

Q &A on Entity Relationship Diagrams

The objective of this lecture is to show you how to construct an Entity Relationship (ER) Diagram. We demonstrate these concepts through an example. To break from monotony of didactic lectures, this session is organized as a dialogue between a student and the instructor.

What is the Point?

Student: "I don't really get this. I mean you have so many new abbreviations and terms. I feel even when I understand what you mean by use cases and entity relationship diagrams, I still do not know how to do it. Can you give me an example?"

Instructor: "Glad you asked. The purpose of an ER diagram (I mean Entity Relationship Diagram) is to help you prepare the structure of the database."

Student: "Yes but what is an ER diagram."

Instructor: "In its simplest form, an ER diagram is a list of entities and their relationship to each other. This begs the question what is an entity. Right?"

Student: "Yes, what is an entity?"

Instructor: "An entity is anything about which we want to keep persistent data. By persistent data I mean facts that we may need about it later in time. The fields of data we keep about an entity is referred to as an attribute of the entity. The actual data is referred to as values of the attribute. So a patient maybe an entity in an Electronic Medical Record, what we abbreviate to EMR. The patient's first name may be an attribute of the entity patient. "George" could be the value of the attribute "first name" of the entity patient. In a database an entity corresponds to a table, an attribute in the entity corresponds to a field, and the value of the attribute is the data."

Student: "I see. Lots of new jargon but nothing earth shattering, right?"

Instructor: "Well the concept that we should organize databases in terms of entities and their relationships is very important. It allows us to create efficient databases that are not full of logically missing data or require repetition of data in each record. ER diagrams make it possible to have large databases and use them efficiently. In this sense, the concept is really important. "

Student: "What do you mean by logically missing data?"

Instructor: "When you organize databases as flat files, you would need data on every field for each record. When you break the data in terms of several tables, you need data only on the attributes within the table. In this fashion you avoid having to force the user of the database to report fields that do not make sense. When users have to do so, they are asked to leave the data items that are not appropriate as blank. This is called logically missing data. For example, in an EMR you may be asked to record a patient's visit. During this visit you may have to collect information about whether the patient has asthma or hypertension. If we have two fields one called asthma and the other hypertension, and if the patient's visit was about asthma, then we are forced to add a blank for hypertension. But the inefficiency does not end there as we are also forced to answer questions such as what was prescribed for hypertension, what date should hypertension be followed up, and so on. We end up having to indicate a large number of fields as blanks."

Student: "I see, that would waste a lot of effort."

Instructor: "Effort is also wasted when we have to repeat data. If you had a flat file design, every time a patient comes for a visit you would need to collect and input all information about the patient (e.g. contact information, insurance information, etc.) This leads to a lot of redundant information which will make the database inefficient."

Student: "Yes, I can see the value of having multiple tables so that we can avoid these inefficiencies. I can also see that the ER diagram could lead to specification of the tables and the fields in the database. Now the real question comes up. How do you construct an ER diagram?"

Recognizing Entities

Instructor: "There are many ways to do so. You can do it from use cases, these are descriptions of the function of the database for different users of it. Lets see if I can walk you through an example. Suppose we want to create an EMR for a clinic, who would use it and how?"

Student: "Well many people but at its core an EMR is used by a provider of care to record information about the patient's visit so that the treatment can be coordinated over time with other providers. But it is also used to bill the patient about treatment they have received."

Instructor: "You just provided me with a statement about use of the EMR database. I can analyze this statement to see who and what are the entities."

Student: "I can see two entities, the patient and the provider. We need information, or in your terminology persistent information, about these two actors. I would organize the database as table for patients and a table for providers."

Instructor: "An entity does not need to be a person. In your statement you also mentioned a visit and treatment; these can be an entities too. We may collect the date of a visit, the diagnosis during the visit and the treatment patient received during the visit. These are facts about the visit. For treatment, we may collect time it takes usually, the cost it has, and the charge we want to make for it."

Student: "Aren't these facts about the patient? I mean why wouldn't diagnosis be a fact about the patient, wouldn't treatment be something that belongs to the patient table?"

Instructor: "No. Diagnosis changes even when the patient has not changed. It cannot be stored in a table of patients. Certainly diagnosis is a characteristic of the patient during a visit but it changes in each visit and therefore it makes sense to organize it in a visit table."

Student: "What about treatment?"

Instructor: "Well you can put this in a separate table too because we probably need to store information about various treatments we provide , their cost, their risks, and so on. In deciding whether to describe something as a separate entity you have to balance how much information you have that describes it. When it comes to treatment, we have several pieces of information that describes treatment independent of the patient, the provider or the visit. For example, risks associated with treatment or cost of treatment. Therefore, we need the entity treatment too. In general each table is about something and you need to make sure that all the facts in the table are about that core concept. We call the core concept of the table a primary key. A table is a collection of facts about the primary key. Furthermore, given a primary key all facts in the table should be unique. "

Primary Key

Student: "Primary key seems to be an important concept for organizing the table."

Instructor: "Every entity can be described in terms of a primary key. If a fact can belong to the primary key and nothing else, then it belongs to the entity. Otherwise it belongs to a different entity. A patient's primary key could be his social security number. If we are willing to store the patient's address in a table with social security as primary key then this attribute belongs to the entity patient. Consider diagnosis. If we were to store diagnosis in the table patient, it would be about the patient but also about the visit. Therefore it does not belong to the table patient."

Student: "You could say that address does not belong to the patient either as it changes too."

Instructor: "Yes, you could. If so, then you would create a different table with patient ID and date as primary keys and store addresses in there. This way both the old and new address can be tracked. In the end it is a choice the designer has regarding how many different entities he/she creates. A working database may have thousands of entities. But for the sake of our example here we will stay with four entities the patient, the provider, the visit and the treatment.

Student: "Ok I understand that deciding about entities is an art but it does involve some do's and don'ts."

Attributes

Instructor: "Now that you have decided about the entities, you can start to list the attributes of each. What information you want to keep about the patient entity?"

Student: "Well lets start with the most important information. I do not want to keep social security as the primary key. I need to find something else. A unique number. Maybe a combination of first name, last name and date of birth."

Instructor: "You can do so. You can decide that a combination of three fields is the primary key for your table patient. You could also leave it up to the database to generate a unique number, often done as consecutive automatically generated number. In this fashion when you add a new patient to the table, the database gives it a unique number. Of course, you have to communicate this number to the patient and the patient has to remember it when they come for a visit. What else besides the primary key do you think belongs in this table?"

Student: "The contact information, demographic data and the address."

Instructor: "Each of these are a collection of fields. If you collect the address as one field, then you would not be able to find all patients that are at a particular zip code. How would you break this information further?"

Student: "I might collect, first name, middle name, last name about the patient. I might collect their race and ethnicity, their date of birth, and their insurance number and company. I might collect their street number, street name, city, state and zip code."

Instructor: "Lets accept these as the attributes for our patient table. Now what would you collect about the provider?"

Student: "The primary key could be the employee number, that is always unique within one employer. The attributes could be employee's first name, middle name, last name, title, graduation year, date of birth, board certification, telephone number, home address (street name, street number, city, state and zip code), work address (street name, street number, city, state and zip code), and current email. These are all facts about the employee/provider."

Instructor: "Great. What would you need to know about treatment?"

Student: "Well we need to have a treatment code, that would be the primary key. Then we would need to have a text describing the treatment, a dollar amount for the cost of treatment, and perhaps a warning statement. If there is a typical medication involved, we might include that here too.

Foreign Keys

Instructor: "Great, now we come to the core table of the entire database: visits. This is a table that will be the largest as there are more visits than patients and there are more visits than providers. But interesting most of the data in this table may depend on items in other tables."

Student: "Yes, that is true. The primary key of this table is probably an auto-number generated by the database. We would need to know which patient was involved in the visit. We can do so by including the patient ID in this table. We would need to know which provider was involved. Here again we need the provider's ID included. We would also need to know the patient's diagnosis during the visit and the treatment the patient received. Well since we have a

table on treatment, all we need to know is the treatment code and we will get the rest of the information from the treatment table."

Instructor: "By including the primary key of other tables into the visits/encounters table you have related this table to other information. These fields are sometimes referred to as foreign key as they link to primary keys of other tables. We are now ready to create the full ER diagram. "

Student: "The ER diagram will show each entity (their attributes) and the relationship between the entities."

Instructor: "Yes, in Access you can do this by creating the tables and then connecting the tables to each other. Under tools, under relationships, you show all tables and then connect them to each other by dragging the primary key of a table to the foreign key in another. Thus from the visits/encounter table we may drag the foreign key patientID to correspond to the primary key ID in the patient table. In this fashion, the database is aware of the relationship between the two tables. In words the relationships says that the patient ID in the visit should correspond to the ID in the patient table."

Referential integrity & Types of relations

Student: "I can see how that creates an ER diagram and how it helps organize the tables in a database. I see also the value of breaking large amount of data in terms of entities. But I have a concern. Keeping track of things with all these numbers and IDs could be time consuming and difficult. What if information in one table is inadvertently deleted. Then we lose the meaning of information in other tables. For example, suppose we delete a patient from the patient table. But we have information about this patient in other tables. All of sudden our database becomes full of data we do not understand and cannot link to a specific patient. How do we avoid this?"

Instructor: "To avoid mismatched data, there is a feature available in most databases which is called referential integrity. This feature forces a cascade of deletion of all related facts when the original fact is deleted. Thus deleting the patient ID will lead to deletion of all visits corresponding to that patient. Obviously thousands of records may be deleted when one patient is deleted. This creates a significant burden to be careful when deleting records from tables, as it will have many other consequences. At the same time, it allows the ease of deleting a record in one place and eliminating all related facts elsewhere. Keep in mind that deleting the patient will delete records of visits which will delete work done by provider. All of a sudden, data on provider's schedule will be distorted. So deletion has a lot of consequences and the best way forward is not to delete data".

Student: "Can deletion occur by mistake?"

Instructor: "Yes and this is quite problematic. When using inner join, if the data in one table is not matched to another table, then the data are deleted. Inner joins require both tables to have the same values in the fields that are matched. No match and then no data. There is another way to go about doing this that reduces the problem with exact matches. There are other types of joins. So far we have talked about joins that must have the same values in both tables. Sometimes you want to join two table where all values of one table are present and only matching values of the other key are present (when there is no match a blank is present). "

Student: "Give me an example of when you want to have this type of joins."

Instructor: "Suppose you want to know if there are providers not taking care of any patients. To answer this you need to be able to go from providers table to visit table and see if there are any providers with no visits. But if we stay with our match of foreign and primary key all cases with no match will be eliminated and we will not be able to see if there is a provider with no patient.

Student: "I see. It is awkward to think of these situations as they involve non-typical uses of the data. But certainly the need may arise for different types of joins and relationships. On a related issue, is there a way of putting words to the relationship between two tables. For example, when we talk of the relationship between two people we have words for it like they are married, or that one is the father of another. Can you put words to relationships?"

Instructor: "Yes. You can. The words of course are implied in the field names. If the foreign key is ID of the son, then it implies that it links the father to the son. Some ER diagrams allow the specification of the relationships in

words. Now that we are talking of types of relationships it is also important to talk about one to one and one to many relationships."

Student: "A one to one relationship requires a record for each item in the other table. For example, a patient's ID and his birth date have a one to one relationship. Each patient has one birthdate."

Instructor: "Yes, a one to many relationship allows one record to have multiple records in another table linked to it. For example, the patient record may have multiple records in the visit table."

Student: "Yes that makes sense."

Junction Table

Instructor: "Sometimes you may want to create many to many relationships among tables. Consider the case of relating patients to their address. Suppose we want to allow a patient to live at two different addresses and two different patients (mother and a child) to live at same address. This creates several problems for us as primary keys need to be unique and in the patient table we cannot point from the same primary key to two different addresses. One way to solve this problem is to introduce a third table in which the relationship between address and patients are kept. These types of tables are called junctions."

Student: "Could you layout the example in more detail?"

Instructor: "Suppose the patient table has first name, last name and the details of the address. First we have to separate the address from the patient table and create a new table with its own auto-number primary key for addresses. Now we need to create a third table that links the patient ID and address ID. This table could also have a third field that describes the type of the relationship."

Patient Address Junction			
ID	Patient ID	Address ID	Relationship
1	234	5	Landlord
2	213	5	Tenant
3	200	4	Lives at

In this junction table, patient 234 is the landlord of address 5. Patient 213 is the tenant of address 5. Patient 200 lives at address 4.

Student: "I see this actually puts words to relationships and also allows us to have multiple people living at same address. We can have primary keys in patient and address table that are unique but have them listed in multiple ways in the junction table. I imagine something like this will also be useful to show relationships among members of the same table. How do you for example show a mother child relationship?"

Instructor: "You are right. Here again we make a junction table with patient IDs. In this case, we want to relate patient IDs to patient IDs. Here for example we have patient ID 234 being mother to patient ID 213; patient ID 213 is mother to patient 215. Patient ID 200 is the father to patient 215. If we need to know who is the parents of patient 215, we can tell from this table. We can also tell who are the grandparents of patient 215. Once the IDs are available we can look up their names and contact information in the patient table."

Patient Patient Junction			
ID	Parent	Child	Relationship
1	234	213	Mother to
2	213	215	Mother to
3	200	215	Father of

Student: "This is a clever way of keeping information."

Instructor: "It may be helpful to summarize what we have covered. We started with some definitions, including terms such as entity, attribute and values. Then we looked at how entities for an electronic medical records can be identified. You then specified the attributes. We translated the entities to tables and attributes to fields in Access and started looking at representing the relationships between these tables using foreign and primary keys. We talked about different types of relationships and referential integrity. We also showed how and junction table can be used to represent many to many relations and relationships between the table and itself.