B561 Assignment 5
Fall 2022

Object-relational databases
Nested relational and semi-structured databases

Dirk Van Gucht

For this assignment, you will need the material covered in the lectures

- Lecture 12: Object-relational databases and queries
- Lecture 13: Nested relational, semi-structured databases, document databases

To turn in your assignment, you will need to upload to Canvas a single file with name `assignment5.sql` which contains the necessary SQL statements that solve the problems in this assignment. The `assignment5.sql` file must be so that the AI's can run it in their PostgreSQL environment. You should use the `Assignment-Script-2022-Fall-assignment5.sql` file to construct the `assignment5.sql` file. (Note that the data to be used for this assignment is included in this file.) In addition, you will need to upload a separate `assignment5.txt` file that contains the results of running your queries.

1 Preliminaries

1.1 Set Operations and Predicates

For the problems in this assignment, you will need to use the polymorphically defined functions and predicates that are defined in the document `SetOperationsAndPredicates.sql`.

Functions

set_union(A,B)	$A \cup B$
set_intersection(A,B)	$A \cap B$
set_difference(A,B)	$A - B$
add_element(x,A)	$\{x\} \cup A$
remove_element(x,A)	$A - \{x\}$
make_singleton(x)	$\{x\}$
bag_to_set(A)	coerce bag A to set

Predicates

is_in(x,A)	$x \in A$
is_not_in(x,A)	$x \notin A$
is_empty(A)	$A = \emptyset$
is_not_emptyset(A)	$A \neq \emptyset$
subset(A,B)	$A \subseteq B$
superset(A,B)	$A \supseteq B$
equal(A,B)	$A = B$
overlap(A,B)	$A \cap B \neq \emptyset$
disjoint(A,B)	$A \cap B = \emptyset$

1.2 Database Schema

In the database for this assignment, we maintain a set of students, `Student(sid,sname,birthYear)`, a set of books, `Book(bno,title,price)`, and a set of majors, `Major(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.[1] The `title` attribute in

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

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(sid,bno) relation. A triple (s, b) indicates that the student with sid s bought 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(sid,major) 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(bno1,bno2) 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.

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.

2 Problems

We now turn to the problems. You will need use the data provided for the `Student`, `Book`, `Buys`, `Major`, `hasMajor`, and `Cites` relations. But before turning to the problems, we will introduce various object-relational views defined over these relations:

- The view `studentBuysBooks(sid,books)` which associates with each student, identified by a sid, the set of bnos of books by that student.

```
create or replace view studentBuysBooks as
   select sid, array(select t.bno
                     from   Buys t
                     where  t.sid = s.sid order by 1) as books
   from   Student s order by 1;
```

- The view `bookBoughtByStudents(bno,students)` which associates with each book, identified by a bno, the set of sids of student who bought that book.

```
create or replace view bookBoughtByStudents as
   select bno, array(select t.sid
                     from   Buys t
                     where  t.bno = b.bno order by 1) as students
   from   Book b order by 1;
```

- The view `studentHasMajors(sid,majors)` which associates with each student, identified by a sid, the set of majors of that student.

```
create or replace view studentHasMajors as
   select sid, array(select hm.major
                     from   hasMajor hm
                     where  hm.sid = s.sid order by 1) as majors
   from   Student s order by 1;
```

- The view `majorOfStudents(major, students)` which associates with each major the set of sids of student who have that major.

```
create or replace view majorOfStudents as
   select major, array(select hm.sid
                       from   hasMajor hm
                       where  hm.major = m.major order by 1) as students
   from   Major m order by 1;
```

- The view `bookCitesBooks(bno,citedBooks)` which associates with each book, identified by a bno, the set of bnos cited by that book.

```
create or replace view bookCitesBooks as
   select bno, array(select c.bno2
                     from   Cites c
                     where  c.bno1 = b.bno order by 1) as citedBooks
   from   Book b order by 1;
```

- The view `bookCitedByBooks(bno,citingBooks)` which associates with each book, identified by a bno, the set of bnos of books citing that book.

```
create or replace view bookCitedByBooks as
   select bno, array(select c.bno1
                     from   Cites c
                     where  c.bno2 = b.bno order by 1) as citingBooks
   from   Book b order by 1;
```

2.1 Object-Relational Queries

For the problems in this section, you are asked to express queries in object-relational SQL. You should use the set operations and set predicates defined in the document `SetOperationsAndPredicates.sql`, the relations

```
Student(sid,sname,birthYear)
Book(bno,title,price)
Major(major)
```

and the views

```
studentBuysBooks(sid,books)
bookBoughtByStudents(bno,students)
studentHasMajors(sid,majors)
majorOfStudents(major, students)
bookCitesBooks(bno,citedBooks)
bookCitedByBooks(bno,citingBooks)
```

Crucially, you are **not** permitted to use the `Buys`, `hasMajor`, and `Cites` relations in the object-relation SQL formulation of the queries. Observe that you actually don't need these relations since they are encapsulated in these views.

Before listing the queries that you are asked to express, we present some examples of queries that are expressed in object-relational SQL using the assumptions stated in the previous paragraph. Your solutions need to be in the style of these examples. The goals is to maximize the utilization of the functions and predicates defined in document `SetOperationsAndPredicates.sql`.

Example 1 *Consider the query* "Find the bno of each book that is bought by a student who is born before 1997."

Here are several ways to express this simple query. In doing so, we show different operations and predicates to accomplish this. Note however, that we are not arguing that some ways are to be preferred over others.

Method 1 *Observe the* `IS_IN` *predicate:*

```
select distinct b.bno
from   Book b
where  true = some (select IS_IN(b.bno, sB.books)
                    from   studentBuysBooks sB
                    where  sB.sid in (select s.sid
                                      from   Student s
                                      where  s.birthYear < 1997));
```

Method 2 *Observe the* UNNEST *operation:*

```
select b.bno
from   Book b
where  b.bno in (select UNNEST(sB.books)
                 from   studentBuysBooks sB
                 where  sB.sid in (select s.sid
                                   from   Student s
                                   where  s.birthYear < 1997));
```

Method 3 *Observe the* UNNEST *operation and the need for using the* DISTINCT *clause:*

```
select DISTINCT UNNEST(sB.books) as bno
from   studentBuysBooks sB
where  sB.sid in (select sid
                  from   Student
                  where  birthYear < 1997);
```

Method 4 *Observe how we construct the* set of sids of students who are born before 1997 *using the* ARRAY *constructor and then use the* IS_IN *predicate:*

```
select b.bno
from   Book b
where  true = some (select IS_IN(sB.sid, ARRAY(select s.sid
                                               from   Student s
                                               where  s.birthYear < 1997)) and
                           IS_IN(b.bno,sB.Books)
                    from   studentBuysBooks sB);
```

Example 2 *Consider the query "*Find each pair (b, S) where b is the bno of a book and S is the set of sids of students who are born before 1997 and who bought that book.*"*

Note the `ARRAY` *(set) construction for each book* `b`*:*

```
select b.bno, ARRAY(select sB.sid
                    from   studentBuysBooks sB
                    where  sB.sid in (select s.sid
                                      from   Student s
                                      where  s.birthYear < 1997) and
                           IS_IN(b.bno,sB.books)) as "setOfStudentBornBefore1997WhoBoughtTheBook"
from   Book b;
```

Example 3 *Consider the query "*Find the set of bnos of books such that each book is this set is bought by some student who is born before 1997."

Note the `UNNEST` *operation followed by the* `ARRAY` *(set) construction:*

```
select ARRAY(select distinct unnest(sB.books)
             from   studentBuysBooks sB
             where  sB.sid in (select sid
                               from   Student
                               where  birthYear < 1997)) as "setOfBooksBoughtbyStudentsBornBefore1997";
```

Example 4 *Consider the query "*Find the sid and name of each student s who (1) has both the CS and Major majors and (2) buys at least 8 books*"*

Note the `ARRAY` *(set) construction* `'{"CS", "Math"}'` *and the* `SUBSET` *predicate. Also note the use of* `CARDINALITY` *function.*

```
select s.sid, s.sname
from   Student s
where  s.sid in (select sM.sid
                 from   studentHasMajors sM
                 where  SUBSET('"CS", "Math"', sM.majors)) and
       CARDINALITY((select sB.books
                    from   studentBuysBooks sB
                    where  sB.sid = s.sid)) >= 8;
```

Example 5 *Consider the query* "Find the sid and name of each student along with the set of his of her majors that are not among the majors of students who bought the book with bno 2000*".*

In this example, focus on (1) the SET_DIFFERENCE *operation and (2) the* UNNEST *operation followed by a* ARRAY *(set) construction.*

```
select s.sid, s.sname, SET_DIFFERENCE((select sM.majors
                                       from   studentHasMajors sM
                                       where  sM.sid = s.sid),
                                       ARRAY(select UNNEST(sM.majors)
                                             from   studentHasMajors sM
                                             where  is_in(sM.sid, (select bS.students
                                                                   from   bookBoughtByStudents bS
                                                                   where  bS.bno = 2000))))
from   Student s;
```

Express the following queries in object-relational SQL.

1. Find the sid and name of each student who bought the most books.
2. Find the sid and name of each student who bought the most books that cost strictly more than $25.
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.
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.
5. Find the bno and title of each book that cites a book and that was bought by a student who majors in CS but not in Math.
6. 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.
7. Find each major that is not a major of any person who bought a book with title 'Databases' or 'Complexity'.
8. Find each major that is the major of at least three students who bought a common book.
9. Find each student who has no major in common with those of students who bought a book with title 'Databases' or 'AI'.

10. Find the set of bnos of books that each cite strictly more than 4 books and that each are cited by fewer than 2 books. (So this query returns **only one** object, i.e., the set of bnos specified in the query.)

11. Find the sid and name of each student who has all the majors of the combined set of all majors of the oldest students who bought the book with bno 2000.

12. Find the following set of sets

$$\{M \mid M \subseteq \texttt{Major} \wedge |M| \leq 3\}.$$

I.e., this is the set consisting of each set of majors whose size (cardinality) is at most 3.

13. Reconsider Problem 12.

Let $\mathcal{M}$ denote the set $\{M \mid M \subseteq \texttt{Major} \wedge |M| \leq 3\}$.

Find the following set of sets

$$\{\mathcal{X} \mid \mathcal{X} \subseteq \mathcal{M} \wedge |\mathcal{X}| \leq 2\}.$$

14. Let t be a number called a *threshold*. We say that an (unordered) triple of different sids $\{s_1, s_2, s_3\}$ *co-occur* with frequency at least t if there are at least t books who are each bought by the students s_1, s_2, and s_3.

Write a function `coOccur(t integer)` that returns the (unordered) triples $\{s_1, s_2, s_3\}$ of bno that co-occur with frequency at least `t`.

Test your `coOccur` function for t in the range $[0, 3]$.

3 Nested Relations and Semi-structured databases

Consider the lecture on Nested relational and semi-structured databases. In that lecture we considered the `studentGrades` nested relation and the `jstudentGrades` semi-structured database and we constructed these using a PostgreSQL query starting from the `Enroll` relation.

15. Write a PostgreSQL view `courseGrades` that creates the nested relation of type

$$(\texttt{cno}, \texttt{gradeInfo}\{(\texttt{grade}, \texttt{students}\{(\texttt{sid})\})\})$$

This view should compute for each course, the grade information of the students enrolled in this course. In particular, for each course and for each grade, this relation stores in a set the sids students who obtained that grade in that course.

Test your view.

16. Starting from the `courseGrades` view in Problem 15 solve the following queries:

    (a) Find each pair (c, S) where c is the cno of a course and S is the set of sids of students who received an 'A' but not a 'B' in course c. The type of your answer relation should be $(\text{cno} : \text{text}, \text{Students} : \{(\text{sid} : \text{text})\})$.

    (b) Find each (s, C) pair where s is the sid of a students and C is the set of cnos of courses in which the student received an 'A' or a 'B' but not a 'C'. The type of your answer relation should be $(\text{sid} : \text{text}, \text{Courses} : \{(\text{cno} : \text{text})\})$.

17. Write a PostgreSQL view `jcourseGrades` that creates a semi-structured database which stores `jsonb` objects whose structure conforms with the structure of tuples as described for the `courseGrades` in Problem 15.

Test your view.

18. Starting from the `jcourseGrades` view in Problem 17 solve the following queries. Note that the output of each of these queries is a nested relation.

    (a) Find each pair (c, s) where c is the cno of a course and s is the sid of a student who received an 'A' in course c. The type of your answer relation should be (cno:text, sid:text).

(b) Find each pair $(\{c_1, c_2\}, S)$ where c_1 and c_2 are the course numbers of two different courses and S is the set of sids of students who received a 'A' in both courses c_1 and c_2. The type of your answer relation should be $(\text{coursePair} : \{(\text{cno} : \text{text})\}, \text{Students} : \{(\text{sid} : \text{text})\})$.