CS233 Database Systems - Worksheet 5

Relational Algebra and ISBL

The following exercises relate to the use of the EDDI relational algebra interpreter that provides the kernel of the SQLEDDI environment. The interpreter is loosely modelled on the Information Systems Base Language (ISBL) developed by Stephen Todd at IBM Peterlee in the 1970s. It has similar modes for defining and assigning relation variables, and uses the same syntax for operators, but differs from the original implementation of ISBL in the 1970s.

The exercises are based upon the simple `FRUITS` database. The EDDI data definitions required to specify the database are to be found in the file `fruits.eddi` in the `~empublic/projects/sqleddiWard2003` directory. Consult the Quick guide to EDDI document for more details about how to interpret the file `fruits.eddi`, and how to use the EDDI interpreter.

```
%eddi

allfruits (name CHAR, begin INT, end INT);
allfruits << ["granny",8,10],["lemon",5,12],["kiwi",5,6],["passion",5,7];
allfruits << ["orange",4,11],["grape",3,6],["lime",4,7],["pear",4,8];
allfruits << ["cox",1,12],["red",4,8];

apple (name CHAR, price REAL, qnt INT);
apple << ["cox",0.20,8],["red",0.35,4],["granny",0.25,10];

citrus (name CHAR, price REAL, qnt INT);
citrus << ["lime",0.30,3],["orange",0.55,8],["kiwi",0.75,5],["lemon",0.50,2];

soldfruit (name CHAR, unitsold INT);
soldfruit << ["cox",100],["granny",153],["red",70];
soldfruit << ["kiwi",23],["lime",15],["lemon",55],["orange",78];

fruits is allfruits % name;
popcitrus is (fruits.citrus % name).(soldfruit : unitsold > 50 % name);
```

---

0. Consult README.txt file for instructions on setting up the SQLEDDI environment.

1. To interrogate the `FRUITS` database, use queries of the form:

```
  ?allfruits;
```

The record for kiwi fruit indicates that it is in season from May to June. Introduce a new record to show that bananas are available from April to September. Add a new category of fruit resembling `citrus` and `apple`, called `other`, in which you include 10 bananas at 25 pence and 4 mangos at 65 pence. What is the result of the query `?other;` before any tuples have been entered into the table?

2. The most primitive algebraic operations in ISBL are as follows:
    Union (+), Difference (-), Intersection (.)

Introduce a new table mix using the following command:

```
    mix is apple + citrus;
```

How would you describe the contents of the `mix` table in words?
`mix` actually corresponds to a view in SQL, rather than the result of a query. Demonstrate this by adding a new variety of apple called `golden`.

3. Explore the potential and limitations of the EDDI system by trying to create other unions, such as:

```
    mix is apple + citrus + other;
    soldorapple is soldfruit + apple;
    combine is allfruits + citrus;
```

What issues are there in making sure that a relational algebra system can perform all the queries the users require, whilst not accepting absurd queries? What features of EDDI are needed for it to meet this requirement? Experiment with views based on the other primitive relational operators. Interpret the results of defining new tables as follows:

```
    x is apple - apple;
    x is apple + citrus - apple;
    x is (apple + citrus) - apple;
```

4. Relational algebra operators include projection and selection. The syntax for these operators is illustrated in the two definitions at the end of the `fruits.eddi` file. By experimenting with the EDDI system, interpret the views `fruits` and `popcitrus` that are defined in the file `fruits.eddi`. Note that
(a) you can introduce auxiliary variables to represent subexpressions e.g.

```
  z is fruits.citrus % name;
```

(b) the algebraic operations are subject to type checking and precedence e.g.

```
  z is (fruits.citrus)%name;
```

Compare the results of the following queries that make use of the selection and projection operators:

```
  ?(citrus % name, price): price > 0.31;
  ?citrus : price > 0.31 % name, price;
  ?citrus % name, price: price > 0.31;
```

What does this reflect about the precedence of the projection and selction operators and the limitations of the EDDI parser?

5. EDDI does not allow a selection that allows a composite predicate, such as

```
  allfruits: begin < 5 and end > 7;.
```

Describe two ways in which you can formulate queries that realise the same effect as a compound query such as are needed to specify the fruits that are available before June and after November. How would you deal with a query concerning fruits available before June *or* after November? What about more general queries involving complex predicates? - as in

```
  allfruits:(begin < 5 and end > 7) or name == "granny";
```

Can you formulate an expression to represent: the varieties of apples and citrus fruit that have been sold? which of these varieties that have been sold cost more than 33 pence? which of these varieties cost more than 33 pence and have sold more than 60? Explain why you can't use the relational algebra operators introduced so far to construct a single table that, for a given set of varieties of fruit, lists the prices and the quantities sold.

6. There is an additional operator in ISBL, called join (*). Inspect the result of joining the `apple` and `soldfruit` tables. By exploiting this join, construct an expression:
(a) to list the varieties of apples that cost more than 22 pence and of which at least a hundred have been sold;
(b) to create a table to show the prices and quantity sold for each variety of apple listed in (a).
Observe how your answer to (b) can be formulated as a projection from a selection from a join.
Show how the strategy of using joins as in (a) can be exploited to solve the queries in exercise 4. What might be the disadvantage of using such a strategy in a real-world database context?

7. Devise an example to show the distinction between the definition of a view X and the assignment of a table Y that make use of the same relational expression on the RHS. For instance, formulate a definition

```
  X is f(R,S);
```

and an assignment of the form

```
  Y = f(R,S)
```

where f() is a relational expression whose operands `R` and `S` are relations in the `FRUITS` database, and demonstrate how a change to R or S can affect X but not Y.

8. Use the project-and-rename operator in EDDI:
(a) to create a relation of type `[name CHAR, beginorend INT]` that consists of pairs that link the name of a fruit to the index of the month in which its season either begins or ends.
(b) to generate a result similar to that which is returned by the SQL query

```
SELECT * FROM APPLE, ALLFRUITS
```

(c) to create a variant of the ALLFRUITS table as a view of type `[name CHAR, first CHAR, last CHAR]` where the `begin` and `end` fields are replaced by abbreviated forms of the names of the associated months, as recorded in a separate `months` table in which typical tuples are [1,"Jan"], [8, "Aug"] etc.

9. Compare and contrast the characteristics of EDDI queries with those of SQL queries e.g. assessing how easily they can be understood by users, considering the extent to which SQL queries can be translated into EDDI queries, and identifying the principal respects in which the results of EDDI queries differ from those of standard SQL queries.