E-R MODELLING HINT

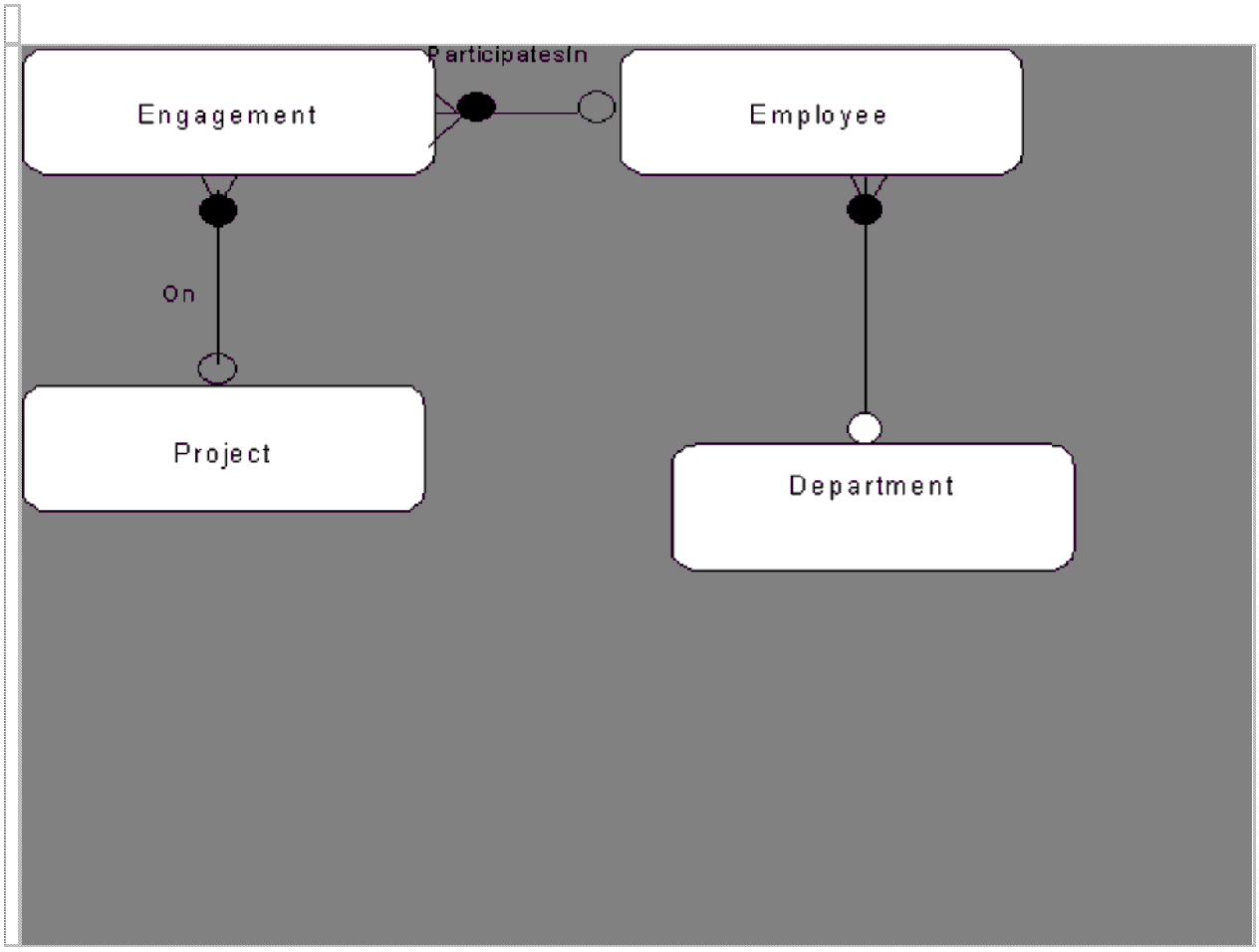
Hugh Darwen, Region 4

Relationships and Foreign Keys

This is about an issue that many students lose marks over every year, including some of my own students, even though I do always try very hard to explain the rather difficult and confusing "rule". **How about this year being the first year where none of my students lose marks on it?**

Consider the following relationships:

E--R Diagram



Entity types

Employee (EmployeeId, Hiredate, Name, Salary)

Department (DeptId, Budget)

Project (ProjectId, Title)

Engagement (EmployeeId, ProjectId, StartingDate)

Notice that Engagement has a ProjectId attribute, whereas Employee does not have a DeptId attribute. You might ask why Employee doesn't have a DeptId attribute, given the existence of the WorksIn relationship. Don't we need an attribute indicating which department the employee works in? After all, the diagram tells

us that every employee does work in some department. Let me try to explain.

The relationship between Employee and Department is represented by the line labelled WorksIn. **We don't need an attribute as well as the line.** Yes, if we choose to design a relational database from this E-R model, then the corresponding Employee *relation* does have DepartmentId attribute but that's another matter. Entity types aren't relations.

A DepartmentId attribute in Employee would be redundant. It would (presumably) be expressing the same thing as that WorksIn line. Worse than that, if it *does* represent that same thing, then we would need to write an entry in the Constraints section, to the effect that the DepartmentId of the Department that an Employee is connected to via WorksIn must be the same as that Employee's DepartmentId.

Now, if you're with me so far, you might now be wondering, in that case, how come there's a ProjectId attribute in Engagement? And an EmployeeId attribute, for that matter. It appears that the Engagement entity type looks *exactly* like its corresponding relation, in which EmployeeId and ProjectId are foreign keys referencing Employee and Project, respectively.

Well, it so happens that some E-R practitioners do insist on leaving out those "foreign key" attributes, even when they form identifiers, as in Engagement. But the M359 course development team, like their predecessors on the old M357 and M358 courses, belonged to the camp that deems it essential to have an *identifier* for every entity type. You might or might not agree with that position, but it does have some disadvantages:

1. Students have to remember to leave out the "foreign key" attributes in cases like WorksIn but to put them back in again in cases like Engagement; otherwise they lose marks.
2. Aren't those identifying attributes in Engagement redundant with the relationship lines, just like the outlawed DepartmentId attribute of Employee would be? And don't we therefore need extra constraints to the effect that the ProjectId of an Engagement must be the same as that of the Project to which it is connected and the EmployeeId of an Engagement must be the same as that of the Employee to which it is connected? Good questions! They are addressed in M359 by the notion of ***weak entity types*** (see Block 4, page 35) . Our Engagement entity type is one of these. Think of it as "weak" because it can be thought of as representing a set of relationships as well as a set of entities. Indeed, some practitioners wouldn't even put put Engagement into an entity type box. Instead, they would have a many-to-many relationship line, EngagedIn, from Employee to Project, and allow that relationship, EngagedIn, to have attributes and be connected to other entity types or other relationships.

To sum up:

Don't include "foreign key" attributes in entity type definitions, unless they are needed to form identifiers.

Notice, by the way, my quotes around "foreign key". I strongly recommend that you forget about relational terminology altogether when you are doing E-R models. For example, the term *primary key* doesn't exist in the E- R world. Instead we have identifiers. And entity types are not relations. Yes, both worlds do have attributes, apparently standing for much the same concept. Sorry about that, but it's a mere coincidence.

END