The following discussion is useful:

* Each COMPANY may list one or more OPENINGs. Because we will maintain COMPANY data even if a company has not (yet!) hired any of TEC's candidates, OPENING is an optional entity in the COMPANY lists OPENING relationship.
* OPENING is existence-dependent on COMPANY, because there cannot be an opening unless a company lists it. If you decide to use the COMPANY primary key as a component of the OPENING's primary key, you have satisfied the conditions that will allow you to classify OPENING as a weak entity and the relationship between COMPANY and OPENING will be strong or identifying. In other words, the OPENING entity is weak if its PK is the combination of OPENING\_NUM and COMP\_CODE. (The COMP\_CODE remains the FK in the OPENING entity.)

Note that there is a 1:M relationship between COMPANY and OPENING, because a company can list multiple job openings. The next table segment shows that the WEST Company has two available job openings and the EAST Company has one available job opening. Naturally, the actual table would have additional attributes in it – but we’re merely illustrating the behavior of the PK components here.

|  |  |
| --- | --- |
| **COMP\_CODE** | **OPENING\_NUM** |
| West | 1 |
| West | 2 |
| East | 1 |

However, if the OPENING’s PK is defined to be a single OPENING attribute such as a *unique* OPENING\_NUM, OPENING is no longer a weak entity. If you use Microsoft Access to implement this design, OPENING\_NUM may be declared as an *autonumber*. Note that this decision causes the relationship between COMPANY and OPENING to be weak. (The relationship line is dashed.) In this case, the COMP\_CODE attribute would continue to be the FK pointing to the COMPANY table, but it would no longer be a part of the OPENING entity PK. The next table segment shows what such an arrangement would look like:

|  |  |
| --- | --- |
| **OPENING\_NUM** | **COMP\_CODE** |
| 10025 | West |
| 10026 | West |
| 10027 | East |

* Similarly, the relationship between PLACEMENT and OPENING may be defined as strong or weak. We have used a weak relationship between OPENING and PLACEMENT.
* A job candidate may have had many jobs - remember that TEC is a *temp* employer. Therefore, a candidate may have many entries in HISTORY. But keep in mind that a candidate may just have completed job training and, therefore, may not have had job experience (i.e., no job history) yet. In short, HISTORY is optional to CANDIDATE.
* To enable TEC or its clients to trace the *entire* employment record of any candidate, it is reasonable to expect that the HISTORY entity also records the job(s) held by the candidate *before* that candidate was placed by TEC. Only the portion of the job history created through TEC placement is reflected in the PLACEMENT entity. Therefore, PLACEMENT is optional to HISTORY.
* The semantics of the problem seem to suggest that the HISTORY is an entity that exists in a 1:1 relationship with PLACEMENT. After all, each placement generates one (and only one) entry in the candidate’s history.
* Because each placement must generate an entry in the HISTORY entity, one would reasonably conclude that HISTORY is mandatory to PLACEMENT. Note that PLACEMENT is redundant, because a job placement obviously creates a job history entry. However, such a redundancy can be justified on the basis that PLACEMENT may be used to track job placement details that are of interest to TEC management.
* HISTORY is clearly existence-dependent on CANDIDATE; it is not possible to make an entry in HISTORY without having a CANDIDATE to generate that history. Given this scenario, the CANDIDATE entity’s primary key may be used as one of the components of the HISTORY entity's primary key, thus making HISTORY a weak entity.
* Each CANDIDATE may have earned one or more QUALIFICATIONs. Although a company may list a qualification, there may not be a matching candidate because it is possible that none of the candidates have this qualification. For instance, it is possible that none of the available candidates is a Pascal programmer. Therefore, CANDIDATE is optional to QUALIFICATION. However, many candidates may have a given qualification. For example, many candidates may be C++ programmers. Keep in mind that each qualification may be matched to many job candidates, so the relationship between CANDIDATE and QUALIFICATION is M:N. This relationship must be decomposed into two 1:M relationships with the help of a composite entity we will name EDUCATION. The EDUCATION entity will contain the qualification code, the candidate identification, the date on which the candidate earned the qualification, and so on. A few sample data entries might look like this:

|  |  |  |
| --- | --- | --- |
| **QUAL\_CODE** | **CAND\_NUM** | **EDUC\_DATE** |
| PRG-VB | 4358 | 12-Dec-00 |
| PRG-C++ | 4358 | 05-Mar-03 |
| DBA-ORA | 4358 | 23-Nov-01 |
| DBA-DB2 | 2113 | 02-Jun-85 |
| DBA-ORA | 2113 | 26-Jan-02 |

Note that the preceding table contents illustrate that candidate 4358 has three listed qualifications, while candidate 2113 has two listed qualifications. Note also that the qualification code DBA-ORA occurred more than once. Clearly, the PK must be a combination of QUAL\_CODE and CAND\_NUM, thus making the relationships between QUALIFICATION and EDUCATION and between EDUCATION and CANDIDATE strong. In this example, the EDUCATION entity is both weak and composite.

|  |
| --- |
| **NOTE for ERD**  **Follow this simple rule: Never declare a FK in an entity by creating it yourself. Let Visio write all the FKs as you establish the relationships.** |

* Each job OPENING requires one QUALIFICATION, and any given qualification may fit many openings, thus producing a 1:M relationship between QUALIFICATION and OPENING. For example, a job opening for a C++ programmer requires an applicant to have the C++ programming qualification, but there may be many job openings for C++ programmers! However, a qualification does *not* require an opening. (After all, if there is no listing with a C++ requirement, a candidate who has the C++ qualification does not match the listing!) Therefore, OPENING is optional to QUALIFICATION.

We decided to define the OPENING entity’s PK to be OPENING\_NUM. This decision produces a non-identifying (weak) relationship between OPENING and QUALIFICATION. However, if you want to ensure that there cannot be a listed opening unless it also lists the required qualification for that opening, the OPENING is existence-dependent on QUALIFICATION. If you then decide to let the OPENING entity inherit QUAL\_CODE from QUALIFICATION as part of its PK, OPENING is properly classified as a weak entity to QUALIFICATION.

* One or more candidates may fill a listed job opening. Also, keep in mind that, during some period of time, a candidate may fill *many* openings. (TEC supplies *temporaries*, remember?) Therefore, the relationship between OPENING and CANDIDATE is M:N. We will decompose this M:N relationship into two 1:M relationships, using the composite entity named PLACEMENT as the bridge between CANDIDATE and OPENING.
* Because a candidate is not necessarily placed, PLACEMENT is optional to CANDIDATE. Similarly, since an opening may be listed even when there is no available candidate, PLACEMENT is optional to OPENING.