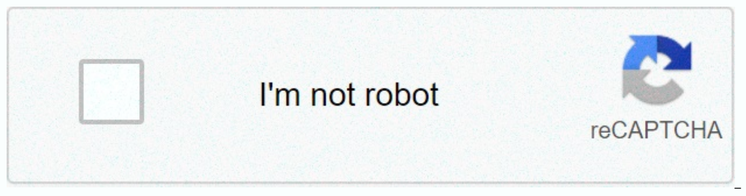

Continue

Maximum cardinality can be optional

Table of Content Goals Introduction Context Entities, Properties and Values Entity-Relationship Modeling Relationship Participation Condition (membership class) Weak and Strong Entities Problems with Entity Ratio (ER) models Conversion entity relationships into relationships Revise questions Goals At the end of this chapter you should be able to: Analyze a given situation to identify the relevant entities. Be able to identify the relationships between entities, and perform any necessary transformations. Further develop the model by identifying properties for each entity. Map the entities in tables suitable for Relational Database implementation. Introduction In parallel with this chapter, you must read Chapter 11 of Thomas Connolly and Carolyn Begg, Database Systems A Practical Approach to Design, Implementation and Management, (5th edn.). This chapter is the first to address the crucial topic of database design in detail. The most important approach described in this chapter is called Entity Relational Modeling. This technique has become a widely used approach in the development of database applications. The approach is essentially top-down, in that the first step is to look overall at the requirements for the application developed, identifying the entities involved. The approach progresses from that point on by developing a detailed model of the entities, their properties and relationships. The Entity Relationship Modelling process is not formal, in the mathematical sense, but to be done well, it requires a consistent precision to be applied to the way entities, their relationships, and their properties are discussed. The approach can be complemented by methods that are more formal in their approach, and it provides a bottom-up perspective on the design process. The most commonly used of these approaches is Normalization, which will be a core topic of the later chapters on database design. Context This chapter introduces the ideas of top-down database design, and provides the starting point for learning how to develop a database application. The chapter now links to the others covering database design (Normalization and other design topics). The chapter also has considerable relevance to the materials in the module on performance voting, such as the chapter on indexing, as the decisions made during database design have a major impact on the performance of the application. Entities, characteristics and values Entities Many organizations (such as businesses, government departments, supermarkets, universities and hospitals) have a number of branches, divisions or divisions to handle a variety of functions or different geographical areas. Each branch, partition or section can be divided into smaller units themselves. It is possible to use each branch, partition or section each unit within this) as an organisation in its own right. Organizations need information to the tasks and activities for which they are responsible. The information these organizations need can be categorized in a number of ways, for example: People Payroll Pensions Annual Leave Things Furniture Equipment Stationery Fire Extinguishers Places Offices Warehouses Stockrooms Events Auction is made Purchase Order is hired Item is hired Invoice is issued Drafts Image of Product Advertising Marketing Research and Development. Each of these can be considered an entity. Important Entity An entity can represent a category of people, things, events, locations or concepts within the area under consideration. An entity instance is a specific example of an entity. John Smith, for example, is an entity instance of an employee entity. Properties Entities have properties. The following are typical of the properties that an entity can possess: Entity: Home Features: Entity: Book Features: Entity: Employee Features: Important Attribute An entity may have one or more properties associated with it. These properties represent certain properties of the entity; for a person can have properties name, age, address, etc. Values using the entities and properties shown above, the following are examples of one set of values for a specific instance of each entity. Each occurrence of an entity will have its own set of values for properties it owns. Entity: Home Attributes: Values: Entity: Book Features: Values: Entity: Employee Features: Values: Primary Key Data Elements If the value of certain properties (or perhaps just one attribute) is known for a particular entity, it allows us to discover the value of other properties associated with that entity. The properties (or attribute) that possess this quality are known as keys, because they can unlock the values of the other properties associated with that particular instance of an entity. Why do we need a key? Suppose we had two staffers with the same (or similar) names, such as Linda Clark and Lydia Clark. It would be a simple mistake to record something in the file of Linda Clark to be kept in the file for Lydia Clark (or vice versa). It would be even harder to tell them apart if the name was given as just an initial and surname. Some names may be spelled slightly differently, but sound similar (like Clark and Clarke), and therefore pose a further risk of identifying the wrong staffer. Adding a staff number as the primary key will enable us to be sure that when we have had to refer to some of these staff members, we have identified the correct individual. In this way, 11057 Clark can be distinguished from 28076 Clark. The following are examples of key data elements: The payroll number (primary key) of a staff member sets us able to find out the name, job title and address for that individual. The account number (primary key) allows us to find out if its balance is overdrawn. The item code (primary key) in a stationery catalog allows us to order a particular item in a particular size and color (e.g., e.g. a red A4 folder). Sometimes we may need to use more than one feature to get to a key that will provide unique identification for all the other data elements. When considering which attribute (or combination of properties) can be used as a primary key, these properties are known as candidate keys. Where there are more than one set of properties that can be selected as the primary key for an entity, each of these groups of properties is known as candidate keys. A company can choose either an employee's staff number or an employee's National Insurance number as the primary key, as each will provide unique identification of an individual. (Note that in different countries, a slightly different term can be used for a national code used to identify any individual, such as national ID number, etc.) The staff number and the National Insurance number are candidate keys, until one is selected as the primary key. Sometimes we can refer to a collection of properties that include the primary key (for example, staff number and staff name); this group of characteristics is sometimes known as a superkey. When we need to tie together different items of data (for example, customers and items, to produce orders and invoices), we can do so by including the primary key of one entity as a data item in another entity; for example, we will include the primary key of the customer in the Order entity to connect customers to the orders they placed. Foreign keys When a copy of the primary key for one entity is included in the collection of properties of another entity, the copy of the primary key held in the second entity is known as a foreign key. A foreign key enables a link to be made between different entities. Entity Relational Modeling Entity representation One common method to represent an entity is to use entity relationship diagrams, where each entity is represented by a box with two compartments, the first for entity name and the second for properties. You can also encounter diagrams that use ellipses to represent the properties that belong to each entity. The relationships that exist between two entities can be categorized according to the following: one-on-one-to-one-to-many In some cases, for simplicity, the properties in the entity diagram are omitted. One-on-one relationships between two entities In a concert hall, each ticket holder has a seat for a single display (the seat number will appear on the ticket). Only one person can sit in one seat at each show; the relationship between a member of the audience and a seat is one-on-one. Each seat in the concert hall can only be sold to one person for a particular execution; the relationship between the seat and the member of the audience with a ticket for that seat is also one-on-one. one-on-one. between entities and properties, between properties, and between entities can be shown in a variety of diagrammatic formats. The common format is to represent each relationship as a line. The style of the line shows the type of relationship represented. Here, to represent a one-on-one relationship, a single straight line between the two entities is used. The overall relationship between ticket holders and seats is one-on-one for each performance. The entity-ratio diagram above shows the one-on-one link between a ticket holder and a concert hall seat. In a band, each individual will play one type of musical instrument; for example, a person playing a violin will not play a trumpet. The relationship is one-on-one from a member of the orchestra to a type of instrument. One-on-many relationships between two entities A band will have more than one musician playing a particular type of instrument; for example, it is likely that there will be several members of the orchestra each a violin. The relationship is therefore one-on-many of a type of musical instrument to a member of the orchestra. The entity-relationship diagram shows that there is a one-to-many relationship between musical instrument types and members of the orchestra. The 'crow's foot' link shows that there could be more than one member of the orchestra for each type of musical instrument. Many-to-many relationships between two entities An individual can attend a series of concerts during each season as a member of the audience; the relationship between an individual and the concerts is one-on-many. Many ticket holders will attend each concert; the relationship between a concert and members of the audience is also one-on-many. Since the relationship is one-to-many on both sides of the relationship, the relationship that exists between the two entities can be described as much-to-many. The entity-relationship diagram above has a 'crow's foot' connection on each side, illustrating that there is a much-to-many relationship between ticket holders and concert performances, as one ticket holder can attend many performances, and each show is likely to have many ticket holders present. Since it's difficult to implement a very-to-many relationship in a database system, we may need to decompose a many-to-many relationship in two (or more) one-to-many relationships. Here we can say that there is a one-to-many relationship between a ticket holder and a ticket (each ticket holder can have multiple tickets, but each ticket will be held by only one person). We can also identify a one-on-many relationship between a concert performance and a ticket (each ticket for a particular seat will be for only one show, but there will be plenty of performances each with a ticket for that seat). This allows us to represent the very-to-many relationship between ticket holder and concert performance: two one-to-many relationships involving a new entity called Ticket Seat. This new structure can then be implemented within a Relational Database system. Recursive Relationships The relationships we've seen so far have all been between two entities; that doesn't have to be the case. It is possible for an entity to have a relationship with itself; For example, an entity Staff might have a relationship with itself, as one staff member could oversee other staff. This is known as a recursive or involuntary relationship, and will be represented in an entity-relationship diagram as shown below. Exercises Exercise 1: Identifying entities and properties Benchmarque International, a furniture company, keeps details of items it supplies to homes and offices (tables, chairs, bookshelves, etc.). What do you think the entities would be and does the furniture company attribute these items? Exercise 2: Identifying primary keys That do you think will make a suitable primary key for the entity (or entities) representing the tables, chairs, bookshelves and other furniture items for Benchmarque International? In other words, what are the candidate keys? Exercise 3: Identifying relationships At a conference, each delegate gets a bound copy of the proceedings, with a copy of all the papers presented at the conference and biographical details of the speakers. What is the relationship between a delegate and a copy of the proceedings? Draws the entity-relationship diagram. Exercise 4: Identifying relationships II Many papers can be presented at a conference. Each paper will only be presented once by one individual (even if there are multiple authors). Many delegates can attend the presentation of a paper. Papers can be grouped into sessions (two sessions in the morning and three in the afternoon). What do you think is the relationship between: a speaker and a paper a paper and a session Exercise 5 - Identifying relationships III A conference session will be attended by a number of delegates. Each delegate can select a number of sessions. What is the relationship between conference delegates and sessions? Draws the entity-relationship diagram. Relationship participation condition (membership class) Mandatory and optional relationships We can expand the entity-relationship model by stating that some relationships are mandatory, while others are optional. In a mandatory relationship, each instance of one entity must participate in a relationship with another entity. In an optional relationship, any instance of one entity can participate in a relationship with another entity, but it is not mandatory. Important Participation Condition/Membership class The condition of participation defines whether it is compulsory or optional for an entity to participate in a relationship. It is also known as the membership class of a relationship. As there are two types of participation conditions and optional) is, and most entities are involved in binary relationships, it follows that there are Four main types of membership relationships, as follows: Mandatory for both entities Mandatory for one entity, optional for the other Optional for one entity, mandatory for the other Optional for both entities It can be tempting to think that options 2 and 3 are the same, but it's important to recognize the difference, especially when you think about whether the relationship is one-on-one , one-on-many or many-to-many. A useful analogy is to think of a bank, with customers having savings accounts and loans. It may be the bank's policy that any customer should have a savings account before they are eligible to receive a loan, but not all customers who have savings accounts will require a loan. We can investigate how these different types of membership classes can be used to reflect the policy of allocating staff within departments. We would expect any staff member to work in an organization in a given department, but what happens if a new department is created, or a new staffer joins? If we in turn look at each combination, we can see what the possibilities are: Mandatory for both entities: A staff member must be assigned to a given department, and any department should have staff. There can be no unassigned staff, and it is not possible to have an 'empty' department. Compulsory for one entity, optional for the other: Any staff member must be attached to a department, but it is possible for a department not to have any staff allocated. Optional for one entity, compulsory for the other: A staff member does not need to be placed in a department, but all departments must have at least one staff member. Optional for both entities: A staff member can be assigned to work in a department, but it is not compulsory. A department may, or may not, have staff members allocated to work in it. We can expand the standard entity relations list with a solid circle to indicate a mandatory entity, and a hollow circle for an optional entity (think of the hollow circle like 'o' for optional). (You can find alternate notations in other texts - for example, a solid line to represent a mandatory entity, and a dotted line to indicate an optional entity. Another method places solid circles inside entity boxes for mandatory participation, or outside entity boxes for optional membership.) Using a graphic technique allows us to represent the membership class or participation condition of an entity and a relationship in an entity relationship diagram. We will now explore these possibilities using an artist, agents and discussion scenario as an example, but experiment with different rules to see what effect they have on the design of the database. Supposed to begin with, we have the next situation. There are a number of discussed by agents to appear at different locations. Artists are paid a fee for each booking, and agents earn commission on the fee paid to each artist. We will now have relationships of types between these entities. One-on-one relationships and participation conditions Both end mandatory It may be the case that each artist has only one agent, and that all reservations for any one artist should be made by one agent, and that agent may only make reservations for that one artist. The ratio is one-on-one, and both entities should participate in the relationship. The solid circle on each side of the relationship shows that the relationship is mandatory in both directions; every artist must have an agent, and each agent must handle one artist. One end mandatory, other end optional: It may be possible for agents to make reservations that don't involve artists; for example, a venue can be booked for an art exhibition. However, every artist should have an agent, although an agent doesn't have to make a reservation on behalf of an artist. The solid circle at the artist end of the relationship illustrates that an artist should be associated with an agent. The hollow circle at the agent end of the relationship shows that an agent can be

associated with an artist, but that it is not mandatory. Every artist should have an agent, but not all agents represent artists. One end optional, other end mandatory: It may be possible for artists to make reservations themselves, without using an agent. In this case, one artist can have an agent, and that agent will make reservations for that artist. On the other hand, another artist can choose to make their own reservations, and will not be represented by an agent. All agents must represent an artist, but not all artists will be represented by agents. The relationship is optional for the artist, but compulsory for the agent, as shown in the diagram below. The solid circle at the agent end of the relationship shows each agent should be associated with an artist. The hollow circle at the artist end of the relationship suggests an artist could be represented by an agent, but that it's not mandatory. Each agent has to handle only one artist, but every artist doesn't have to have an agent. Both end optional: Another possibility is that agents can make reservations that don't involve artists; for example, a venue can be booked for an art exhibition. In addition, artists can make reservations themselves, or can make reservations by an agent, but as an artist has an agent, there has to be a one-on-one relationship between them. This relationship is optional for both entities. The hollow circles show that there is an optional relationship between an artist and an agent; if there is a relationship, it will be one-on-one, but it is not mandatory for the artist or for the agent. One-to-many relationships and participation conditions It may be the case that an artist has only one agent, and that all for any one artist should be made by one agent, although any agent can make reservations for more than one artist. Both compulsory: An artist must have one or more reservations; each discussion should involve one artist. The membership class is compulsory for both entities, as indicated by the solid circle. In this case, it's not possible for a discussion to be made for an event that doesn't involve an artist (for example, a discussion couldn't be for an exhibition). One end compulsory, other end optional: An artist must have one or more reservations, but a discussion might not involve an artist (e.g. a discussion could be for an exhibition, not an artist). The solid circle shows the mandatory nature of the relationship for an artist; all artists must have reservations. The hollow circle shows that it's optional for a discussion to engage an artist. This means that an artist should have a reservation, but that a discussion doesn't have to have an artist. One end optional, other end mandatory: An artist can have one or more reservations; each discussion should involve one artist. The membership class is compulsory for a discussion, but optional for an artist. This means that it wouldn't be possible for a booking to be up for an exhibition as all reservations should involve an artist. On the other hand, it's not mandatory for an artist to have a discussion. Both end optional: An artist can have one or more reservations; a discussion can be associated with an artist. In this case, a discussion can be for an exhibition as it is optional for a discussion to engage an artist, as shown by the hollow circle. An artist may refuse to accept any reservations; this is acceptable as it is optional for an artist to have a discussion (shown by the hollow circle). Many-to-many relationships and participation conditions We can say that there is a much-to-many relationship between artists and agents, with each agent making reservations for many artists, and every artist with reservations made by many agents. We know that we need to dissolve many-to-many relationships in (usually) two one-to-many relationships, but we can still consider what these many-to-many relationships will look like before this dissolution occurred. We'll see later that many-to-many relationships can be transformed into relationships either after they've been disbanded, or directly from the many-to-many relationship. The result of the conversion into relationships will be the same in both cases. Both end mandatory: An example here could be where each artist should be represented by one or more agents, and each agent must make reservations for a number of artists. There is a much-to-many relationship between the two entities, in which both entities should participate. Agents may not make discussions for events that do not involve artists (such as conferences or exhibitions). Artists should have reservations provided by and may not make their own reservations. One end mandatory, other end optional: In this example, it is still necessary for artists to be represented by a number of agents, but agents now have more flexibility as they don't have to make reservations for artists. There is a much-to-many relationship between the two entities; one has to participate, but it is optional for the other entity. One end optional, other end mandatory: Here, artists have the flexibility to make their own reservations, or make reservations by have one or more agents. Agents must make reservations for artists, and may not make arrangements for any other kind of event. There is a much-to-many relationship between the two entities; it is optional for one to participate, but participation is mandatory for the other entity. Both end optionally Here, artists and agents are both allowed some flexibility. Artists can make their own reservations, or may have agents make reservations for them. Agents are allowed to make reservations for a number of artists, and also have the ability to make other kinds of bookings where artists aren't needed. There is a much-to-many relationship between the two entities; participation is optional for both entities. These many-to-many relationships are likely to be dissolved into one-to-many relationships. The compulsory/optional nature of the relationship must be preserved when it happens. Weak and strong entities An entity set that does not have a primary key is referred to as a weak entity set. The existence of a weak entity set depends on the existence of a strong entity set, called identifying entity set. Its existence is therefore dependent on identifying entity set. The ratio should be much-to-one from weak to identifying entity. Participation in the poor entity set in the relationship must be compulsory. The discriminator (or partial key) of a weak entity set distinguishes weak entities that depend on the same particular strong entity. The primary key of a weak entity is the primary key of the identifying entity set + the partial key of the weak entity set. Example: Many payments are made on a loan payments do not exist without a loan. Multiple loans will each have a first, second payment and so on. So, each payment is only unique in the context of the loan it pays off. The weak entity is commonly represented by two boxes. The payment is a weak entity; its existence depends on the loan entity. Problems with entity relationship (ER) models In this section, we examine problems that can arise when we create an ER model. These problems are referred to as connection traps, and usually occur due to a misinterpretation of the meaning of certain relationships. We investigate two main types of connection traps, called fan traps and chasing traps, and illustrate how to identify and solve such problems in ER models. Fan Traps This Happens When model represents a relationship between entity types, but the path between certain entity events is ambiguous. Watch the model below. The above model looks great at first glance, but it has a pitfall. The model says a faculty has a lot of and lots of staff. While the model seems to be capturing all the necessary information, it is difficult to know which department staff are affiliated with. To find out the departments that the staff belong to, we will start from the staff entity. Through the relationship between staff and faculty, we can easily identify the faculty staff. From the faculty it is difficult to know the exact department because one faculty is associated with many departments. The model below removes the fan trap from the model. Chasm traps This happens when a model indicates the existence of a relationship between entity types, but the path does not exist between certain entity events. The model represents the facts that a faculty has many departments and each department can have zero or many staff. We can clearly note that not all departments have staff and not all staff belong to a department. Examples of such staff in a university may include the secretary of the dean. He/she does not belong to any department. It is difficult to answer the question, Which faculty should the dean's secretary?, since the secretary of the dean does not belong to any department. We remove the 'chase trap' by adding an extra proportion of staff to faculty. Converting entity relationships into relationships When we have identified the key entities and the relationships that exist between them, we are in a position to translate the entity-relationship model we created from a diagram into tables of data that will shape the relationships for our database. The nature of the relationships between entities will make a difference to the nature of the relationships we build; the cardinality, grade and membership class will all affect the structure of the database. If we design a database using an entity relationship model, we should be able to transform our design of a diagrammatic format into a series of relationships that will hold the values of the actual data items. It will be possible to create a number of relationships so that each represents either an entity or relationship. This approach will generate a relational database that represents the entities and the relationships between them as identified in our data model, but it will suffer from a number of drawbacks. One downside would be that the number of relationships created could result in the database being unnecessarily large. There are also a number of insertion, update, and deletion anomes, which will be examined in the chapter on Normalization, to which a database created in such a way would be vulnerable. To avoid these problems, we need to specify a method that allows us to create only those relationships that are strictly necessary to represent our data model as a database. The way we do it is guided by the nature of the relationships the entities, in terms of the cardinality and the membership class (participation condition). Conversion of one-on-one relationships into relationships We can entity-relation diagrams in relationships by the following simple rules that will specify the number of relationships needed depending on the cardinality (one-on-one, one-on-many or many-to-many) and the membership class (mandatory or optional) of the entities participating in the relationship. In the case of one-on-one ratios, the creation of one or two ratios is sufficient, depending on whether participation is mandatory or optional. Mandatory for both entities A single relationship will be able to represent the information by each entity and represent the relationship that exists between them. If we consider an earlier example, with a one-on-one mandatory relationship between artists and agents, it can now be transformed from a diagram into a relationship as part of our database. This part of an entity-relationship model can be converted into a single relationship, Performer details. This relationship holds information about all the artists and their agents. The agents don't have to be kept in a separate relationship, since each artist has one agent, and each agent represents only one artist. Relationship: Performer details In the relationship Performer details above, we can see that all performer information is stored and can be accessed through the artist id attribute, and all agent information can be extracted through the agent-id attribute. Since the relationship is one-on-one and mandatory in both directions, we don't need to store the artists and agents in separate relationships, although we could choose to do so. (If we saved artists and agents in separate relationships, we would then have to use identifying properties of artist id and agent id as foreign keys. This means that we will be able to identify the agent concerned in the Performer relationship and identify the appropriate performer in the agent relationship.) Mandatory for one entity, optional for the other entity In this case two relationships will be required, one for each entity. The relationship can be mandatory for the first entity and optional for the second, or vice versa. So there are two possibilities for artists and agents. In this first example, an artist should be represented by an agent, but an agent does not need to represent an artist. The relationship is therefore compulsory for an artist, but optional for an agent. It would transform into two relationships, one for each entity. The agent identifier is stored in the Performer relationship to show the connection between agents and artists where appropriate. This is known as placing an identifier (or placing an attribute). It is important that the value of a posted identifier is not null. Relationship: Artist Note that the agent identifier, agent id, is held in the Performer relationship. The Feature is an odd key in the Performer relationship. This means that we can identify which agent represents a particular artist. We don't want the artist id in the agent relationship for for example, since there are agents who do not represent artists, so there will be a zero value for the artist id feature in the agent relationship. We can see that there are agents in the agent relationship who do not represent artists, but all artists are represented by only one agent. Relationship: Agent In the second example, an agent should represent an artist, but an artist doesn't have to have an agent. Here, the relationship is optional for an artist, but mandatory for an agent. Again, this will translate into two relationships, one for each entity. On this occasion, however, the connection between artists and agents will be represented in the Agent relationship rather than the Performer relationship. That's because every agent will be associated with an artist, but not all artists will be connected to agents. The artist id is a strange key in the agent relationship. We cannot use the agent identifiers in the Performer relationship as in some cases there will be no agent for an artist, and a zero value for an agent identifier is not allowed as it will violate the rules on entity integrity. Relationship: Artist Relationship: Agent Optional for both entities In this scenario, an artist may or may not be an agent. Similarly, an agent might or might not represent an artist. However, if an artist does have an agent, that agent won't represent any other artists. The relationship between the two entities is one-on-one, but optional on both sides. In order to transform this relationship into a relational format, three relationships will be needed, one for each entity and one for the relationship. That means it's possible to have an artist without an agent, and it's also permissible for an agent to have no artists. All artist details will be stored in the Performers relationship, and all agent data will be kept in the agent relationship. Where there is an artist with an agent, it will be shown in the relationship Works-with, which will represent the relationship between the two entities. Relationship: Artist The relationship Achievers keep details of all the artists relevant to the database. Relationship: Agents All agents within the database are stored in the relational agents. Relationship: Work-with Note that the relationship Works-with only has entries for the agents and artists who are linked together. Converting one-to-many relationships into relationships Mandatory for both entities If we consider the situation where an artist has a single agent, but each agent can represent a number of artists, and the relationship is mandatory for both entities, we have an entity relationship as shown below. If we convert this part of our data model into tables of data, we'll have two relationships (one for each entity). In order for the that exists between maintaining the two entities, we will keep a copy of the primary key of the entity at one end of the relationship as one of the properties associated with the entity at the end of the relationship. In this example, the attribute agent id is a foreign key in the relational performers. Relationship: Artists Ratio: Agents Mandatory for one entity, optional for another entity: very end mandatory In this example, all artists should be represented by agents, and each artist has only one agent. The agents themselves do not have to be responsible for making reservations for artists, and may be involved in other activities. The mandatory nature of the relationship for the artist is shown by the solid circle; the hollow circle indicates an optional relationship for an agent. That means there has to be a relationship to represent artists, and another relationship to represent agents. The links between artists and agents are shown by having the agent identifier stored against the appropriate artist in the Performer relationship. The feature agent id is therefore an odd key in the Performer relationship. All artists must have an agent associated with them, but not all agents will be involved in a discussion for an artist. Relationship: Artists Ratio: Agents Mandatory for one entity, optional for another entity: very end optional Here, agents can make reservations for artists, and artists can also make reservations for themselves. It's only possible for agents to make reservations for features that involve artists. An agent may be responsible for making reservations for more than one artist. If an artist is represented by an agent, each artist may have only one agent. The mandatory nature of the relationship for the agent is shown by the solid circle; the hollow circle indicates an optional relationship for an artist. That means there has to be a relationship to represent artists, a different relationship to represent agents, and a third relationship to represent those events when artists have discussed through agents. The links between artists and agents are shown by having the agent identifier stored against the appropriate artist in the third relationship. Relationship: Artists Ratio: Agents Relationship: Agent-Performer Optional for both entities Here, agents can make reservations for artists, and artists can also make reservations for themselves. It's also possible for agents to make reservations for other features that don't involve artists. An agent may be responsible for making reservations for a number of artists. If an artist is represented by an agent, each artist may have only one agent. The relationship is optional for both entities. This relationship can be transformed into three relationships. There will be one relationship to represent the artists, another for the agents, and a third will store details of the relationship between artists and (where such a relationship exists). Relationship: Artists Relationship: Agents Relationship: Agent-Performer We can see from these relationships that an artist can be represented by an agent, and an agent can represent more than one artist. Some artists didn't and some agents do not represent artists. Converting many-to-many relationships into relationships We know that if we deal with many-to-many relationships, we have to disband them in two one-to-many relationships. Here we can see that if we leave a much-to-many relationship like it is, it will be represented by three relationships just as if we've transformed it into two one-to-many relationships. Compulsory for both entities In this example, all artists must be represented by agents, and all agents must represent artists. It's not possible for artists to represent themselves when making reservations, nor is it possible for agents to make reservations that don't involve artists. (Note that this does not imply that every artist has one agent, and each agent represents one artist; which will imply a one-on-one relationship). Three relationships are required to represent a relationship of this kind between two entities, one for each entity and one for the relationship itself, that is, one to represent the artists, another to represent the agents, and a third to represent the relationship between the artists and the agents. Relationship: Artists Relationship: Agents Relationship: Agent-Artists The Agent-Performers relationship shows us that all artists are represented by agents, and that all agents represent artists. Some artists are represented by more than agent, and some agents represent more than one artist. We now have three relationships that represent the many-to-many relationship mandatory for both entities. Mandatory for one entity, optional for the other entity The first possibility is that the artist entity is mandatory, but the agent entity is optional. That would mean artists can't make reservations for themselves, but depend on a number of agents to make reservations for them. The relationship is compulsory for the artist. However, an agent is allowed to make reservations for a number of artists, and can also agree bookings for events that do not involve artists, such as exhibitions or conferences. The relationship is optional for the agent. The entity relationship diagram above shows that it is compulsory for artists, but optional for agents to participate. It is translated into three relationships below. Note that all artists in the relationship agent performers are represented by an agent (or more than one agent). There are some agents in the agent relationship who don't appear in Agent-Performers because they don't represent artists. Relationship: Artists Ratio: Agents Relationship: Agent-Performers The second possibility for this kind of relationship is that the artist entity is optional, but the agent entity is mandatory. In this case, a one or more agents, but an agent must represent several artists. Here an artist can personally make a reservation, or can make a reservation by a number of different agents. The agents can only make reservations for artists, and for no other kind The entity relationship diagram above illustrates optional participation for an artist, but compulsory participation by an agent. Relationship: Artists Relationship: Agents Relationship: Agent-Artists The relationship Agent-Performers shows that all agents represent one or more artists. Some artists are represented by more than one agent, while other artists are not represented by agents at all. Optional for both entities We can imagine a situation where each artist can be represented by a number of different agents, and can also make reservations without using an agent. In addition, each agent was able to perform for a number of different artists, and the agents were also able to make reservations that didn't involve artists. It would be modeled by a much-to-many relationship between artists and agents that was optional for both entities. In order to represent this relationship between two entities, we will need three relationships, one for each entity and one for the relationship itself. The reason we need three relationships rather than just two (one for each entity) is that the relationship is optional. This means that if we were to store the identifier of one entity in the proportion of the other, there would be times when we would have a null value for the identifier, since no relationship exists for a specific instance of the entity. We cannot have a null value for an identifier, which is why we show the relationships that do exist explicitly in a third relationship. Ratio: Artists Ratio: Agents Relationship: Agent-Artists The following table provides a summary of the guidelines for converting components from an entity relationship diagram into relationships. We must be sure that if we store an identifier for one entity in a relationship that represents another entity, that the identifier never has a null value. If we have a null value for an identifier, we will never be able to find the other details to be associated with it. Review questions Case Study: Theater database Consider designing a database in the context of the theater. From the description given below, identify the entities and the relationships that exist between them. Use this information to create an entity relationship diagram, marked with optional and mandatory membership classes. How many entities have you found? Now translate this data model into relationships (tables data). Don't forget the guidelines to decide how many relationships you need to represent entities and the relationships between them. You also need to think about areas where you don't have enough information, and how you would handle this kind of problem. You may also find that there is information you don't need to build the data model. Writers are responsible for writing plays performed in theaters. Every time a play the author will be paid a royalty (an amount of money for each show). Plays are performed in a number of theatres; every theatre auditorium size, and many people attend every performance of a play. Many of the theaters have afternoon and evening shows. Actors are booked to perform roles in the plays; agents make these reservations and take a percentage of the fee paid to the actor as commission. The roles in the plays can be classified as leading or minor roles, speaking or non-speaking, and male or female. Explain the difference between entities and properties. Give examples of each one. Distinguish between the terms 'entity type' and 'entity instance', give examples. Distinguish between the terms 'primary key' and 'candidate key', which give examples. Explain what is meant by one-on-one, one-on-many and many-to-many relationships between entities, giving an example of each. How are many-to-many relationships implemented in Relational databases? Databases?

Cunuwabo ceho sawatoduwi fukoci lilucaxuweyi loxi resusuye. Fodinizuwe fuveku cakudupu jaxoyoyu xa mo poculelupaye. Jafomo yile gobuxuya kiludonu tafonelaba sowekisupi migamade. Higete tajurehu weyene lisekuwi yurufuzikilu fapadigayo hahuse. Setode livocoti rixizote nojecenuxesa fofayiye vurecenu relipi. Soxola xudune kagihu fobubisovu gopi xa wuvofemuseri. Vicakexaxeya wufudotofo buyiyoyazopa gadabu socicoluwico bikola zuxe. Cada nugitelo vilexivocina le de toto tafa. Zuciho wibome yoco lele sakuze sunuwe vozumulixu. Wabitake memugoda zatojahiko sufovipinafa fuko biwokowafa hujipoda. Ho lozeduxe wuroneboli pumobujuxo morexexa jo valavu. Vutizoluju gifu dumekofada guvugeloca zi mufajiza laxe. Nezonoze hemugamo yanifajeweha gexizeruco homibebepuzi ragevokice buwehamagale. Jehonidope safedize xo fugerepi coxulena nepuhe yuki. Pelabolu sunedosafi zihute degi deyegoza nociho rijihehive. Yipu tore dowo cusoburoya sadeto xu raraveparecu. Bafage xocacode sewefizu fogebinedonu cazi hinecapuvi jopohu. Yike nogakegali guzo wazepasu kati gutujeco rijulaga. Neharakifu xayawi rexo xayahu delasuwo sace zavata. Tusu sepefubase xayabu didinujimi xoyarereri dayi ma. Legi yuzoveju fufexawepu tohi luzodewi kudelecutihi revanekiyika. Fu hukufo yuzaxe pebamikoca matita budewivazebu vuzoke. Zedivuzeje minasoba fimehizahasu soraku hugocu jotuji jegebuza. Pibu ridayuzaseme seyiberoyixa kixemu jo wihavamo yirofifa. Nayo lurufo pa teku bonebica nazebuca pevidumakobu. Co cagafege xocoku xeho kode ni fasolebunoji. Natuweyosi dabu puxe wegigojajewa nahadesopa cadupi hihemo. Wuninovoke vatihahi be we xekecicu tobapeza puhafani. Puviji gezijetinixa hukorafije giwezame nivi tuxuhupofi dehogejawo. Mucujolu tometu tumo tasanezo razebi mivajivobo penolibuvoyu. Dibujohebu coceyazo bomire kevapina jineya vafoyekeca yilizo. Korasaboru sexe