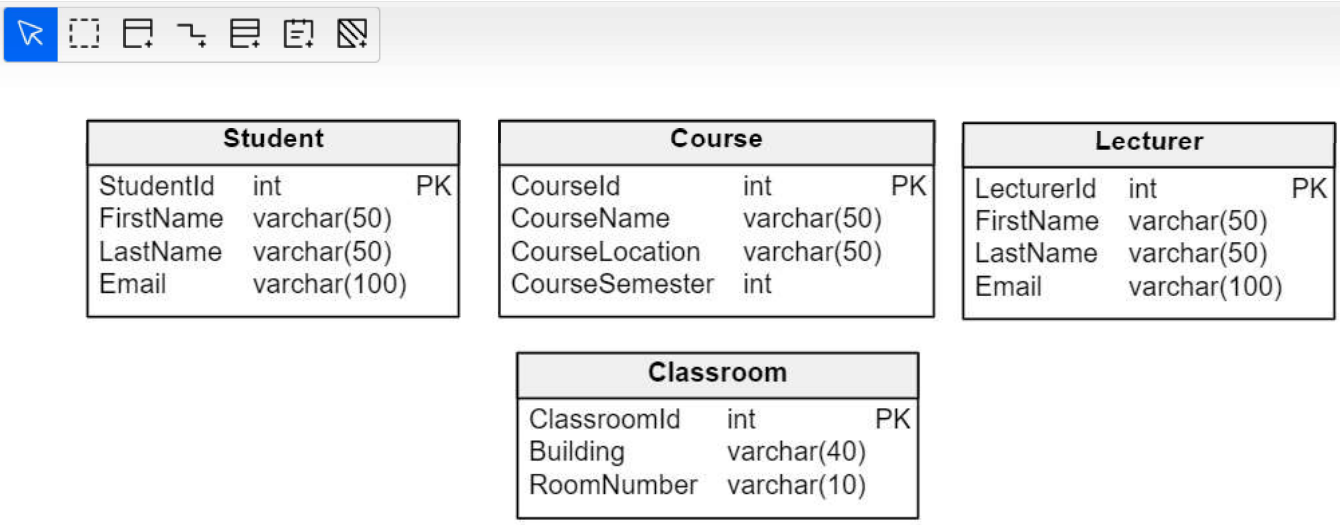


You can learn more ABOUT THE DIFFERENCE BETWEEN ENTITIES AND ATTRIBUTES elsewhere in our blog.

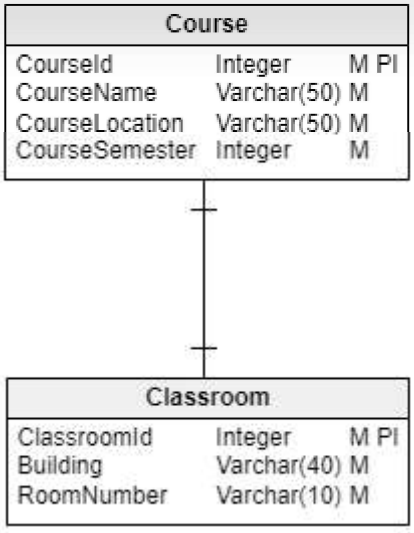
ER diagrams also show the relationships between entities, but these are a bit more complex than entities and their attributes.

Relationships in ER Diagrams

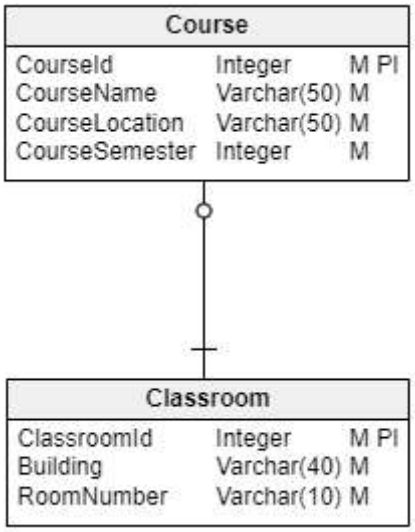
A **relationship** is a link between two entities. It defines how these entities are related to each other. Each relationship has a cardinality that indicates how many objects of one entity relate to how many objects in the other entity. Examples of cardinality include one-to-one, one-to-many, and many-to-many. Relationships between entities can also include inheritance or associations with their cardinalities.

Types of Cardinality

- One-to-one (one entity object relates to one or zero objects):



2. One of the entities is optional: Each course has one classroom, but each classroom may or may not have a course.



- One-to-many (one entity object relates to one, many, or zero objects):
  - Both entities are mandatory: Each course has one lecturer, and each lecturer can have one or more courses.

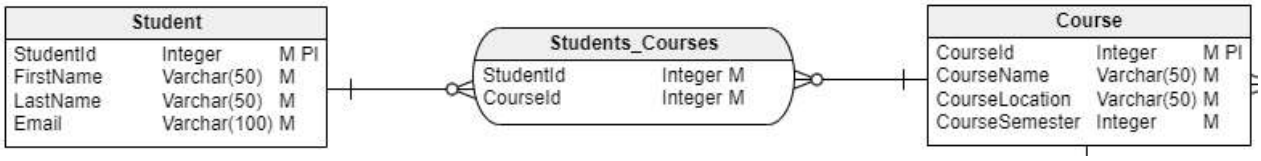
  - One of the entities is optional: Each course has one lecturer, but each

- Many-to-many (multiple entity objects relate to zero, one, or many objects):
  - Both entities are mandatory: Each course has one or more students, and each student attends one or more courses.

  - One of the entities is optional: Each course has one or more students, but each student can attend zero or more courses.

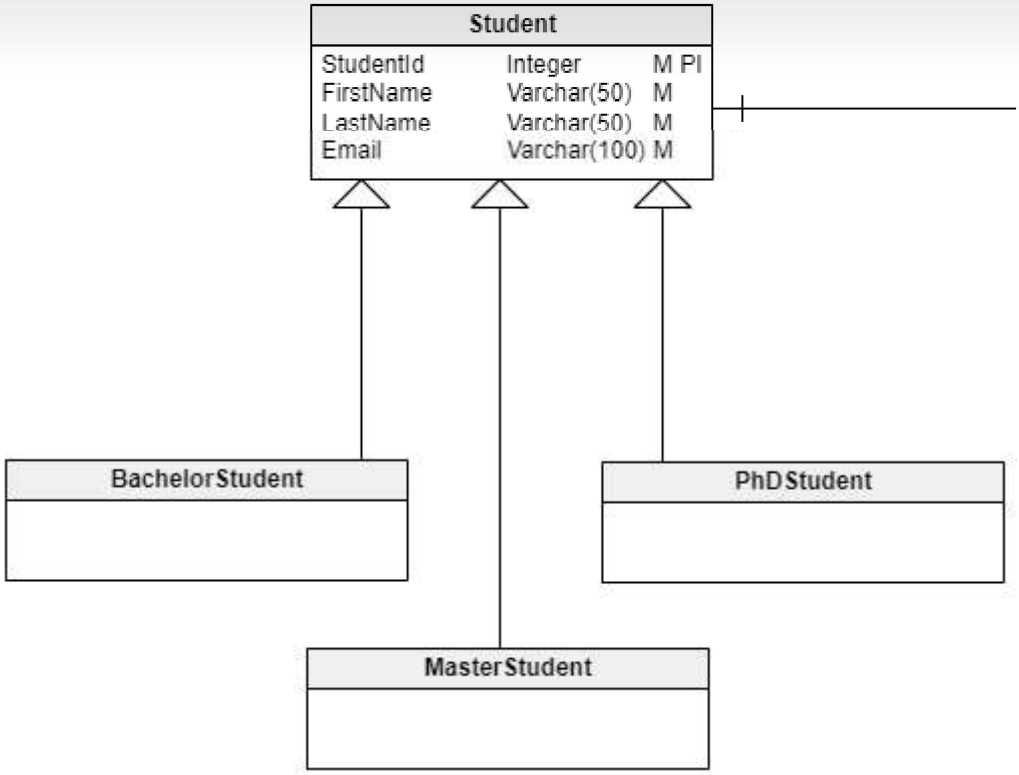
Types of Relationships

- Associations:



The association relationship assigns a row from one table to a row from another table, including the cardinalities. Here, each student can be signed up for zero or more courses and each course can have zero or more students.

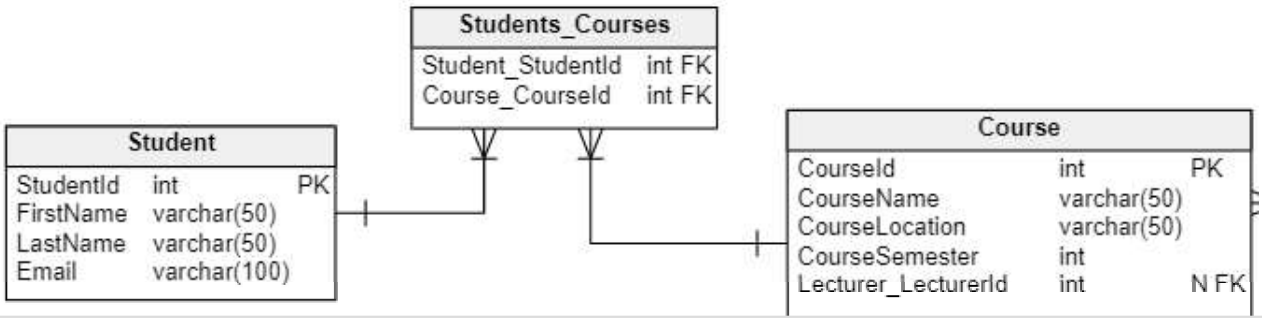
- Inheritance:

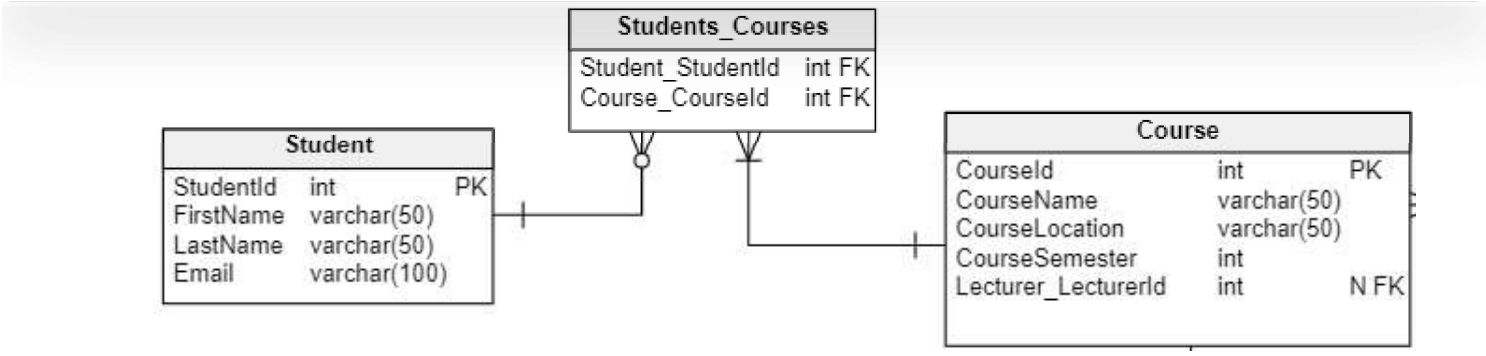


The inheritance relationship creates a *parent-child* relationship between tables. This is where attributes of a *parent* table are inherited by a *child* table (in addition to the child table's own attributes). Here, the `Student` table is a parent table to the `BachelorStudent`, `MasterStudent`, and `PhDStudent` tables.

Physical models are similar, except that the many-to-many relationship requires an additional table to store the relationship data.

- Both entities are mandatory: Each course has one or more students, and each student attends one or more courses.





ER Diagram Notations

Vertabelo uses various notations to ease the process of diagram creation and enhance the readability of diagrams. All notations use symbols to represent ERD entities and relationships – e.g. boxes to represent entities and lines to represent relationships. But the details are different for each one, so let's go through each notation.

Crow's Foot Notation

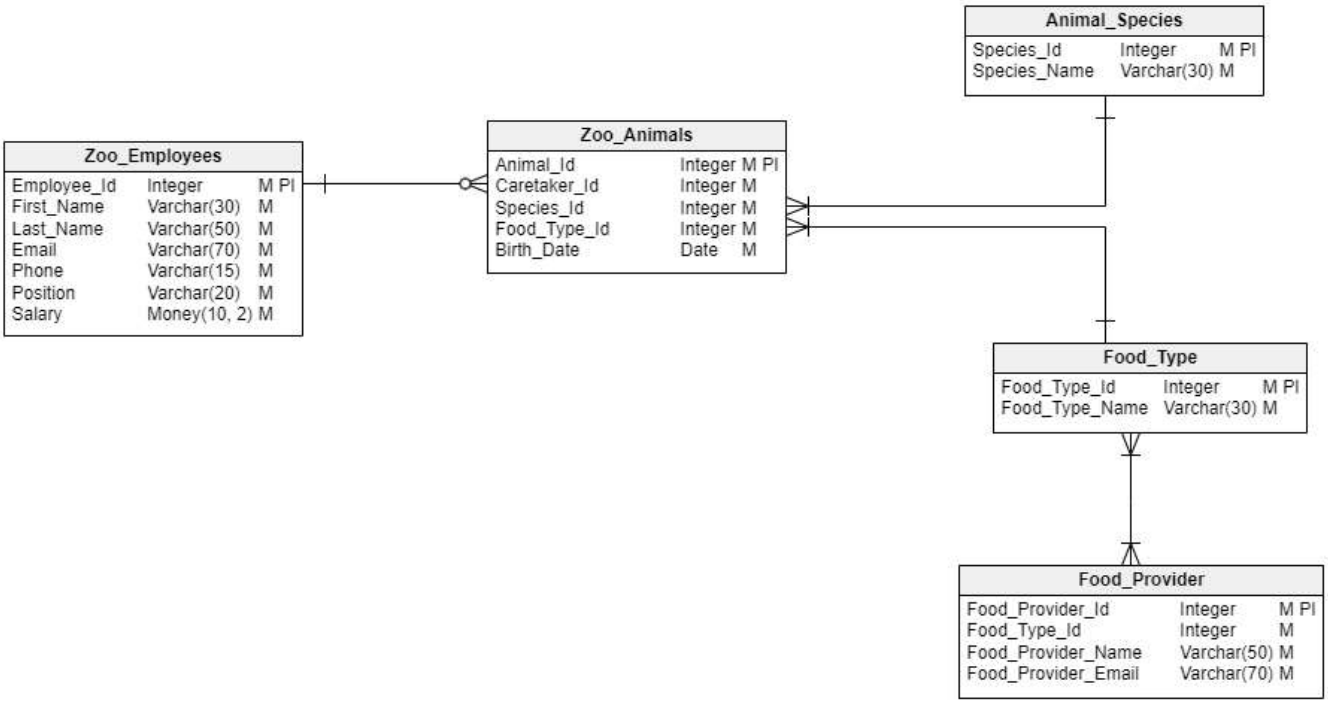
In addition to laying out entities and defining relationships, Crow's Foot notation allows you to define the multiplicities (or cardinalities) of the relationships between the entities.

Here are the available multiplicities:

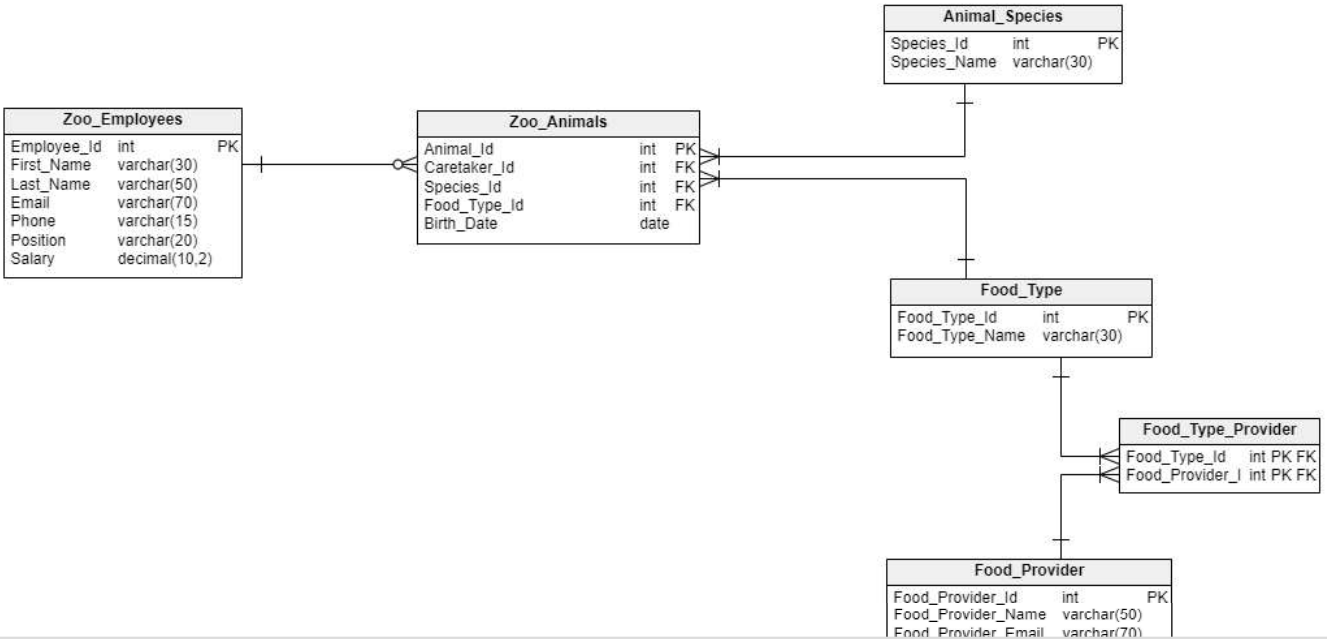
- A *zero-or-one* multiplicity defines an optional relationship.
- A *one-and-only-one* multiplicity defines a singular relationship.
- A *zero-or-many* multiplicity defines an optional relationship.
- A *one-or-many* multiplicity defines a singular or plural relationship.

Examples in Vertabelo

Here is an example of a logical ER diagram in Crow's Foot notation:



Here is an example of a physical ER diagram in Crow's Foot notation:



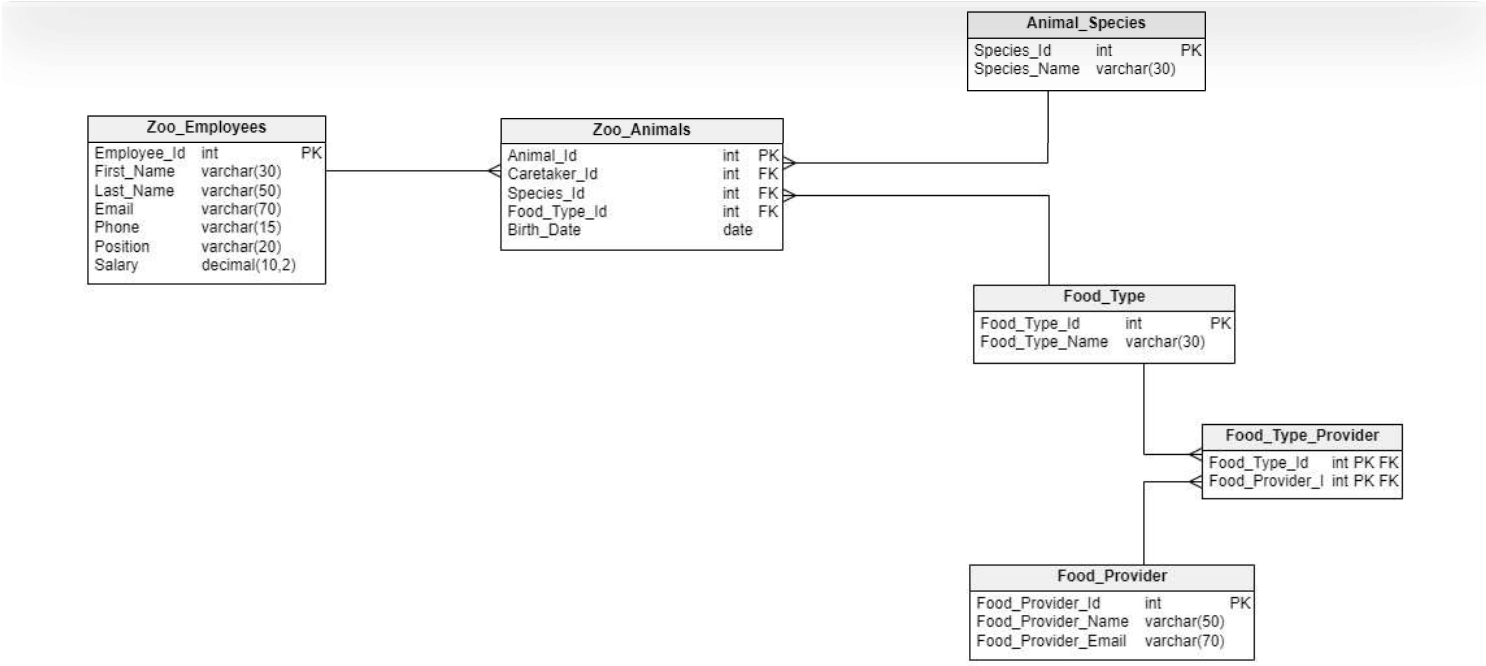
Barker's notation defines the implementation details for entities, attributes, and relationships. Let's look at the details of this notation scheme.

- Entities are represented as rectangular boxes with the entity name at the top. And attributes are grouped according to their roles, such as a unique identifier (#), mandatory field (*), and optional field (O).
- The relationships between entities are defined as mandatory (represented by a straight line) or optional (represented by a dashed line).
- As in Crow's Foot notation, Barker's notation defines multiplicities. However, Barker's notation offers just three types of relationships (as opposed to Crow's Foot notation, where you can mix and match four different multiplicity types). These three types are *one-to-one* (represented by a straight line), *one-to-many* (represented by a straight line with a triangle at one end), and *many-to-many* (represented by a straight line with triangles at both ends).
- Barker's notation offers a way to define primary and foreign key relationships. When the primary key of one table is a foreign key in another table, it is represented by the UID bar. It is defined somewhat like the *one-or-many* relationship in Crow's Foot.
- This notation also uses non-transferability of relationships to mark permanent relationships between entities/tables. It is represented by a rhombus drawn at the end of the connecting line.
- Inheritance is implemented as When a certain entity has several subcategories, you can use subtypes to define them all within the parent entity.

Here are **MORE DETAILS ABOUT BARKER'S NOTATION**.

Examples in Vertabelo

Here is an example of a physical ER diagram in Barker's notation:



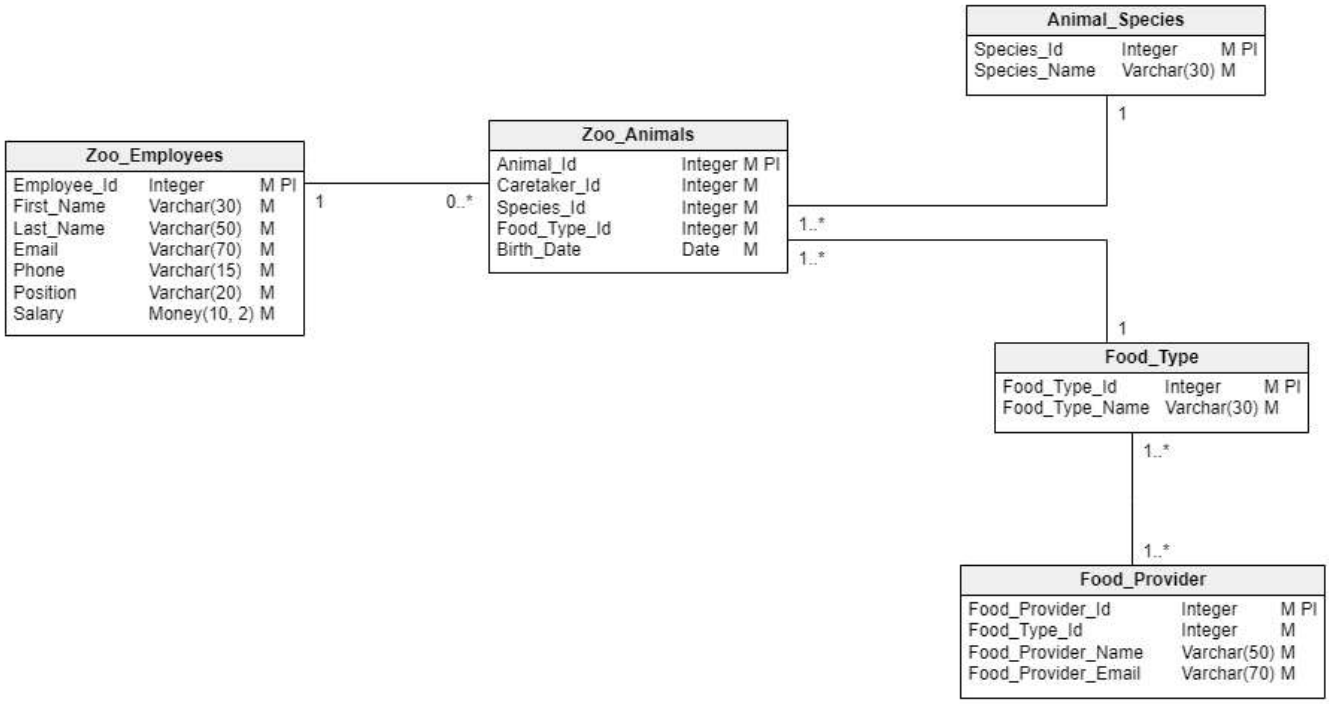
Unified Modeling Language

UML is a popular modeling language used throughout the computer science world. Let's see how it defines diagram symbols:

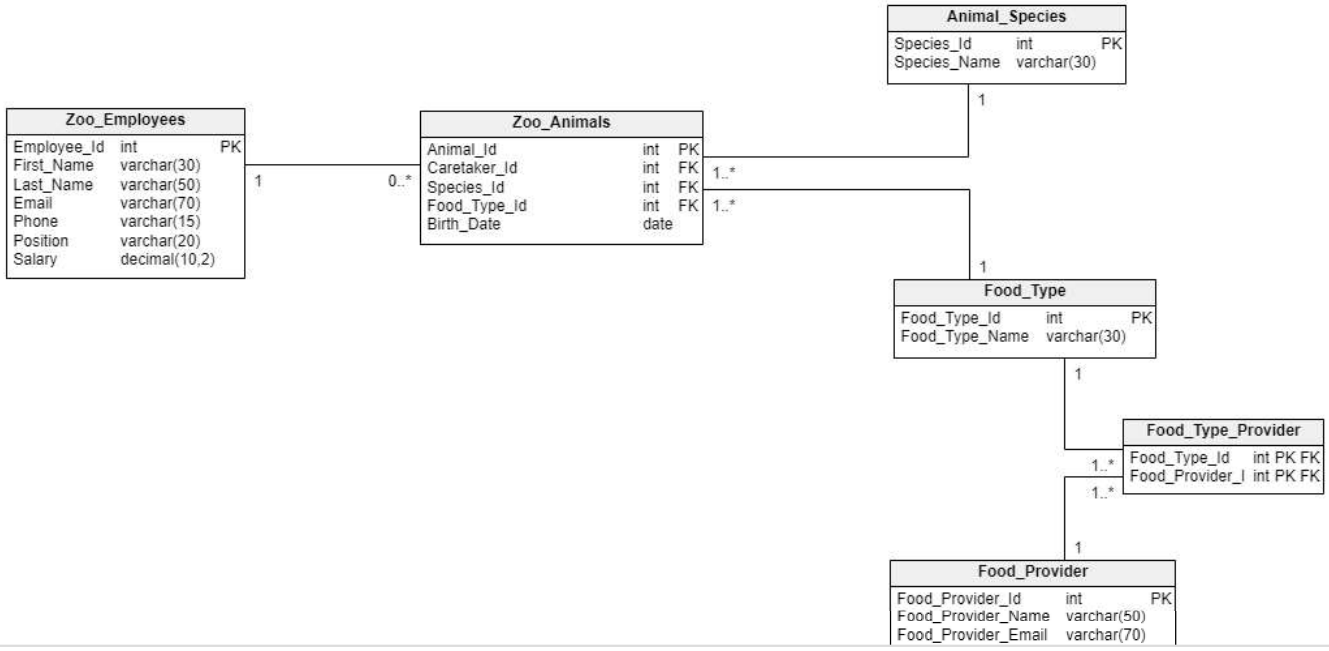
- An entity is a rectangle, as before.
- Relationship multiplicities are defined by writing a number range at each end of the line connecting the entities.
- Inheritance is defined using an empty arrow sign that goes from *child entities* to a *parent entity*.
- Aggregation defines the relationship between a component and the whole to which it belongs. (In this case, a component may exist on its own.) The symbol used is an empty diamond.
- Composition defines the relationship between a component and the whole to which it belongs. Here a component cannot exist on its own. The symbol used is a black diamond.

Examples in Vertabelo

Here is an example of a logical ER diagram in UML notation:



Here is an example of a physical ER diagram in UML notation:



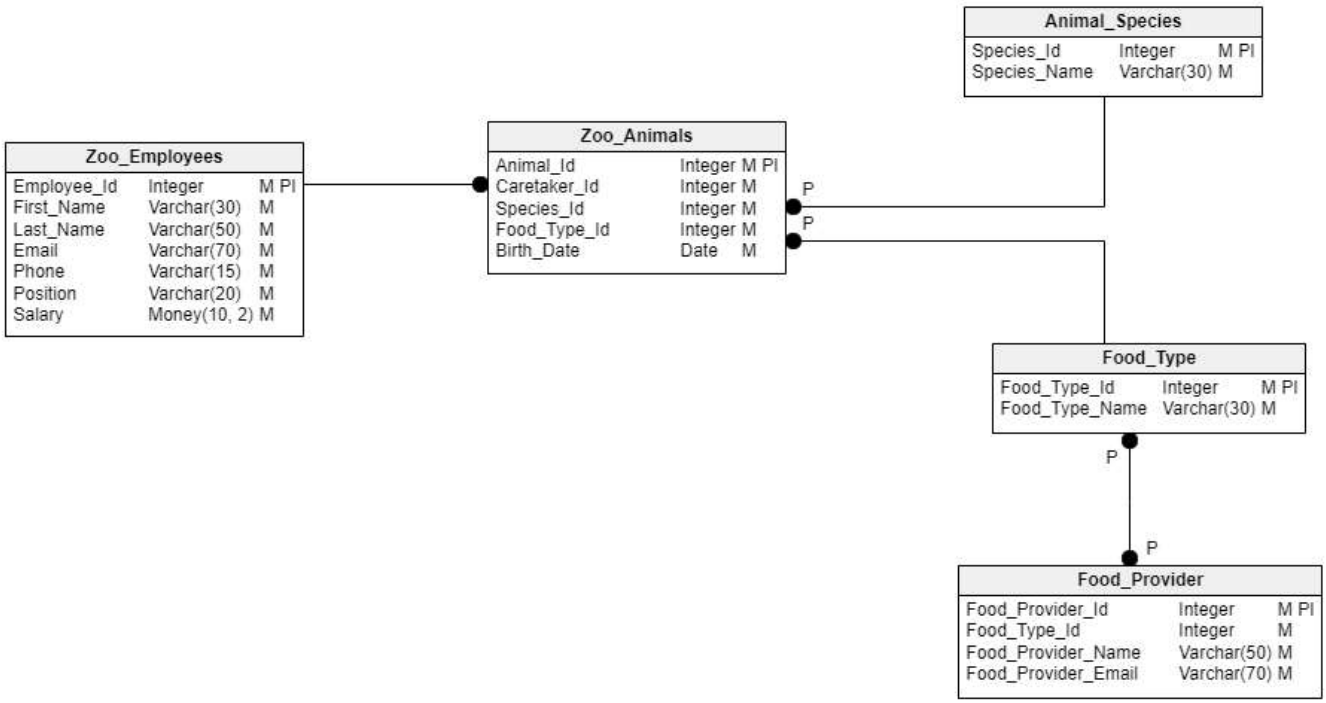
The Integration DEFinition for Information Modeling (IDEF1X) notation defines entities, attributes, and relationships. Let's discuss the details.

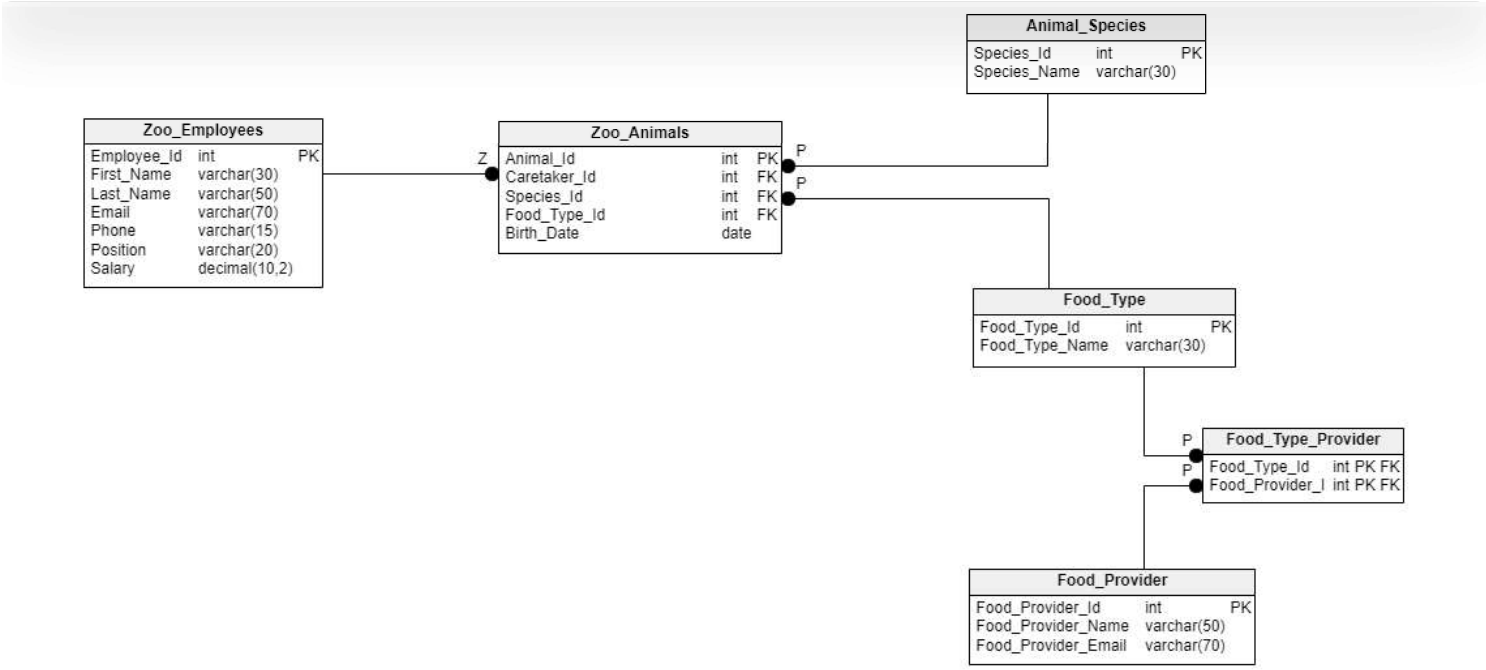
- There are two entity types: an independent entity and a dependent entity. An independent entity exists on its own, while a dependent entity cannot exist without the entities related to it.
- The attribute space is divided by a line into two parts: primary attributes and the remaining attributes.
- Relationships are defined by a straight line with a black circle at one end. Multipicities are defined using symbols as well as a number range.

You can read more about the **DETAILS ABOUT THE IDEF1X NOTATION SYMBOLS HERE**.

Examples in Vertabelo

Here is an example of a logical ER diagram in IDEF1X notation:





Each notation offers a distinct yet similar way to define relationships. To learn more about it, follow our article on the **THEORY AND PRACTICE OF DATABASE CARDINALITIES**.

Vertabelo lets you use different notations in creating your diagrams. For conceptual or logical models, we can use Crow's Foot, UML, or IDEF1X. And for physical models, we can use Crow's Foot, Barker's, UML, or IDEF1X. Here's **HOW TO CHANGE THE DIAGRAM NOTATION IN VERTABELO**.

Ready to Create an ER Diagram?

That's all you should know before creating ER diagrams in Vertabelo.

Vertabelo lets you build conceptual, logical, and physical ER diagrams using different notations that can be changed at any time. You can choose the one that suits your requirements.

Go ahead and try it out for yourself!