United Helpers is a nonprofit organization that provides aid to people after natural disasters. Based on the following brief description of operations, create an appropriate fully labeled Crow’s Foot ERD representing a set of relations that are each normalized to 3NF.

- Individuals volunteer their time to carry out the tasks of the organization. For each volunteer, their ID, name, address, and telephone number are tracked, and they are assigned one task at a time. United Helpers keeps track of both the ID and description of each instance of a task that the volunteer is assigned to, as well as when the volunteer started and finished their work on it.

- United Helpers also has a separate list of their different Chapters, for which they record the Chapter ID, name, and location. Each volunteer is assigned to exactly one Chapter and each Chapter has one or more volunteers participating in its activities.
Additional example of database normalization

There are no transitive dependencies or non-key attributes that are determinants for key attributes, so each relation (and the entire set of relations together) is in 3NF and BCNF also.

The resulting new ER diagram is as follows:

Each table is first drawn with the attributes that resulted from the normalization. It is typical to rename the attributes so that it is clear which relation they belong to - this helps, in particular, to differentiate between different types of "ID" values. Notice that foreign keys are italicized in the diagram and primary keys are underlined. We then need to establish the appropriate relationships between the tables.
Additional example of database normalization

– they should correspond to the rules for converting from an EER diagram to a set of relations, so a foreign key always goes over to the “Many” side of a 1:M relationship, and we can determine the cardinalities by revisiting the problem description and making assumptions if necessary:

- Chapter (1) to Volunteer (M): We know from the original ER diagram that Chapter and Volunteer have a relationship. Each Volunteer is assigned to exactly one Chapter (min 1, max 1), and each Chapter has one or more Volunteers (min 1, max many)
- Volunteer (1) to Assignment (M): Because the Volunteer ID is part of the primary key for Assignment, there must be exactly one Volunteer for each Assignment (min 1, max 1) (i.e., each row in the Assignment relation corresponds to exactly one row in the Volunteer relation). Because Volunteers are assigned one task at a time, it is reasonable to assume that a Volunteer can take on multiple tasks at different times – one might also assume that a Volunteer could sign up to be a volunteer before actually being assigned to a task (min 0, max many).
- Task_Timing (1) to Assignment (M): Because the Task ID and the start time are both part of the primary key for Assignment, there must be exactly one Task for each Assignment (min 1, max 1). Since the same combination of task_ID and tt_start could be associated with different volunteers, however, this implies that more than one person can be assigned to the same task (everyone will begin working on the task at the same time, and the task will be completed when the group completes it). The existence of the Task_Timing relation indicates that you don’t need to know who is assigned to a task to know when that task is finished, so this implies that a Task can exist independent of whether it is assigned to one or more volunteers (min 0, max many).
- Task (1) to Task_Timing (M): Each Task_Timing can only be associated with exactly one Task (min 1, max 1). We are assuming that each Task can be completed more than once at different times, and it’s reasonable to also assume that a Task can be defined before it is actually scheduled (min 0, max many)

Each piece of information given in blue is an assumption that could be stated and then checked with the client to verify that it is accurate.