:

- You can only use `WHERE` clauses that use conditions involving constants. For example conditions of the form '$t.A\,\theta\,\mathbf{a}$', i.e., these involving a constant '**a**' are permitted, but conditions of the form '$t.A\,\theta\,s.B$', i.e., join conditions, are not allowed. These should be absorbed in `JOIN` operations in the `FROM` clause.
- You can not use commas in any `FROM` clause. Rather, you should use `JOIN` operations.

You can use the following letters, or indexed letters, to denote relation names in RA expressions:

| | |
|---|---|
| $S$, $S_1$, $S_2, \cdots$ | `Student` |
| $B$, $B_1$, $B_2, \cdots$ | `Book` |
| $T$, $T_1$, $T_2, \cdots$ | `Buys` |
| $M$, $M_1$, $M_2, \cdots$ | `Major` |
| $hM$, $hM_1$, $hM_2, \cdots$ | `hasMajor` |
| $C$, $C_1$, $C_2, \cdots$ | `Cites` |

To illustrate what you need to do, reconsider the query in Example 2 in Section 2.

**Example 3** *Consider the query* 'Find the bno and title of each book that cites a book and that was bought by a student who majors in 'CS' or in 'Math'.'

**(a)** *Express this query in Relational Algebra in standard notation.*

*A possible solution is*

$$\pi_{bno,title}(Book \bowtie_{bno=bno_1} Cites) \cap \pi_{bno,title}(\sigma_{major=\mathbf{CS} \vee major=\mathbf{Math}}(Book \bowtie Buys \bowtie hasMajor))$$

*If we use the letters in the above box, this expression becomes more succinct:*

$$\pi_{bno,title}(B \bowtie_{bno=bno_1} C) \cap \pi_{bno,title}(\sigma_{major=\mathbf{CS} \vee major=\mathbf{Math}}(B \bowtie T \bowtie hM))$$

*You are also allowed to introduce letters that denote expressions. For example, let $E$ and $F$ denote the expression*

$$\pi_{bno,title}(B \bowtie_{bno=bno_1} C)$$

*and*

$$\pi_{bno,title}(\sigma_{major=\mathbf{CS} \vee major=\mathbf{Math}}(B \bowtie T \bowtie hM)),$$

*respectively. Then we can write the solution as follows:*

$$E \cap F.$$

**(b)** *Express this query in RA SQL.*

*A possible solution is the following. Note that the* `WHERE` *clause only use conditions involving constants.*

```
select bno, title
from   Book
       JOIN Cites ON bno = bno1
intersect
select bno, title
from   Book
       NATURAL JOIN Buys
       NATURAL JOIN hasMajor
where  major = 'CS' or major = 'Math';
```

*Another possible solution is the following. In this solution, we have associated the condition involving just constants, i.e., the condition* '`major = 'CS' or major = 'Math'`'*, with the* `hasMajor` *relation.*

```
select bno, title
from   Book
       JOIN Cites ON bno = bno1
intersect
select bno, title
from   Book
       NATURAL JOIN Buys
       NATURAL JOIN (select sid
                     from   hasMajor
                     where  major = 'CS' or major = 'Math') s;
```

*We conclude with a solution that using a temporary view* '`CS_or_Math_Student`'*:*

```
with   CS_or_Math_Student as
       (select sid
        from   hasMajor
        where  major = 'CS' or major = 'Math')
select bno, title
from   Book
       JOIN Cites ON bno = bno1
intersect
select bno, title
from   Book
       NATURAL JOIN Buys
       NATURAL JOIN CS_or_Math_student;
```

We now turn to the problems in this section.

10. Reconsider the query in Problem 1 '*Find the bno and title of each book that was bought by exactly one student.*'

    (a) • Express this query in Relational Algebra in standard notation.

    (b) • Express this query in RA SQL.

11. Reconsider the query in Problem 2 '*Find each pair* $(m, b)$ *where* $m$ *is a major and* $b$ *is the bno of a book bought by a student who has major* $m$ *and such that the price of* $b$ *is the lowest among the set of books bought by students with major* $m$.'

    (a) • Express this query in Relational Algebra in standard notation.

    (b) • Express this query in RA SQL.

12. Reconsider the query in Problem 3 '*Find the bno and title of each book that is bought by a student who is (strictly) younger than each student who majors in Chemistry and who also bought that book.*'

    (a) • Express this query in Relational Algebra in standard notation.

    (b) • Express this query in RA SQL.

13. Reconsider the query in Problem 4 '*Find each student-book pair* $(s, b)$ *where* $s$ *is the sid of a student who majors in CS and who bought each book that costs no more than book* $b$.'

    (a) • Express this query in Relational Algebra in standard notation.

    (b) • Express this query in RA SQL.

14. Reconsider the query in Problem 5 '*Find the sid and name of each student who bought all-but-one book that cost strictly more than \$30.*'

    (a) • Express this query in Relational Algebra in standard notation.

    (b) • Express this query in RA SQL.

The remaining problems in this section are practice problems (indicated with a red bullet •).

Repeat parts (a) and (b) for these queries.

15. • Consider the query '*Find the sid and sname of each student who majors in 'CS' and who is (strictly) younger than some other student who also majors in 'CS'.*

16. • Consider the query '*Find each* $(s, b)$ *pair where* $s$ *is the sid of a student who bought book* $b$ *and such that each other book bought by* $s$ *is a book cited by* $b$*.*

17. • Consider the query '*Find each major that is not a major of any person who bought a book with title 'Databases' or 'Philosophy'.*

18. • Consider the query *'Find the bno and title of each book that is bought by a student who majors in CS and who is, among all students who major in CS, the next-to-oldest.*'

## 4 Expressing constraints using Relational Algebra

Recall that it is possible to express constraints as TRC sentences and as boolean SQL queries. It is also possible to express constraints using RA expressions. Let $E$, $F$, and $G$ denote RA expressions. Then we can express constraints using set comparisons between RA expressions as follows:

| | |
|---|---|
| $E \neq \emptyset$ | 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 4** 'Some students majors in CS.' *This constraint can be expressed as follows:*

$$\pi_{sid}(\sigma_{major=\mathbf{CS}}(hasMajor)) \neq \emptyset.$$

*Indeed, the RA expression*

$$\pi_{sid}(\sigma_{major=\mathbf{CS}}(hasMajor))$$

*computes the set of all student sids who major in CS. If this set is not empty then there are indeed students who major in CS.*

*Incidentally, the constraint* 'No one majors in CS' *can be written as follows:*

$$\pi_{sid}(\sigma_{major=\mathbf{CS}}(hasMajor)) = \emptyset.$$

**Example 5** Each student buys at least two books. *This constraint can be expressed as follows:*

$$\pi_{sid}(S) \subseteq \pi_{T_1.sid}(\sigma_{T_1.bno \neq T_2.bno}(T_1 \bowtie_{T_1.sid=T_2.pid} T_2)).$$

*Indeed,*

$$\pi_{sid}(S)$$

*computes the set of all student sids and*

$$\pi_{T_1.sid}(\sigma_{T_1.bno \neq T_2.bno}(T_1 \bowtie_{T_1.sid=T_2.pid} T_2))$$

*computes the set of all sids of student who buy at least two books. When the first set is contained in the second, we must have that each student buys at least two books.*

*Incidentally, the constraint* 'Some student buys fewer than two books" *can be written as follows:*

$$\pi_{sid}(S) \not\subseteq \pi_{T_1.sid}(\sigma_{T_1.bno \neq T_2.bno}(T_1 \bowtie_{T_1.sid=T_2.pid} T_2)).$$

We now turn to the problems in this section.

Express each of the following constraints using set comparisons between RA expressions as illustrated in Example 4 and Example 5.

19. • Among the books that cite a book, there are books that cite the same set of other books.

20. • Some student who majors only bought books that where bought by students who major in Math.

21. • There are pairs of majors that have no common students who have those majors.

22. • Attribute 'sid' in the relation hasMajor is a foreign key referencing the primary key 'sid' in the relation Student.

For each of the following practice problem, express constraints using set comparisons between RA expressions as illustrated in Example 4 and Example 5.

23. • No book cites each book that cost strictly less than $30.

24. • Each major is not the major of all students.

25. • Each pair of different students who share a major buy a book with the same bno.

26. • The attribute sid is a primary key of the Student relation.

## 5 Views

Express the following parameterized and recursive views. You are allowed to combine the features of both Pure SQL and RA SQL.

27. • Define a parameterized view '`PurchasedBookLessPrice(sid int, price integer)`' that returns, for a given student identified by '`sid`' and a given '`price`' value, the subrelation of '`Book`' of books that are bought by that student and that cost strictly less than '`price`'.

    Test your view for the parameter values (1001,15), (1001,30), (1001,50), and (1002,30).

28. • Define a parameterized view '`CitedBookbyMajor(major text)`' that returns for a major '`major`' the subrelation of '`Book`' of books that are cited by a book bought by a the student who majors in '`major`'.

    Test your view for each major.

29. • Define a parameterized view '`JointlyBoughtBook(sid1 integer, sid2 integer)`' that returns the subrelation of '`Book`' of books that are bought by both the students with sids '`sid1`' and '`sid2`', respectively.

    Test your view for the parameters (1001, 1002), (1001,1003), and (1002,1003).

30. • Let `PC`($parent$ : $integer, child$ : $integer$) be a rooted parent-child tree. So a pair $(n, m)$ in '`PC`' indicates that node $n$ is a parent of node $m$. The '`sameGeneration(n1, n2)`' binary relation is inductively defined using the following two rules:

    - If $n$ is a node in `PC`, then the pair $(n, n)$ is in the `sameGeneration` relation. (**Base rule**)
    - If $n_1$ is the parent of $m_1$ in `PC` and $n_2$ is the parent of $m_2$ in `PC` and $(n_1, n_2)$ is a pair in the `sameGeneration` relation then $(m_1, m_2)$ is a pair in the `sameGeneration` relation. (**Inductive Rule**)

    Write a recursive view for the `sameGeneration` relation. Test your view.

## 6 Theory about safe TRC and Relational Algebra

31. • Prove that each safe TRC formula $F(t_1, \ldots, t_n)$ can be brought into the following normal form

$$\bigvee_{i=1}^{k} R_1^i(t_1) \wedge \cdots \wedge R_n^(t_n) \wedge G^i j t_1, \ldots t_n)$$

where

- $R_i^j$ is a relation name for each $i \in [1, n]$ and $j \in [1, k]$; and
- $G^j(t_1, \ldots, t_n)$ is a TRC formula wherein no disjunctions ('$\vee$') occur for each $i \in [1, k]$.

32. • Prove that for each RA expression '$E$' with attributes $(A_1, \ldots, A_k)$, there exists a safe TRC formula '$F_E$' such that '$E$' and '$\{(A_1 : u_1(t_1), \ldots, A_k : u_k(t_1, \ldots, t_n)) \mid F_E(t_1, \ldots, t_n)\}$' express the same query.

33. • Prove that for each safe TRC formula '$F(t_1, \ldots, t_k)$' there exists a RA expressions '$E_F$' such that '$\{(A_1 : u_1(t_1), \ldots, A_k : u_k(t_1, \ldots, t_n)) \mid F_E(t_1, \ldots, t_n)\}$' and $E_F$ express the same query.