

Chapter 6

R e l a t i o n a l A l g e b r a

6.2 JOIN is discussed in Section 6.4. INTERSECT can be defined as follows:

$$A \text{ INTERSECT } B = A \text{ MINUS } (A \text{ MINUS } B)$$

or (equally well)

$$A \text{ INTERSECT } B = B \text{ MINUS } (B \text{ MINUS } A)$$

These equivalences, though valid, are slightly unsatisfactory, since $A \text{ INTERSECT } B$ is symmetric in A and B and the other two expressions are not. Here by contrast is a symmetric equivalent:

$$(A \text{ MINUS } (A \text{ MINUS } B)) \text{ UNION } (B \text{ MINUS } (B \text{ MINUS } A))$$

Note: Given that A and B must be of the same type, we also have:

$$A \text{ INTERSECT } B = A \text{ JOIN } B$$

As for DIVIDEBY, we have:

$$A \text{ DIVIDEBY } B \text{ PER } C = A \{ X \} \\ \text{MINUS } ((A \{ X \} \text{ TIMES } B \{ Y \}) \\ \text{MINUS } C \{ X, Y \}) \{ X \}$$

Here X is the set of attributes common to A and C and Y is the set of attributes common to B and C .

Note: DIVIDEBY as just defined is actually a generalization of the version defined in the body of the chapter -- though it is still a Small Divide [6.3] -- inasmuch we assumed previously that A had no attributes apart from X , B had no attributes apart from Y , and C had no attributes apart from X and Y . The foregoing generalization would allow, e.g., the query "Get supplier numbers for suppliers who supply all parts," to be expressed more simply as just

$$S \text{ DIVIDEBY } P \text{ PER } SP$$

instead of (as previously) as

$S \{ S\# \} \text{ DIVIDEBY } P \{ P\# \} \text{ PER } SP \{ S\#, P\# \}$

6.3 $A \text{ INTERSECT } B$ (see the answer to Exercise 6.2 above). *Note:* We remark that since `TIMES` is a special case of `JOIN`, we could regard `JOIN` as a primitive operator instead of `TIMES` -- a preferable alternative, in fact, precisely because it is more general.

6.5 The short answer is no. Codd's original `DIVIDEBY` did satisfy the property that

$$(A \text{ TIMES } B) \text{ DIVIDEBY } B = A$$

However:

- Codd's `DIVIDEBY` was a dyadic operator; our `DIVIDEBY` is triadic, and hence cannot possibly satisfy a similar property.
- In any case, even with Codd's `DIVIDEBY`, dividing A by B and then forming the Cartesian product of the result with B will yield a relation that *might* be identical to A , but is more likely to be some proper subset of A :

$$(A \text{ DIVIDEBY } B) \text{ TIMES } B \leq A$$

(using " \leq " to mean "is a subset of"). Codd's `DIVIDEBY` is thus more analogous to *integer* division in ordinary arithmetic (i.e., it ignores the remainder).

6.6 The trap is that the join involves the `CITY` attributes as well as the `S#` and `P#` attributes. The result looks like this:

```

+-----+
-+
| S# | SNAME | STATUS | CITY   | P# | QTY | PNAME | COLOR | WEIGHT |
+-----+-----+-----+-----+-----+-----+-----+-----+
| S1 | Smith | 20     | London | P1 | 300 | Nut   | Red   | 12.0   |
| S1 | Smith | 20     | London | P4 | 200 | Screw | Red   | 14.0   |
| S1 | Smith | 20     | London | P6 | 100 | Cog   | Red   | 19.0   |
| S2 | Jones | 10     | Paris  | P2 | 400 | Bolt  | Green | 17.0   |
| S3 | Blake | 30     | Paris  | P2 | 200 | Bolt  | Green | 17.0   |
| S4 | Clark | 20     | London | P4 | 200 | Screw | Red   | 14.0   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

6.7 2^n . This count includes the *identity* projection (i.e., the projection over all n attributes), which yields a result that is identical to the original relation A , and the *nullary* projection

(i.e., the projection over no attributes at all), which yields TABLE_DUM if the original relation A is empty and TABLE_DEE otherwise [5.5].