ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

this is not a problem to us in this case, then we can proceed to use the revised design, though, as we noted, we would still have the redundancy problem.

7.1.1 Decomposition

The only way to avoid the repetition-of-information problem in the *in_dep* schema is to decompose it into two schemas (in this case, the *instructor* and *department* schemas). Later on in this chapter we shall present algorithms to decide which schemas are appropriate and which ones are not. In general, a schema that exhibits repetition of information may have to be decomposed into several smaller schemas.

Not all decompositions of schemas are helpful. Consider an extreme case in which all schemas consist of one attribute. No interesting relationships of any kind could be expressed. Now consider a less extreme case where we choose to decompose the *employee* schema (Section 6.8):

$$employee\ (ID, name, street, city, salary)$$

into the following two schemas:

$$employee1\ (ID, name)$$
$$employee2\ (name, street, city, salary)$$

The flaw in this decomposition arises from the possibility that the enterprise has two employees with the same name. This is not unlikely in practice, as many cultures have certain highly popular names. Each person would have a unique employee-id, which is why *ID* can serve as the primary key. As an example, let us assume two employees,