

Chapter 9

V i e w s

9.1 We have numbered the following solutions as 9.1.*n*, where *n* is the number of the original example in Section 9.1. We make our usual assumptions regarding range variables.

9.1.1 VAR REDPART VIEW

```
( PX.P#, PX.PNAME, PX.WEIGHT AS WT, PX.CITY )
  WHERE PX.COLOR = COLOR ( 'Red' ) ;
```

9.1.2 VAR PQ VIEW

```
( PX.P#,
  SUM ( SPX WHERE SPX.P# = PX.P#, QTY ) AS TOTQTY ) ;
```

9.1.3 VAR CITY_PAIR VIEW

```
( SX.CITY AS SCITY, PX.CITY AS PCITY )
  WHERE EXISTS SPX ( SPX.S# = SX.S# AND
                    SPX.P# = PX.P# ) ;
```

9.1.4 VAR HEAVY_REDPART VIEW

```
RPX WHERE RPX.WT > WEIGHT ( 12.0 ) ;
```

RPX here is a range variable that ranges over REDPART.

9.2 VAR NON_COLOCATED VIEW

```
( S TIMES P ) { S#, P# } MINUS
( S JOIN P ) { S#, P# } ;
```

9.3 VAR LONDON_SUPPLIER VIEW

```
( S WHERE CITY = 'London' ) { ALL BUT CITY } ;
```

Note: We omit the CITY attribute here because we know its value must be London for every supplier in the view. Observe, however, that this omission means that any INSERT on the view will necessarily fail (unless the default value for attribute CITY in the underlying suppliers relvar happens to be London). In other words, a view like this one probably cannot support INSERT operations at all. (Alternatively, we might consider the possibility of defining the default value for CITY *for tuples inserted via this view* to be London. This idea of **view-specific defaults** requires more study.)

9.4 The question here is: How should attribute QTY be defined in the SP view? The sensible answer seems to be that, for a given

S#-P# pair, it should be the *sum* of all SPJ.QTY values, taken over all J#'s for that S#-P# pair:

```
VAR SP VIEW
  SUMMARIZE SPJ PER SPJ { S#, P# }
  ADD SUM ( QTY ) AS QTY ;
```

9.5 VAR JC VIEW

```
( ( SPJ WHERE S# = S# ( 'S1' ) ) { J# } JOIN
  ( SPJ WHERE P# = P# ( 'P1' ) ) { J# } ) JOIN
  J { J#, CITY } ;
```