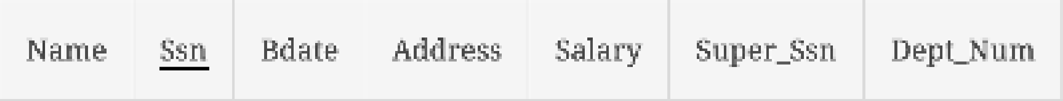
Relational Database Schemas

In this lesson, we will discuss the basic concepts behind relational database schemas.

Relational database schemas

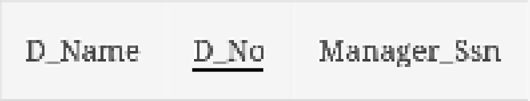
A relational database schema **S** is a set of relation schemas S = {R_1, R_2, ... , R_m} and a set of integrity constraints **IC**. A relational database state **DB** of S is a set of relation states DB = {r_1, r_2, ... , r_m} such that each r_i is a state of R_i. The figure below shows a relational database schema that we call COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, DEPENDENT}. In each relation schema, the underlined attribute represents the primary key.

EMPLOYEE



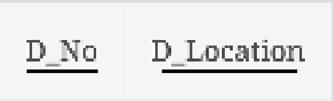
Name	Ssn	Bdate	Address	Salary	Super_Ssn	Dept_Num

Department



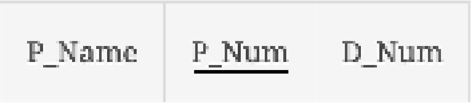
D_Name	D_No	Manager_Ssn

Dept_Locations



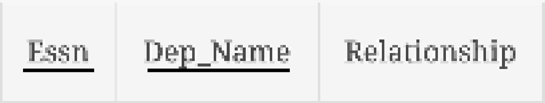
D_No	D_Location

Project



P_Name	P_Num	D_Num

Dependent



Essn	Dep_Name	Relationship

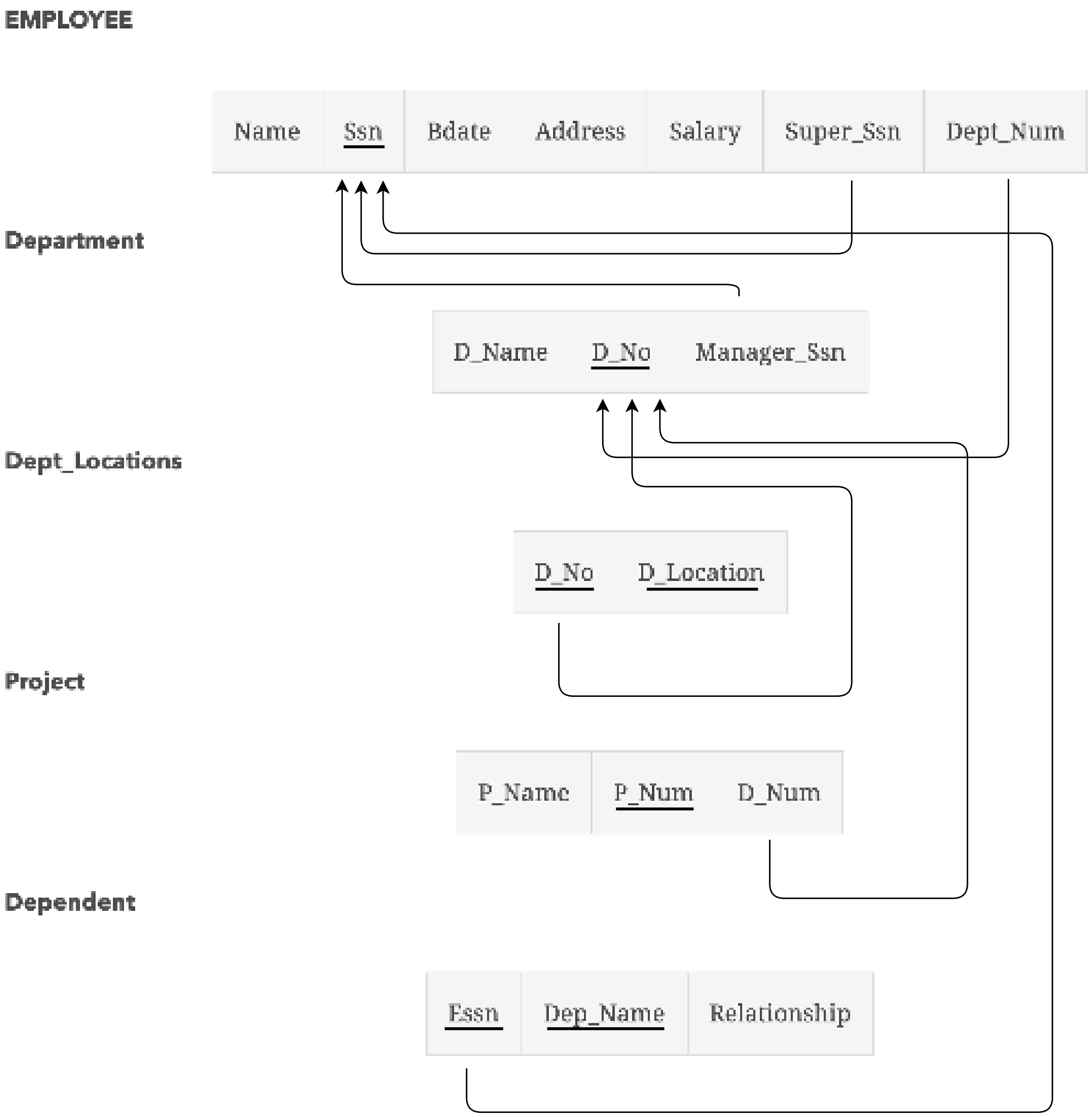
In the diagram above, the `D_No` attribute in both DEPARTMENT and DEPT_LOCATIONS stands for the same real-world concept—the number given to a department. That same concept is called `Dept_Num` in EMPLOYEE and `D_Num` in PROJECT. Attributes that represent the same real-world concept may or may not have identical names in different relations.

Representing referential integrity constraints in a schema

Referential integrity constraints typically arise from the relationships among the entities represented by the relation schemas. Consider the above example in which the attribute `Dept_Num` in the EMPLOYEE relation, refers to the department for which an employee works. Hence, we designate `Dept_Num` to be a foreign key of EMPLOYEE referencing the DEPARTMENT relation.

This means that a value of `Dept_Num` in any tuple t_i of the EMPLOYEE relation must match a value of the primary key of DEPARTMENT—the `D_No` attribute—in some tuple t_j of the DEPARTMENT relation. It could also be the case that the value of `Dept_Num` can be `NULL` if the employee does not belong to a department or will be assigned to a department later.

We can diagrammatically display referential integrity constraints by drawing a directed arc from each foreign key to the relation it references. For clarity, the arrowhead may point to the primary key of the referenced relation. The illustration below shows the schema in with the referential integrity constraints displayed in this manner:



Relational database schema of COMPANY database

From the above diagram, we can conclude that the `D_No` attribute in the DEPT_LOCATIONS table refers to the `D_NO` in the DEPARTMENTS table so we again draw an arrow to signify referential integrity constraint.

Also the `Manager_Ssn` attribute from the DEPENDENT table refers to the `Ssn` in the EMPLOYEE table. Since the manager is also an employee, the `Manager_Ssn` is derived from employee `Ssn`.

Similarly, the `Essn` attribute from the DEPENDENT table is a foreign key that refers to the `Ssn` in the EMPLOYEE table. If we need information regarding the parent of child (dependent) then we can use the `Essn` foreign key to retrieve that information from the EMPLOYEE table.

Furthermore, notice that a foreign key can refer to its own relation. For example, the attribute `Super_Ssn` in EMPLOYEE refers to the supervisor of an employee; this is another employee, represented by a tuple in the EMPLOYEE relation. Hence, `Super_Ssn` is a foreign key that references the EMPLOYEE relation itself.

In the next lesson, we will discuss the different operations that can be carried out on relational databases.