Solutions for the Module 9 Exercise Problems

1. For the following problem, define an ERD for the initial requirements and then revise the ERD for the new requirements. Your solution should have an initial ERD, a revised ERD, and a list of design decisions for each ERD. In performing your analysis, you may want to follow the approach presented in module 8.

   The database supports the placement office of a leading graduate school of business. The primary purpose of the database is to schedule interviews and facilitate searches by students and companies. Consider the following requirements in your initial ERD:

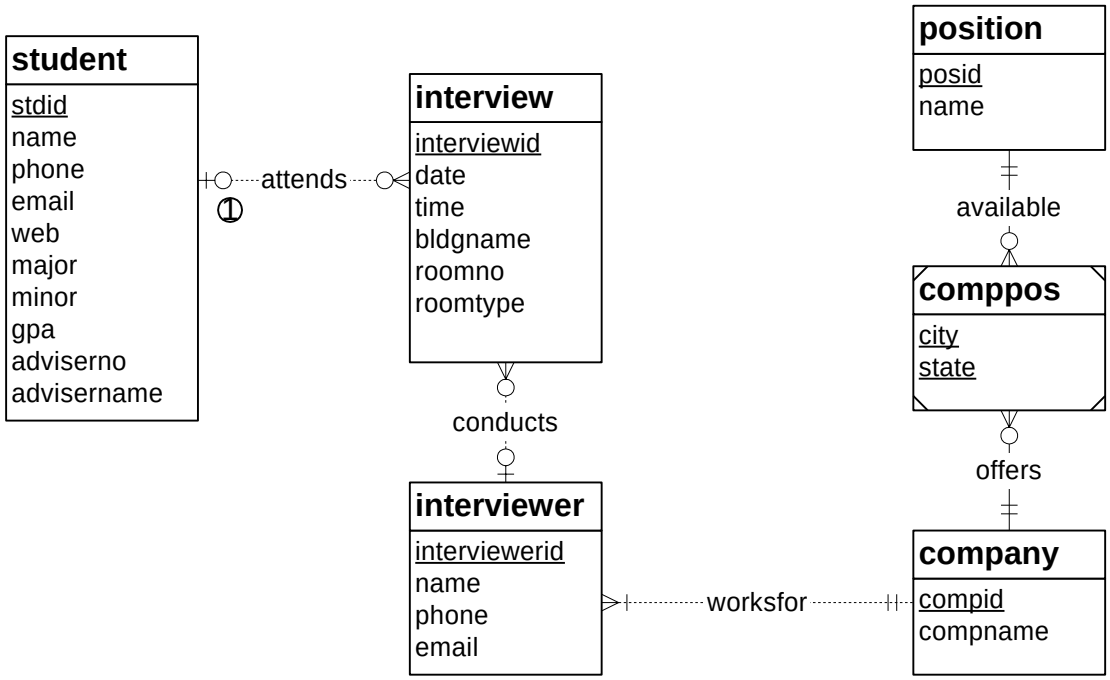
   - Student data include a unique student identifier, a name, a phone number, an e-mail address, a web address, a major, a minor, and a GPA.
   - The placement office maintains a standard list of positions based on the Labor Department's list of occupations. Position data include a unique position identifier and a position description. Because the position list is standardized, multiple companies may interview for the same position.
   - Company data include a unique company identifier, a company name, and a list of positions and interviewers. Each company must map its positions into the position list maintained by the placement office. For each available position, the company lists the cities in which positions are available.
   - Interviewer data include a unique interviewer identifier, a name, a phone, an e-mail address, and a web address. Each interviewer works for one company.
   - An interview includes a unique interview identifier, a date, a time, a location (building and room), an interviewer, and a student. An interviewer may conduct multiple interviews.

   After reviewing your initial design, the placement office decides to revise the requirements. Make a separate ERD to show your refinements. Refine your original ERD to support the following new requirements:

   - Allow companies to use their own language to describe positions. The placement office will not maintain a list of standard positions.
   - Allow companies to indicate availability dates and number of openings for positions.

- Allow companies to reserve blocks of interview time. The interview blocks will not specify times for individual interviews. Rather a company will request a block of X hours during a specified week. Companies reserve interview blocks before the placement office schedules individual interviews. Thus, the placement office needs to store interviews as well as interview blocks.
- Allow students to submit bids for interview blocks. Students receive a set amount of bid dollars that they can allocate among bids. The bid mechanism is a pseudo-market approach to allocating interviews, a scarce resource. A bid contains a unique bid identifier, a bid amount, and a company. A student can submit many bids and an interview block can receive many bids.

Solution



1. An interview may be initially setup without knowing the student or interviewer.

The Student, Interview, Interviewer, Company, and Position entity types follow directly from the narrative. The narrative statement clearly indicates attributes and a primary key for each entity type. The M side of the 1-M relationships, Attends, Conducts, and WorksFor, follows from the narrative with students attending multiple interviews, interviewers conducting multiple

interviews, and companies having a list of interviewers. The one side of the 1-M relationships follows from an interview having an interviewer and student, as well as an interviewer working for one company. The minimum cardinalities are not specified in the narrative so additional requirements collection is necessary.

Note that there is no entity type for the placement office. The placement office is the client for the database. If the problem indicated that the database should support multiple placement offices, a placement entity type may be useful.

The difficult part of the narrative involves the relationship between company and positions. The problem narrative indicates that multiple companies may interview for the same position and a company may interview for a list of positions so an M-N relationship is necessary.

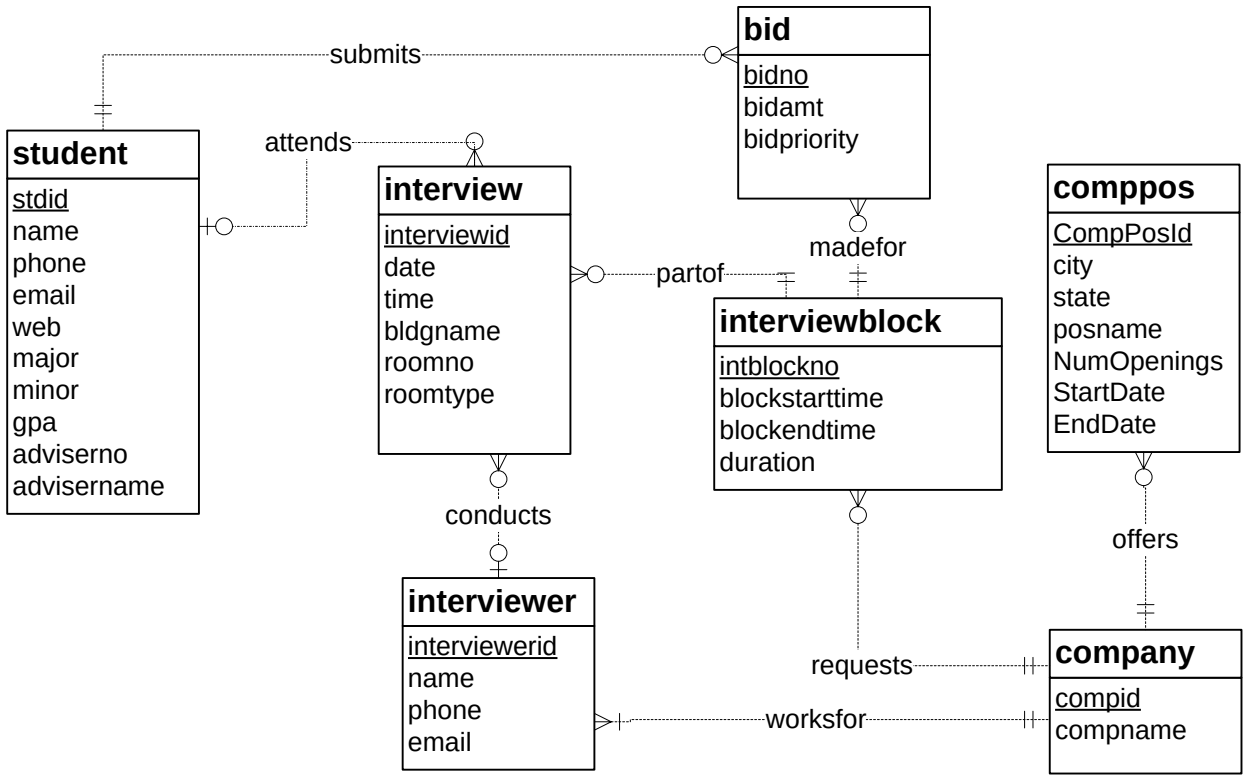
The most difficult part of the narrative indicates that a company lists the cities in which positons are available. The solution ERD uses an associative entity type (CompPos) in which the primary key is a combination of CompId, PosId, City, and State. City by itself may not be unique. To incorporate companies with international employment opportunities, country should be added.

If a Location entity type is used, a more complex solution is possible. An associative entity type with three 1-M identifying relationships among Location, Position, and Company would work. The Location entity type would have attributes for city, state, and possibly country. More requirements collection is needed to determine the most appropriate solution. As a default, the CompPos entity type in the solution ERD is preferred because it is simpler.

Transformations: see ERD below

- Remove the *position* entity type so that positions are not standardized across companies. Add attributes to the *comppos* entity type for the position name, start date, end date, and number of openings. To reduce the number of attributes in the combined primary key of comppos, a unique identifier has been added. In this case, the *compid*, *city*, *state, and startdate* attributes are a combined candidate key.

- Add an entity type (*interviewblock*) and relationships between *interviewblock* and *interview* and between *interviewblock* and *company*. The new entity type and relationships allow companies to schedule blocks of time. An interview block will contain a collection of interviews with the same company.
- Add an entity type for bids. The *bid* entity type should be connected to *student* and *interviewblock*.



The Position entity has been eliminated because the problem indicates that a standardized position list is not maintained. A transformation to contract the entity type structure has been made so that CompPos only has one identifying relationship.

The problem definition indicates the need for the Bid and InterviewBlock entity types. For interview blocks, the attributes must be discerned from the problem statement. To represent blocks of time, the InterviewBlock entity type has attributes for start time, end time, and duration for each interview. The primary key (IntBlockNo) is not specified in the problem statement so additional requirements collection is needed.

The narrative provides incomplete details about some new relationships. The narrative indicates that a student can submit many bids and a bid contains one student. However, the minimum cardinality specifications are omitted. For interview blocks, the narrative indicates that companies request interview blocks before related interviews are added. A bid is made for a slot in an interview block with a company. The minimum cardinalities in the MadeFor, Requests, and PartOf relationships are not specified in the problem statement.

The problem statement needs another revision about the bidding process. The current design does not store details about connecting winning bids to interviews. More requirements collection is necessary.

2. For the Expense Report ERD shown in Figure 1, identify and resolve errors and note incompleteness in the specifications. Your solution should include a list of errors and a revised ERD. For each error, identify the type of error (diagram or design) and the specific error within each error type. Note that the ERD may have both diagram and design errors. If you are using the ER Assistant, you can use the Check Diagram feature after checking the diagram rules yourself. Specifications for the ERD appear below:
    - The Expense Reporting database tracks expense reports and expense report items along with users, expense categories, status codes, and limits on expense category spending.
    - For each user, the database records the unique user number, the first name, the last name, the phone number, the e-mail address, the spending limit, the organizational relationships among users, and the expense categories (at least one) available to the user. A user can manage other users but have at most one

manager. For each expense category available to a user, there is a limit amount.

- For each expense category, the database records the unique category number, the category description, the spending limit, and the users permitted to use the expense category. When an expense category is initially created, there may not be related users.
- For each status code, the database records the unique status number, the status description, and the expense reports using the status code.
- For each expense report, the database records the unique expense report number, the description, the submitted date, the status date, the status code (required), the user number (required), and the related expense items.
- For each expense item, the database records the unique item number, the description, the expense date, the amount, the expense category (required), and the expense report number (required).

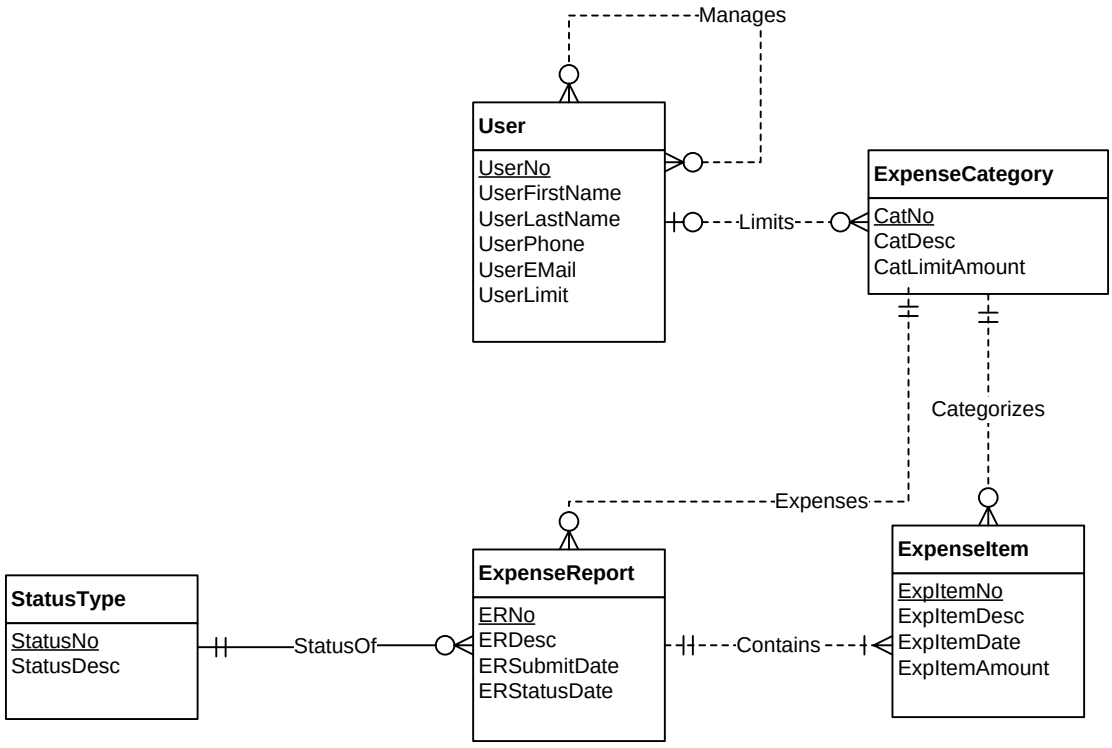


Figure 1: ERD for the Expense Reporting Database

Solution

The ERD below fixes the errors in the original ERD. The following list describes the problem resolutions.

- The Manages relationship should be 1-M, not M-N (cardinality design error).
- The Limits relationship should be M-N, not 1-M (cardinality design error).
- The minimum cardinality in the Limits relationships for Users should be 1, not 0 so that a user has at least one expense category.
- The Limits relationship should have an attribute for the amount.
- The minimum cardinality of the Limits relationship for a user should be 1, not 0 (cardinality design error).
- *ExpenseItem* should not be a weak entity type (weak entity type diagram error).
- The Submits relationship was added from *User* to *ExpenseReport*. Without this relationship, the user for an expense report cannot be determined (missing relationship design error).
- The *StatusOf* relationship should not be identifying (identifying relationship diagram error).

- The *Expenses* relationship from *ExpenseCategory* to *ExpenseReport* should be removed because it is inconsistent with the specification and redundant if corrected (cardinality and redundant relationship design errors). The specification indicates that an expense report is related to many expense items and that each expense item is related to one expense category. Thus, an expense report can be related to many expense categories. Revising the Expenses relationship to M-N makes the relationship redundant with the Contains and Categorizes relationships.
- The specification is incomplete about attribute constraints such as null value allowed and data types.