DOWNLOAD PDF ADDITIONAL FEATURES OF ER MODEL IN DBMS

Chapter 1 : What is entity-relationship model? at EXPLAIN EXTENDED

Extended E-R Features Although the basic E-R concepts can model most database features, some aspects of a database may be more aptly expressed by certain extensions to the basic E-R model. In this section, we discuss the extended E-R features of specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and.

This course will enable students to Provide a strong foundation in database concepts, technology, and practice. Practice SQL programming through a variety of database problems. Demonstrate the use of concurrency and transactions in database Design and build database applications for real world problems. Module â€” 1 Teaching Hours 10 Introduction to Databases: Overview of Database Languages and Architectures: Data Models, Schemas, and Instances. Three schema architecture and data independence, database languages, and interfaces, The Database System environment. Conceptual Data Modeling using Entities and Relationships: Entity types, Entity sets, attributes, roles, and structural constraints, Weak entity types, ER diagrams, examples, Specialization and Generalization. Relational Model Concepts, Relational Model Constraints and relational database schemas, Update operations, transactions, and dealing with constraint violations. Examples of Queries in relational algebra. Mapping Conceptual Design into a Logical Design: Module â€” 4 Teaching Hours 10 Normalization: Introduction to Transaction Processing, Transaction and System concepts, Desirable properties of Transactions, Characterizing schedules based on recoverability, Characterizing schedules based on Serializability, Transaction support in SQL. Concurrency Control in Databases: Two-phase locking techniques for Concurrency control, Concurrency control based on Timestamp ordering, Multiversion Concurrency control techniques, Validation Concurrency control techniques, Granularity of Data items and Multiple Granularity Locking. The students should be able to: Design and build simple database systems Develop application to interact with databases. The question paper will have TEN questions. There will be TWO questions from each module. Each question will have questions covering all the topics under a module. Navathe, 7th Edition, , Pearson. Database management system is software designed to assist the maintenance and utilization of large scale collection of data. DBMS came into existence in by Charles. Integrated data store which is also called as the first general purpose DBMS. In to there were advances in DBMS e. Which is the task associated with gathering the data as and when they originate. Captured data has to be classified based on the nature and intended usage. The segregated data has to be stored properly.

Chapter 2 : Generalization, Specialization and Aggregation in ER Model | Studytonight

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases. An entity can be a real-world object, either animate or inanimate, that can be easily.

In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram. For example, in the below diagram, anyone can see and understand what the diagram wants to convey: Developer develops a website, whereas a Visitor visits a website. Components of ER Diagram Entitiy, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them. Entity Simple rectangular box represents an Entity. Relationships between Entities - Weak and Strong Rhombus is used to setup relationships between two or more entities. Attributes for any Entity Ellipse is used to represent attributes of any entity. It is connected to the entity. Weak Entity A weak Entity is represented using double rectangular boxes. It is generally connected to another entity. Key Attribute for any Entity To represent a Key attribute, the attribute name inside the Ellipse is underlined. Derived Attribute for any Entity Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth. To represent a derived attribute, another dotted ellipse is created inside the main ellipse. Multivalued Attribute for any Entity Double Ellipse, one inside another, represents the attribute which can have multiple values. Composite Attribute for any Entity A composite attribute is the attribute, which also has attributes. Entity An Entity can be any object, place, person or class. In ER Diagram, an entity is represented using rectangles. Consider an example of an Organisation- Employee, Manager, Department, Product and many more can be taken as entities in an Organisation. The yellow rhombus in between represents a relationship. Weak Entity Weak entity is an entity that depends on another entity. Double rectangle is used to represent a weak entity. Attribute An Attribute describes a property or characterstic of an entity. For example, Name, Age, Address etc can be attributes of a Student. An attribute is represented using eclipse. Key Attribute Key attribute represents the main characterstic of an Entity. It is used to represent a Primary key. Ellipse with the text underlined, represents Key Attribute. Composite Attribute An attribute can also have their own attributes. These attributes are known as Composite attributes. Relationship A Relationship describes relation between entities. Relationship is represented using diamonds or rhombus. There are three types of relationship that exist between Entities.

Chapter 3 : Create a Database Model (also known as Entity Relationship diagram) - Visio

additional features of er model in dbms and conceptual design with er model in dbms. theinnatdunvilla.comros,United Kingdom,Teacher. Published Date:

Image by Samuel Mann A relational database, as we all know from the previous article , stores relations between integers, strings and other simple types in a very plain way: This model is extremely flexible, since other relations can be easily derived from existing ones using mathematical formulae, and the relational database takes care of that. However, database should reflect the real world in some way to be really of use. Since the relational databases store relations between mathematical abstractions and not real world things, we should make some kind of a mapping of ones to the others. This is what entity-relationship model is for. An entity-relationship model, as one can easily guess from its name, models relationships between entities. But since we know that databases do essentially the same, how does it differ from the database model? An entity-relationship model states which data and relations between them should be stored A database model states how these relations are stored In other words, ER model is design and database model is one of the ways to implement it. ER model is said to be above the database model in the waterfall developement. Note that in both cases I use the word stored above. The model says nothing of data and relations between them that may or should exist, only of those that should be stored. Every human being participates in thousands if not millions relationships, but an ER model should state which of them are to be stored, and a relational model should decide how to store them. Of course it would be nice to store them all and police and insurance would be just happy , but the technology does not allow it yet. What does it mean? Square boxes mean entites. What do we describe in our database? In our case, the model requires that we store information about clients, orders and items ordered. Square boxes are useless without additional information. We just make a file to which a new line is added each time a new customer registers: A customer has registered A customer has registered A customer has registered â€¦ , another file to which a new line is added when an order is made: An order is made An order is made An order is made â€¦ and a third file to keep the list of items ordered. An item is ordered An item is ordered â€¦ Quite useless, right? However, there still is some information, as per request. This is better than nothing but still far from being usable. Note that the diagram tells us nothing of how they should be stored. We could write them into a file, or on a paper or carve it on a stone. Relationships Diamonds mean relationships. How the entities are related to each other? We see that we should store the relationships between orders and clients as well as those between items and orders. In our cases, we have a 1: Want to store an order? Be prepared to link it to the piece of information about the client who made it and make sure this information describes exactly one client. However, a client can have an arbitrary number of orders: The database should provide an ability to store the clients and orders this way. Again, the entity-relationship model does not prescribe how should we store that data, as long as the storage method satisfies the conditions above. We could store it in two files and relate them using the row numbers: A customer has registered A customer has registered A customer has registered â€¦ An order is made by the customer described on line 1 An order is made by the customer described on line 2 An order is made by the customer described on line 2 , or we can just keep the information in a single text file: The latter one limits us to only 2 clients and 5 orders, but, you know, every system has its limitations. Attributes Ovals are attributes. What information is stored in the database? Mere enumerating the clients is nice but serves no purpose. Much better if you know the names of the clients; the orders need to be assigned with unique numbers that help to identify them; and it would be great to record which good did which item contain so not only the number of packages could be checked but their contents too. This is in fact what the database is for: Not only the links between the entities but the descriptions of the entities too. Entity-relationship and relational model Everything above should be squeezed into relational model, which as we all know stores relations. Since the time relational database appeared, they were mostly used to implement ER models. Multiple database manuals and guides describe the relational databases solely from that point of view. Various tools exist to automatically generate relational structure given a model. However, ER model and a relational database are not the same. There is even no mapping to either side: Due to the way the data are stored in a relational model, there is no reliable way to tell between attributes, entities and relationships by looking only at the relational model. These terms belong to the ER model. In a relational model, one thing can be implemented as an entity, relationship or an attribute. In this article, I will give several examples. Imagine a simple model as pictured in the diagram on the right. The model requires that the database store fictional characters as the entities. For each fictional character it should store their name, address, town and state as the attributes. There are no relations here: As I already said earlier, the ER model specifies what should be stored and the database design relational model in this case decides how. One ot the benefits of the relational model is that is can construct the data representation on the fly, using SQL statements. One of the possible ways to implement this ER model in a relational database is to store the entities and their attributes in one table, like this:

Chapter 4 : Creating ER Diagram Representation in DBMS | Studytonight

Working with ER Diagrams. ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram.

Different shapes at the ends of these lines represent the relative cardinality of the relationship. With this notation, relationships cannot have attributes. Where necessary, relationships are promoted to entities in their own right: Model usability issues[edit] You can help by adding to it. February In using a modeled database, users can encounter two well known issues where the returned results mean something other than the results assumed by the query author. It occurs with a master table that links to multiple tables in a one-to-many relationship. This type of model looks similar to a star schema , a type of model used in data warehouses. When trying to calculate sums over aggregates using standard SQL over the master table, unexpected and incorrect results. The solution is to either adjust the model or the SQL. This issue occurs mostly in databases for decision support systems, and software that queries such systems sometimes includes specific methods for handling this issue. A chasm trap occurs when a model suggests the existence of a relationship between entity types, but the pathway does not exist between certain entity occurrences. For example, a Building has one-or-more Rooms, that hold zero-or-more Computers. One would expect to be able to query the model to see all the Computers in the Building. However, Computers not currently assigned to a Room because they are under repair or somewhere else are not shown on the list. Another relation between Building and Computers is needed to capture all the computers in the building. This last modelling issue is the result of a failure to capture all the relationships that exist in the real world in the model. See Entity-Relationship Modelling 2 for details. Entityâ€"relationships and semantic modeling[edit] Semantic model[edit] A semantic model is a model of concepts, it is sometimes called a "platform independent model". It is an intensional model. At the latest since Carnap , it is well known that: The first part comprises the embedding of a concept in the world of concepts as a whole, i. The second part establishes the referential meaning of the concept, i. Extension model[edit] An extensional model is one that maps to the elements of a particular methodology or technology, and is thus a "platform specific model". The UML specification explicitly states that associations in class models are extensional and this is in fact self-evident by considering the extensive array of additional "adornments" provided by the specification over and above those provided by any of the prior candidate "semantic modelling languages". It incorporates some of the important semantic information about the real world. Plato himself associates knowledge with the apprehension of unchanging Forms The forms, according to Socrates, are roughly speaking archetypes or abstract representations of the many types of things, and properties and their relationships to one another. Limitations[edit] ER assume information content that can readily be represented in a relational database. They describe only a relational structure for this information. They are inadequate for systems in which the information cannot readily be represented in relational form[citation needed], such as with semi-structured data. For many systems, possible changes to information contained are nontrivial and important enough to warrant explicit specification. An alternative is to model change separately, using a process modeling technique. Additional techniques can be used for other aspects of systems. For instance, ER models roughly correspond to just 1 of the 14 different modeling techniques offered by UML. Even where it is suitable in principle, ER modeling is rarely used as a separate activity. These tools can readily extract database diagrams that are very close to ER diagrams from existing databases, and they provide alternative views on the information contained in such diagrams. In a survey, Brodie and Liu [20] could not find a single instance of entityâ€"relationship modeling inside a sample of ten Fortune companies. Badia and Lemire [21] blame this lack of use on the lack of guidance but also on the lack of benefits, such as lack of support for data integration. The enhanced entityâ€"relationship model EER modeling introduces several concepts not in ER modeling, but are closely related to object-oriented design, like is-a relationships. For modelling temporal databases , numerous ER extensions have been considered.

Chapter 5 : Additional Features of the ER Model IN DBMS Archives - Jugaruinfo

Chapter 2: Entity-Relationship Model Entity Sets Relationship Sets Design Issues Mapping Constraints Keys E-R Diagram Extended E-R Features Design of an E-R Slideshare uses cookies to improve functionality and performance, and to provide you with relevant advertising.

Less With the Database Model Diagram template, you can create a new model or reverse engineer an existing database into a model by using either the relational or object relational modeling concepts. Use the Entity Relationship stencil to model databases that are based on the SQL92 and earlier standards. Use the Object Relational stencil, which has additional shapes for working with types, to model databases that are based on SQL99 and later standards. If you are using Visio Pro for Office and want to learn about how to engineer an existing database into a database model, see the topic, Reverse engineer an existing database. This article describes how you can create a database model and what you can do with the model after you create it. Not every edition of Microsoft Visio has the database model feature. If you cannot find the features described in the procedures in this article, most likely you have an edition of Visio that does not include them. Microsoft Visio Professional and Premium editions support the reverse engineering features for the Database Model Diagram template that is, using an existing database to create a model in Visio , but it does not support forward engineering that is, using a Visio database model to generate SQL code. For more information, see see the topic, Reverse engineer an existing database. To start your database model diagram, do one of the following: Click the File tab. On the Database tab, in the Manage group, click Display Options. In the Database Document Options dialog box, select the symbol set that you want to use and other table and relationship options, and then click OK. Use an existing database as a starting point If you have a database that you want to model so that you can understand it better or use it as a starting place for a new model, you can use the Reverse Engineer Wizard to extract the schema, or structure, of the database and build a new model. Before you start the wizard: If you are reverse engineering a Microsoft Excel workbook, before you start the wizard you need to open the workbook and name the group or range of cells that contains the column headings. If you want to use more than one worksheet, just name the group of column cells in each worksheet. These ranges are treated like tables in the wizard. For more information about how to name a range of cells, see the topic in your Excel help titled Define named cell references or ranges. For best results, set your default driver to the target database that you want to reverse engineer before you run the Reverse Engineer Wizard. This step ensures that the wizard maps the native data types correctly and that all the code that is extracted by the wizard is correctly displayed in the Code window. On the Database tab, in the Model group, click Reverse Engineer. On the first screen of the Reverse Engineer Wizard, do the following: Select the database driver for your database management system DBMS. Select the data source of the database that you are updating. If you have not already created a data source for the existing database, click New to do so now. When you create a new source, its name is added to the Data Sources list. When you are satisfied with your settings, click Next. Follow the instructions in any driver-specific dialog boxes. For example, in the Connect Data Source dialog box, type the user name and password, and then click OK. If you use the ODBC Generic Driver, you may receive an error message that indicates that the reverse engineered information may be incomplete. Select the check boxes for the type of information that you want to extract, and then click Next. Some items may be unavailable appear grayed out because not all database management systems support all the kinds of elements that the wizard can extract. Select the check boxes for the tables and views, if any that you want to extract, or click Select All to extract them all, and then click Next. If you selected the Stored Procedures check box, select the procedures that you want to extract, or click Select All to extract them all, and then click Next. Select whether you want the reverse engineered items to be added automatically to the current page. You can choose to have the wizard automatically create the drawing, in addition to listing the reverse engineered items in the Tables and Views window. If you decide not to have the drawing created automatically, you can drag the items from the Tables and Views window onto your drawing page to manually assemble the database model. Review your selections to verify that you are extracting the information that you want, and then click

Finish. The wizard extracts the selected information and displays notes about the extraction process in the Output window. This ability is limited to only VisioModeler 2. On the Database tab, in the Model group, click Import, and then click the model type. Type the path and file name for the model that you want to import, or click the Browse button to locate the model file, and then click Open. In the Import dialog box, click OK. Visio imports the file and displays its progress in the Output window. The imported tables are displayed in the Tables and Views window. In the Tables and Views window, select the tables that you want to model, and then drag them onto the drawing page. After you create a database model diagram, the work of refining the diagram begins. You can add and customize tables and views, create relationships, and customize columns and data types. Tables Use the Entity shape to create a table in your diagram. From either the Entity Relationship or Object Relational stencil, drag an Entity shape onto the drawing. Double-click the shape to open the Database Properties window. Under Categories, click Definition and type a name for the table. Under Categories, click Columns, type a name, and choose a data type. Select the PK primary key check box for columns that uniquely identify each row in the database table. Columns Use the Database Properties window to add or change properties for columns, including data types and primary keys. Double-click the table in your diagram. In the Database Properties window, under Categories, click Columns. Click in the first empty Physical Name cell, and type a name. For example, you can type decimal 8,2 or char To specify that the column is a primary key, select the PK check box. To see more column properties in addition to those that appear when you click the Columns category, select the column and then click Edit. Relationships Relationships use primary and foreign keys to allow databases to match a row in one table with a row in a related table. You can show those relationships in your diagram. Create a relationship between tables: Make sure that both tables are visible in the diagram. If you reverse engineered the model from an existing database, you may need to drag one or both from the Tables and Views window onto the drawing page. Double-click the table that you want for the primary key side of the relationship. In the grid, click the column that you want to use to uniquely identify each row in the table, and select the PK check box to set it as the primary key. From the Object Relational or Entity Relationship stencil, drag a Relationship shape and drop it onto a blank space on the page. Connect the higher end to the table with the parent table. Connect the other end to the child table. If relationship lines disappear, on the Database tab, in the Manage group, click Display Options. On the Relationships tab, under Show, select the Relationships check box. In the Database Properties window, under Categories, click Miscellaneous. Under Cardinality, choose the cardinality that best fits the relationship. For one-to-many relationships, the best choice is either Zero or more or One or more. For one-to-one relationships, the best choice is either Zero or one or Exactly one. To make other refinements to your diagram such as creating indexes, check clauses, and triggers you can do the following: Create indexes Indexes improve the performance, or speed, of your database when you run a query. Open the database model diagram. Double-click the table to which you want to add an index, and in the Database Properties window, in the Categories list, click Indexes. In the Create Index dialog box, type a name for the index, and then click OK. In the Index Type list, select an option to create a unique or non-unique index. In the Indexed Columns list, select the Asc check box to create an index that has an ascending sort order, or clear the check box to create an index that has a descending sort order. The database model diagram is updated. Create views You can think of a view as a saved query. Views are particularly handy if you need to repeatedly access the same information from multiple tables, or if you want to expose the data to users without letting them change the actual tables. Set extended properties for tables and views Depending on your database management system DBMS , you may be able to set extended properties for tables or views to determine where they are stored. Double-click the table or view whose extended properties you want to set, and in the Database Properties window, in the Categories list, click Extended. Create check clauses Use check clauses to ensure that the data that is entered into a column is within a particular range of values. For example, you can create a check clause that requires the data in a column called "Age" to be over Double-click the table to open the Database Properties window. Under Categories, click Columns and then click the column that you want to add a check clause to. On the Check tab of the Column Properties dialog box, enter the constraints that you want. The check clause is added to the Code window under Local code. Create stored procedures and user-defined functions Use stored procedures and user-defined functions to create packets of code that you can reuse to perform the same actions repeatedly. The major difference between the two is that a user-defined function returns a value, whereas the stored procedure executes code without returning a value. Click Global Code and then click New.

DOWNLOAD PDF ADDITIONAL FEATURES OF ER MODEL IN DBMS

Chapter 6 : DBMS BY RAGHU theinnatdunvilla.com - Google Drive

Database Management System(DBMS) A database is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about the following: Entities such as students, faculty, courses, and classrooms.

Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram. Entity Entities are represented by means of rectangles. Rectangles are named with the entity set they represent. Attributes Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity rectangle. If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse. Multivalued attributes are depicted by double ellipse. Derived attributes are depicted by dashed ellipse. Relationship Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities rectangles participating in a relationship, are connected to it by a line. Binary Relationship and Cardinality A relationship where two entities are participating is called a binary relationship. Cardinality is the number of instance of an entity from a relation that can be associated with the relation. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship. It depicts many-to-many relationship. Total participation is represented by double lines. Partial participation is represented by single lines.

Chapter 7 : Database Management System Notes pdf files - DBMS Notes pdf files - JNTU World Updates

Data base design and ER diagrams - Beyond ER Design Entities, Attributes and Entity sets - Relationships and Relationship sets - Additional features of ER Model - Concept Design with the ER Model - Conceptual Design for Large enterprises.

Email Extended E-R Features Although the basic E-R concepts can model most database features, some aspects of a database may be more aptly expressed by certain extensions to the basic E-R model. In this section, we discuss the extended E-R features of specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and aggregation. Specialization An entity set may include subgroupings of entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings. Consider an entity set person, with attributes name, street, and city. For example, customer entities may be described further by the attribute customer-id, whereas employee enti- ties may be described further by the attributes employee-id and salary. The process of designating subgroupings within an entity set is called specialization. The specialization of person allows us to distinguish among persons according to whether they are employees or customers. As another example, suppose the bank wishes to divide accounts into two categories, checking account and savings account. Savings accounts need a minimum balance, but the bank may set interest rates differently for different customers, offering better rates to favored customers. The bank could then create two specializations of account, namely savings-account and checking-account. As we saw earlier, account entities are described by the attributes account-number and balance. The entity set savings-account would have all the attributes of account and an additional attribute interest-rate. The entity set checking account would have all the attributes of account, and an additional attribute overdraft amount. An entity set may be specialized by more than one distinguishing feature. In our example, the distinguishing feature among employee entities is the job the employee performs. Another, coexistent, specialization could be based on whether the person is a temporary limited-term employee or a permanent employee, resulting in the entity sets temporary-employee and permanent-employee. When more than one specialization is formed on an entity set, a particular entity may belong to multiple specializations. For instance, a given employee may be a temporary employee who is a secretary. The ISA relationship may also be referred to as a superclass-subclass relationship. Higher- and lower-level entity sets are depicted as regular entity sets â€” that is, as rectangles containing the name of the entity set. The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features. There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. In our example, person is the higher-level entity set and customer and employee are lower-level entity sets. Higher- and lower-level entity sets also may be designated by the terms superclass and subclass, respectively. The person entity set is the superclass of the customer and employee subclasses. For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R schema for an enterprise. In terms of the E-R diagram itself, we do not distinguish be- tween specialization and generalization. New levels of entity representation will be distinguished specialization or synthesized generalization as the design schema comes to express fully the database application and the user requirements of the database. Differences in the two approaches may be characterized by their starting point and overall goal. Specialization stems from a single entity set; it emphasizes differences among entities within the set by creating distinct lower-level entity sets. These lower-level entity sets may have attributes, or may participate in relationships, that do not apply to all the entities in the higher-level entity set. Indeed, the reason a designer applies specialization is to represent such distinctive features. If customer and employee neither have attributes that person entities do not have nor participate in different relationships than those in which person entities participate, there would be no need to specialize the person entity set. Generalization proceeds from the recognition that a number of entity sets share some common features namely, they are described by the same attributes and participate in the same relationship sets. On the basis of their commonalities, generalization synthesizes these entity sets into a single, higher-level entity set. Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences; it also permits an economy of representation in that shared attributes are not repeated. Attribute Inheritance A crucial property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance. The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets. For example, customer and employee inherit the attributes of person. Thus, customer is described by its name, street, and city attributes, and additionally a customer-id attribute; employee is described by its name, street, and city attributes, and additionally employee-id and salary attributes. A lower-level entity set or subclass also inherits participation in the relationship sets in which its higher-level entity or superclass participates. Attribute inheritance applies through all tiers of lower-level entity sets. The above entity sets can participate in any relationships in which the person entity set participates. Whether a given portion of an E-R model was arrived at by specialization or generalization, the outcome is basically the same: In a hierarchy, a given entity set may be involved as a lower-level entity set in only one ISA relationship; that is, entity sets in this diagram have only single inheritance. If an entity set is a lower-level entity set in more than one ISA relationship, then the entity set has multiple inheritance, and the resulting structure is said to be a lattice. Constraints on Generalizations To model an enterprise more accurately, the database designer may choose to place certain constraints on a particular generalization. One type of constraint involves determining which entities can be members of a given lower-level entity set. Such membership may be one of the following: For example, assume that the higher-level entity set account has the attribute account-type. For instance, let us assume that, after 3 months of employment, bank employees are assigned to one of four work teams. We therefore represent the teams as four lower-level entity sets of the higher-level employee entity set. Instead, the user in charge of this decision makes the team assignment on an individual basis. The assignment is implemented by an operation that adds an entity to an entity set. A second type of constraint relates to whether or not entities may belong to more than one lower-level entity set within a single generalization. The lower-level entity sets may be one of the following: A disjointness constraint requires that an entity belong to no more than one lower-level entity set. In our example, an account entity can satisfy only one condition for the account-type attribute; an entity can be either a savings account or a checking account, but cannot be both. In overlapping generalizations, the same entity may belong to more than one lower-level entity set within a single generalization. For an illustration, consider the employee work team example, and assume that certain managers participate in more than one work team. A given employee may therefore appear in more than one of the team entity sets that are lower-level entity sets of employee. Thus, the generalization is overlapping. As another example, suppose generalization applied to entity sets customer and employee leads to a higher-level entity set person. The generalization is overlapping if an employee can also be a customer. Lower-level entity overlap is the default case; a disjointness constraint must be placed explicitly on a generalization or specialization. We can note a disjointedness constraint in an E-R diagram by adding the word disjoint next to the triangle symbol. This constraint may be one of the following: Each higher-level entity must belong to a lower-level entity set. Some higher-level entities may not belong to any lower-level entity set. Partial generalization is the default. We can specify total generalization in an E-R diagram by using a double line to connect the box representing the higher-level entity set to the triangle symbol. This notation is similar to the notation for total participation in a relationship. The account generalization is total: All account entities must be either a savings account or a checking account. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total. When the generalization is partial, a higher-level entity is not constrained to appear in a lower-level entity set. The work team entity sets illustrate a partial specialization. Since employees are assigned to a team only after 3 months on the job, some employee entities may not be members of any of the lower-level team entity sets. We may characterize the team entity sets more fully as a partial, overlapping specialization of employee. The generalization of checking-account and savings-account into account is a total, disjoint generalization. The completeness and disjointness constraints, however, do not depend on each other. Constraint patterns may also be partial-disjoint and total-overlapping. We can see that certain insertion and deletion requirements follow from the constraints that apply to a given generalization or specialization. For instance, when a total completeness constraint is in place, an entity inserted into a higher-level entity set must also be inserted into at least one of the lower-level entity sets. Finally, an entity that is deleted from a higher-level entity set also is deleted from all the associated lower-level entity sets to which it belongs. Aggregation One limitation of the E-R model is that it cannot express relationships among relationships. To illustrate the need for such a construct, consider the ternary relationship works-on, which we saw earlier, between a employee, branch, and job see Figure 2. Now, suppose we want to record managers for tasks performed by an employee at a branch; that is, we want to record managers for employee, branch, job combinations. Let us assume that there is an entity set manager. One alternative for representing this relationship is to create a quaternary relation- ship manages between employee, branch, job, and manager. A quaternary relationship is required â€" a binary relationship between manager and employee would not permit us to represent which branch, job combinations of an employee are managed by which manager. We have omitted the attributes of the entity sets, for simplicity. It appears that the relationship sets works-on and manages can be combined into one single relationship set. Nevertheless, we should not combine them into a single relationship, since some employee, branch, job combinations many not have a manager. If the manager were a value rather than an manager entity, we could instead make manager a multivalued at- tribute of the relationship works-on.

Chapter 8 : Entityâ€“relationship model - Wikipedia

to draw an Er-diagram you need >> objects >> relationship between these objects >>database associated with each object and relationship er digram is a graphical representatio n of a database.

Chapter 9 : JNTUH theinnatdunvilla.com Database Management Systems R09 Syllabus | theinnatdunvilla.

Let us now learn how the ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram. Attributes are the properties of entities. Attributes are.