We use the Sailors, Reserves, and Boats schema for all our queries.

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Sailors S3

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Reserves R2

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Boats B1

Queries using Relational Algebra

Q1. Find the names of sailors who have reserved boat 103.

A: $\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$, which results Dustin, Lubber and Horatio.

Q2. Find the names of sailors who have reserved a red boat.

A: $\pi_{sname}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$.

This query involves a series of two joins. First, we choose (tuples describing) red boats. Then, we join this set with Reserves (natural join, with equality specified on the bid column) to identify reservations of red boats. Next, we join the resulting intermediate relation with Sailors (natural join, with equality specified on the *sid* column) to retrieve the names of sailors who have made reservations for red boats. Finally, we project the sailors' names. The answer, when evaluated on the instances B1, R2, and S3, contains the names Dustin, Horatio, and Lubber.

An equivalent expression, which generates intermediate relations with fewer fields (and is therefore likely to result in intermediate relation instances with fewer tuples as well).

$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$.

Q3. Find the colors of boats reserved by Lubber.

A: $\pi_{color}((\sigma_{sname='Lubber'} Sailors) \bowtie Reserves \bowtie Boats)$.

This query is very similar to the query we used to compute sailors who reserved red boats. On instances Bl, R2, and S3, the query returns the colors green and red.

Q4. Find the names of sailors who have reserved at least one boat.

A: $\pi_{sname}(Sailors \bowtie Reserves)$

The join of Sailors and Reserves creates an intermediate relation in which tuples consist of a Sailors tuple 'attached to' a Reserves tuple. A Sailors tuple appears in (some tuple of) this intermediate relation only if at least one Reserves tuple has the same *sid* value, that is, the sailor has made some

reservation. The answer, when evaluated on the instances Bl, R2 and S3, contains the three tuples (Dustin), (Horatio) , and (Lubber). Even though two sailors called Horatio have reserved a boat,

the answer contains only one copy of the tuple (Horatio) , because the answer is a relation, that is, a set of tuples, with no duplicates.

Q5. Find the names of sailors who have reserved a *red* or a *green* boat.

A: $\sigma\left(Tempboats, (\sigma_{color='red'}Boats) \cup (\sigma_{color='green'}Boats)\right)$
$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$

We identify the set of all boats that are either red or green (Tempboats, which contains boats with the bids 102, 103, and 104 on instances B1, R2, and S3). Then we join with Reserves to identify *sids* of sailors who have reserved one of these boats; this gives us *sids* 22, 31, 64, and 74 over our example instances. Finally, we join (an intermediate relation containing this set of *sids*) with Sailors to find the names of Sailors with these *sids*. This gives us the names Dustin, Horatio, and Lubber on the instances B1, R2, and S3. Another equivalent definition is following.

$$\sigma\left(Tempboats, (\sigma_{color='red' \vee color='green'}Boats)\right)$$
$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

Q6. Find the names of sailors who reserved a red and a green boat.

A: It is very tempting to try to do this query by simply replacing $\cap$ *by* $\cup$ in the query Q5 in the definition of Tempboats.

$$\sigma\left(Tempboats, (\sigma_{color='red'}Boats) \cap (\sigma_{color='green'}Boats)\right)$$
$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

However, this solution is incorrect – it instead tries to compute sailors who have reserved a boat that is both red and green.

The correct approach is to find sailors who have reserved a red boat, then sailors who have reserved a green boat, and then take the intersection of these two sets:

$$\sigma\left(Tempred, \pi_{sid}((\sigma_{color='red'}Boats) \bowtie Reserves)\right)$$
$$\sigma\left(Tempgreen, \pi_{sid}\left((\sigma_{color='green'}Boats) \bowtie Reserves\right)\right)$$
$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Q7. Find the names of sailors who have reserved at least two boats.

A:

1. $\rho\left(Reservations, \pi_{sid,sname,bid}(Sailors \bowtie Reserves)\right)$
2. $\rho(Reservationpairs(1 \to sid1, 2 \to sname1, 3 \to bid1, 4 \to sid2, 5 \to sname2, 6 \to bid2), Reservations \times Reservations)$

3. $\pi_{sname1}\sigma_{sid1=sid2\wedge(bid1\neq bid2)}Reservationpairs$

First we compute tuples of the form <*sid,sname,bid*>, where sailor sid has made a reservation for boat bid; this set of tuples is the temporary relation Reservations. Next we find all pairs of

Reservations tuples where the same sailor has made both reservations and the boats involved are distinct. Here is the central idea: In order to show that a sailor has reserved two boats, we

must find two Reservations tuples involving the same sailor but distinct boats. Over instances B1, R2, and S3, the sailors with sids 22, 31, and 64 have each reserved at least two boats.

Finally, we project the names of such sailors to obtain the answer, containing the names Dustin, Horatio, and Lubber.

Q8. Find the sids of sailors with age over 20 who have not reserved a red boat.

A:

$$\pi_{sid}(\sigma_{age>20}Sailors) = \pi_{sid}((\sigma_{color='red'}Boats) \bowtie Reserves \bowtie Sailors)$$

This query illustrates the use of the set-difference operator. Again, we use the fact that sid is the key for Sailors. We first identify sailors aged over 20 (over instances B1, R2, and S3, sids 22, 29, 31, 32, 58, 64, 74, 85, and 95) and then discard those who have reserved a red boat (sids 22, 31, and 64), to obtain the answer (sids 29, 32, 58, 74, 85, and 95). If we want to compute the names of such sailors, we must first compute their sids (as shown above), and then join with Sailors and project the sname values.

Q9. Find the names of sailors who have reserved all boats.

A: The use of the word all (or every) is a good indication that the division operation might be applicable:

1. $\rho(Tempsids, (\pi_{sid,bid}\ Reserves)/(\pi_{bid}\ Boats))$
2. $\pi_{sname}(Tempsids \bowtie Sailors)$

The intermediate relation Tempsids is defined using division, and computes the set of sids of sailors who have reserved every boat (over instances B1, R2, and S3, this is just sid 22). Notice how we define the two relations that the division operator (=) is applied to – the first relation has the schema (sid,bid) and the second has the schema (bid). Division then returns all sids such that there is a tuple (sid,bid) in the first relation for each bid in the second. Joining Tempsids with Sailors is necessary to associate names with the selected sids; for sailor 22, the name is Dustin.

Q10. Find the names of sailors who have reserved all boats called *Interlake*.

A:

$$\rho(Tempsids, (\pi_{sid,bid}\ Reserves)/(\pi_{bid}\ (\sigma_{bname='Interlake'}\ Boats))\ ))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

The only difference with respect to the previous query is that now we apply a selection to Boats, to ensure that we compute only bids of boats named *Interlake* in defining the second argument to the division operator. Over instances B1, R2, and S3, Tempsids evaluates to sids 22 and 64, and the answer contains their names, Dustin and Horatio.

Queries using TRC:

Q11. Find all the sailors with a rating above 7.

A: $\{S|S \in Sailors\ \wedge S.rating > 7\}$

When this query is evaluated on an instance of the Sailors relation, the tuple variable S is instantiated successively with each tuple, and the test S.rating>7 is applied. The answer contains those instances of S that pass this test. On instance S3 of Sailors, the answer contains Sailors tuples with sid 31, 32, 58, 71, and 74.

Q12. Find the names and ages of sailors with a rating above 7.

A:

$$\{P|\exists S \in Sailors(S.rating > 7 \wedge\ P.name = S.sname \wedge\ P.age = S.age)\}$$

This query illustrates a useful convention: P is considered to be a tuple variable with exactly two fields, which are called name and age, because these are the only fields of P that are mentioned and P does not range over any of the relations in the query; that is, there is no subformula of the form P ∈ Relname. The result of this query is a relation with two fields, name and age. The atomic formulas P:name = S:sname and P:age = S:age give values to the fields of an answer tuple P. On instances B1, R2, and S3, the answer is the set of tuples (Lubber; 55:5), (Andy; 25:5), (Rusty; 35:0), (Zorba; 16:0), and (Horatio; 35:0)

Q13. Find the sailor name, boat id, and reservation date for each reservation.

A:

$$\{P|\exists R \in Reserves\ \exists S \in Sailors\ (R.sid = S.sid \wedge P.bid = R.bid \wedge P.day = R.day \wedge P.sname = S.sname)\}.$$

For each Reserves tuple, we look for a tuple in Sailors with the same sid. Given a pair of such tuples, we construct an answer tuple P with fields *sname*, *bid*, and *day* by copying the corresponding fields from these two tuples. This query illustrates how we can combine values from different relations in each answer tuple. The answer to this query on instances B1, R2, and S3 is

sname	*bid*	*day*
Dustin	101	10/10/98
Dustin	102	10/10/98
Dustin	103	10/8/98
Dustin	104	10/7/98
Lubber	102	11/10/98
Lubber	103	11/6/98
Lubber	104	11/12/98
Horatio	101	9/5/98
Horatio	102	9/8/98
Horatio	103	9/8/98

Q1. Find the names of sailors who have reserved boat 103.

A: $\{P|\exists S \in Sailors\ \exists R \in Reserves(R.sid = S.sid \wedge R.bid = 103 \wedge P.sname = S.sname)\}$

This query can be read as follows: "Retrieve all sailor tuples for which there exists a tuple in Reserves, having the same value in the sid field, and with bid=103". This is, for each sailor tuple, we look for a tuple in Reserves that shows that this sailor has reserved boat 103. The answer tuple

P contains just one tuple, sname.

Q2. Find the names of sailors who have reserved a red boat.

A:

$$\left\{P \middle| \exists S \in Sailors\ \exists R \in Reserves \begin{pmatrix} R.sid = S.sid \wedge P.sname = S.sname \wedge \\ \exists B \in Boats(B.bid = R.bid \wedge B.color =' red') \end{pmatrix}\right\}$$

This query can be read as follows: \Retrieve all sailor tuples S for which there exist tuples R in Reserves and B in Boats such that S:sid = R:sid, R:bid = B:bid, and B:color ='red'." Another way to write this query, which corresponds more closely to this reading, is as follows:

$$\left\{P \middle| \begin{matrix} \exists S \in Sailors\ \exists R \in Reserves\ \exists B \in Boats \\ (R.sid = S.sid \wedge B.bid = R.bid \wedge B.color =' red' \wedge P.sname = S.sname) \end{matrix} \right\}$$

Q7. Find the names of sailors who have reserved at least two boats.

A:

$$\left\{P \middle| \begin{matrix} \exists S \in Sailors\ \exists R1 \in Reserves\ \exists R2 \in Reserves \\ (S.sid = R1.sid \wedge R1.sod = R2.sid \wedge R1.bid \neq R2.bid \wedge P.sname = S.sname) \end{matrix} \right\}$$

Contrast this query with the algebra version and see how much simpler the calculus version is. In part, this difference is due to the cumbersome renaming of fields in the algebra version, but the calculus version really is simpler.

Q9. Find the names of sailors who have reserved all boats.

A:

$$\left\{P \middle| \exists S \in Sailors\ \forall B \in Boats\ \left(\begin{matrix} \exists R \in Reserves \\ (S.sid = R.sid \wedge R.bid = B.bid \wedge P.sname = S.sname) \end{matrix} \right) \right\}$$

This query was expressed using the division operator in relational algebra. Notice how easily it is expressed in the calculus. The calculus query directly reflects how we might express the query in

English: “Find sailors S such that for all boats B there is a Reserves tuple showing that sailor S has reserved boat B."

Q14. Find sailors who have reserved all red boats.

A:

$$\{S | S \in Sailors \wedge \forall B \in Boats\ (B.color =' red' \Rightarrow (\exists R \in Resources(S.sid = R.sid \wedge R.bid = B.bid)))$$

This query can be read as follows: For each candidate (sailor), if a boat is red, the sailor must have reserved it. That is, for a candidate sailor, a boat being red must imply the sailor having reserved it.

Observe that since we can return an entire sailor tuple as the answer instead of just the sailor's name, we have avoided introducing a new free variable (e.g., the variable P in the previous example) to hold the answer values. On instances B1, R2, and S3, the answer contains the Sailors

tuples with sids 22 and 31.

We can write this query without using implication, by observing that an expression of the form $p \Rightarrow q$ is logically equivalent to $p' \vee q$:

$$\{S|S \in Sailors \wedge \forall B \in Boats\ (B.color \neq' red' \vee (\exists R \in Resources(S.sid = R.sid \wedge R.bid = B.bid)))$$

This query should be read as follows: "Find sailors S such that for all boats B, either the boat is not red or a Reserves tuple shows that sailor S has reserved boat B."

Queries using DRC:

Q11. Find all the sailors with a rating above 7.

A: $\{(I,N,T,A)|(I,N,T,A) \in Sailors \wedge T > 7\}$

This differs from the TRC version in giving each attribute a (variable) name. The condition <I;N; T;A> ∈ Sailors ensures that the domain variables I, N, T, and A are restricted to be fields of the

same tuple. In comparison with the TRC query, we can say T > 7 instead of S.rating > 7, but we must specify the tuple (I, N, T, A) in the result, rather than just S.

Q1. Find the names of sailors who have reserved boat 103.

A:

$$\left\{(N)\middle|\exists I,T,A\left(\begin{matrix}(I,N,T,A) \in Sailors \\ \wedge \exists Ir,Br,D\big((Ir,Br,D) \in Reserves \wedge Ir = I \wedge Br = 103\big)\end{matrix}\right)\right\}$$

Notice that only the sname field is retained in the answer and that only N is a free variable. We use the notation $\exists Ir,Br,D(\ldots)$ as a shorthand for $\exists Ir\left(\exists Br\big(\exists D(\ldots)\big)\right)$. Very often, all the quantified variables appear in a single relation, as in this example. An even more compact notation

in this case is $\exists(Ir,Br,D) \in Reserves$. With this notation, which we will use henceforth, the above query would be as follows:

$$\left\{(N)\middle|\exists I,T,A\big((I,N,T,A) \in Sailors \wedge \exists(Ir,Br,D) \in Reserves(Ir = I \wedge Br = 103)\big)\right\}$$

The comparison with the corresponding TRC formula should now be straightforward. This query can also be written as follows; notice the repetition of variable I and the use of the constant 103:

$$\left\{(N)\middle|\exists I,T,A\left((I,N,T,A) \in Sailors \wedge \exists D\big((I,103,D) \in Reserves\big)\right)\right\}$$

Q2. Find the names of sailors who have reserved a red boat.

A:

$$\{(N) | \exists I, T, A((I, N, T, A) \in Sailors \wedge \exists (I, Br, D) \in Reserves \wedge \exists (Br, BN, 'red') \in Boats)\}$$

Q7. Find the names of sailors who have reserved at least two boats.

A:

$$\left\{(N) \middle| \exists I, T, A \left(\exists Br1, Br2, D1, D2 \begin{pmatrix} (I, N, T, A) \in Sailors \wedge \\ (I, Br1, D1) \in Reserves \wedge \\ (I, Br2, D2) \in Reserves \wedge Br1 \neq Br2 \end{pmatrix} \right) \right\}$$

Q9. Find the names of sailors who have reserved all boats.

A:

$$\left\{(N) \middle| \exists I, T, A \left(\forall B, BN, C \begin{pmatrix} (I, N, T, A) \in Sailors \wedge \\ \neg((B, BN, C) \in Boats) \vee \\ (\exists (It, Br, D) \in Reserves (I = Ir \wedge Br = B)) \end{pmatrix} \right) \right\}$$

This query can be read as follows: "Find all values of N such that there is some tuple (I,N, T,A) in Sailors satisfying the following condition: for every (B; BN;C), either this is not a tuple in Boats or there is some tuple (Ir;Br;D) in Reserves that proves that Sailor I has reserved boat B." The ∀ quantifier allows the domain variables B, BN, and C to range over all values in their respective attribute domains, and the pattern `$\neg((B; BN; C) \in Boats) \vee '$ is necessary to restrict attention to those values that appear in tuples of Boats. This pattern is common in DRC formulas, and the notation $\forall(B, BN, C) \in$ Boats can be used as a shorthand instead.

Q14. Find sailors who have reserved all red boats.

A:

$$\left\{(I, N, T, A) \middle| \begin{pmatrix} (I, N, T, A) \in Saliors \wedge \forall (B, BN, C) \in Boats \\ (C =' red' \Rightarrow \exists (Ir, Br, D) \in Reserves (I = Ir \wedge Br = B)) \end{pmatrix} \right\}$$

Here, we find all sailors such that for every red boat there is a tuple in Reserves that shows the sailor has reserved it.