Lecture No. 14

Reading Material

"Database Systems Principles, Design and Implementation" written by Catherine Ricardo, Maxwell Macmillan.	Section 6.1 – 6.3.3
"Database Management Systems", 2nd edition, Raghu Ramakrishnan, Johannes Gehrke, McGraw-Hill	

Overview of Lecture

- Logical Database Design
- Introduction to Relational Data Model
- Basic properties of a table
- Mathematical and database relations

From this lecture we are going to discuss the logical database design phase of database development process. Logical database design, like conceptual database design is our database design; it represents the structure of data that we need to store to fulfill the requirements of the users or organization for which we are developing the system. However there are certain differences between the two that are presented in the table below:

	Conceptual Database Design	**Logical Database Design**
1	Developed in a semantic data model (generally E-R data model)	In legacy data models (relational generally in current age)
2	Free of data model in which going to be implemented; many/any possible	Free of particular DBMS in which going to be implemented; many/any possible
3	Results from Analysis Phase	Obtained by translating the conceptual database design into another data model

© Copyright Virtual University of Pakistan

4	Represented graphically	Descriptive
5	More expressive	Relatively less expressive
6	Going to be transformed and then implemented	Going to be implemented
You can think more, give a try		

Table 1: Differences between Conceptual and Logical Database Designs

As we have already discussed in previous lectures and as is given in row 2 of the above table, the conceptual database design can be transformed into any data model, like, hierarchical, network, relational or object-oriented. So the study of the logical database design requires first involves the study of the data model/(s) that we can possibly use for the purpose. However, in the current age, since early eighties, the most popular choice for the logical database design is the relational data model; so much popular that today it can be considered to be the only choice. Why? Because of its features we are going to discuss in today's lecture. That is why rather than studying different data models we will be studying only the relational data model. Once we study this, the development of logical database design is transformation of conceptual database design to relational one and the process is very simple and straightforward. So from today's lecture our discussion starts on the relational data model. Just for the sake of revision we repeat the definition of data model "a set of constructs/tools used to develop a database design; generally consists of three components which are constructs, manipulation language and integrity constraints". We have discussed it earlier that the later part of the definition (three components) fits precisely with the relational data model (RDM), that is, it has these components defined clearly.

Relational Data Model

The RDM is popular due to its two major strengths and they are:

- o Simplicity
- o Strong Mathematical Foundation

The RDM is simple, why, there is just one structure and that is a relation or a table. Even this single structure is very easy to understand, so a user of even of a moderate

© Copyright Virtual University of Pakistan

genius can understand it easily. Secondly, it has a strong mathematical foundation that gives many advantages, like:

- Anything included/defined in RDM has got a precise meaning since it is based on mathematics, so there is no confusion.
- If we want to test something regarding RDM we can test it mathematically, if it works mathematically it will work with RDM (apart from some exceptions).
- The mathematics not only provided the RDM the structure (relation) but also well defined manipulation languages (relational algebra and relational calculus).
- It provided RDM certain boundaries, so any modification or addition we want to make in RDM, we have to see if it complies with the relational mathematics or not. We cannot afford to cross these boundaries since we will be losing the huge advantages provided by the mathematical backup.

"An IBM scientist E.F. Codd proposed the relational data model in 1970. At that time most database systems were based on one of two older data models (the hierarchical model and the network model); the relational model revolutionized the database field and largely replaced these earlier models. Prototype relational database management systems were developed in pioneering research projects at IBM and UC-Berkeley by the mid-70s, and several vendors were offering relational database products shortly thereafter. Today, the relational model is by far the dominant data model and is the foundation for the leading DBMS products, including IBM's DB2 family, Informix, Oracle, Sybase, Microsoft's Access and SQLServer, FoxBase, and Paradox. Relational database systems are ubiquitous in the marketplace and represent a multibillion dollar industry" [1]

The RDM is mainly used for designing/defining external and conceptual schemas; however to some extent physical schema is also specified in it. Separation of conceptual and physical levels makes data and schema manipulation much easier, contrary to previous data models. So the relational data model also truly supports "Three Level Schema Architecture".

Introduction to the Relational Data model

The RDM is based on a single structure and that is a relation. Speaking in terms of the E-R data model, both the entity types and relationships are represented using relations

© Copyright Virtual University of Pakistan

in RDM. The relation in RDM is similar to the mathematical relation however database relation is also represented in a two dimensional structure called table. A table consists of rows and columns. Rows of a table are also called tuples. A row or tuple of a table represents a record or an entity instance, where as the columns of the table represent the properties or attributes.

stID	stName	clName	doB	sex
S001	M. Suhail	MCS	12/6/84	M
S002	M. Shahid	BCS	3/9/86	M
S003	Naila S.	MCS	7/8/85	F
S004	Rubab A.	MBA	23/4/86	F
S005	Ehsan M.	BBA	22/7/88	M

Table 2: A database relation represented in the form of a table

In the above diagram, a table is shown that consists of five rows and five columns. The top most rows contain the names of the columns or attributes whereas the rows represent the records or entity instances. There are six basic properties of the database relations which are:

- Each cell of a table contains atomic/single value

  A cell is the intersection of a row and a column, so it represents a value of an attribute in a particular row. The property means that the value stored in a single cell is considered as a single value. In real life we see many situations when a property/attribute of any entity contains multiple values, like, degrees that a person has, hobbies of a student, the cars owned by a person, the jobs of an employee. All these attributes have multiple values; these values cannot be placed as the value of a single attribute or in a cell of the table. It does not mean that the RDM cannot handle such situations, however, there are some special means that we have to adopt in these situations, and they can not be placed as the value of an attribute because an attribute can contain only a single value. The values of attributes shown in table 1 are all atomic or single.

© Copyright Virtual University of Pakistan

- Each column has a distinct name; the name of the attribute it represents

  Each column has a heading that is basically the name of the attribute that the column represents. It has to be unique, that is, a table cannot have duplicated column/attribute names. In the table 2 above, the bold items in the first row represent the column/attribute names.

- The values of the attributes come from the same domain

  Each attribute is assigned a domain along with the name when it is defined. The domain represents the set of possible values that an attribute can have. Once the domain has been assigned to an attribute, then all the rows that are added into the table will have the values from the same domain for that particular column. For example, in the table 2 shown above the attribute doB (date of birth) is assigned the domain “Date”, now all the rows have the date value against the attribute doB. This attribute cannot have a text or numeric value.

- The order of the columns is immaterial
  If the order of the columns in a table is changed, the table still remains the same. Order of the columns does not matter.

- The order of the rows is immaterial

  As with the columns, if rows’ order is changed the table remains the same.

- Each row/tuple/record is distinct, no two rows can be same

  Two rows of a table cannot be same. The value of even a single attribute has to be different that makes the entire row distinct.

There are three components of the RDM, which are, construct (relation), manipulation language (SQL) and integrity constraints (two). We have discussed the relation so far; the last two components will be discussed later. In the next section we are going to

© Copyright Virtual University of Pakistan

discuss the mathematical relations briefly that will help to link the mathematical relations with the database relations and will help in a better understanding of the later.

Mathematical Relations

Consider two sets

A = {x, y} B = {2, 4, 6}

Cartesian product of these sets (A x B) is a set that consists of ordered pairs where first element of the ordered pair belongs to set A where as second element belongs to set B, as shown below:

A X B= {(x,2), (x,4), (x,6), (y,2), (y,4), (y,6)}

A relation is some subset of this Cartesian product, For example,

- R1= {(x,2), (y,2),(x,6),(x,4)}
- R2 = {(x,4), (y,6), (y,4)}

The same notion of Cartesian product and relations can be applied to more than two sets, e.g. in case of three sets, we will have a relation of ordered triplets

Applying the same concept in a real world scenario, consider two sets Name and Age having the elements:

- Name = {Ali, Sana, Ahmed, Sara}
- Age = {15,16,17,18,…….,25}

Cartesian product of Name & Age

Name X Age= {(Ali,15), (Sana,15), (Ahmed,15), (Sara,15), …., (Ahmed,25), (Sara,25)}

Now consider a subset CLASS of this Cartesian product

CLASS = {(Ali, 18), (Sana, 17), (Ali, 20), (Ahmed, 19)}

This subset CLASS is a relation mathematically, however, it may represent a class in the real world where each ordered pair represents a particular student mentioning the name and age of a student. In the database context each ordered pair represents a tuple and elements in the ordered pairs represent values of the attributes. Think in this way, if Name and Age represent all possible values for names and ages of students, then any class you consider that will definitely be a subset of the Cartesian product of the Name and Age. That is, the name and age combination of all the students of any class

© Copyright Virtual University of Pakistan

will be included in the Cartesian product and if we take out particulars ordered pairs that are related to a class then that will be a subset of the Cartesian product, a relation.

Database Relations

Let A1, A2, A3, …, An be some attributes and D1, D2, D3,…, Dn be their domains A relation scheme relates certain attributes with their domain in context of a relation. A relation scheme can be represented as:

R = (A1:D1, A2:D2, ……, An:Dn), for example,

STD Scheme = (stId:Text, stName: Text, stAdres:Text, doB:Date) OR

STD(stId, stName, stAdres, doB)

Whereas the stId, stName, stAdres and doB are the attribute names and Text, Text, Text and Date are their respective domains. A database relation as per this relation scheme can be:

STD={(stId:S001, stName:Ali, stAdres: Lahore, doB:12/12/76), (stId:S003, stName:A. Rehman, stAdres: RWP, doB:2/12/77)} OR

STD={(S001, Ali, Lahore, 12/12/76), (S003, A. Rehman, RWP, 2/12/77)}

The above relation if represented in a two dimensional structure will be called a table as is shown below:

stId	stName	stAdres	doB
S001	Ali	Lahore	12/12/76
S002	A. Rehman	RWP	2/12/77

With this, today's lecture is finished; the discussion on RDM will be continued in the next lecture.

Summary

In this lecture we have started the discussion on the logical database design that we develop from the conceptual database design. The later is generally developed using E-R data model, whereas for the former RDM is used. RDM is based on the theory of mathematical relations; a mathematical relation is subset of the Cartesian product of two or more sets. Relations are physically represented in the form of two-dimensional

© Copyright Virtual University of Pakistan

structure called table, where rows/tuples represent records and columns represent the attributes.

Exercise:

Define different attributes (assigning name and domain to each) for an entity STUDENT, then apply the concept of Cartesian product on the domains of these attributes, then consider the records of your class fellows and see if it is the subset of the Cartesian product.

© Copyright Virtual University of Pakistan

Lecture No. 15

Reading Material

Overview of Lecture

- Database and Math Relations
- Degree and Cardinality of Relation
- Integrity Constraints
- Transforming conceptual database design into logical database design
- Composite and multi-valued Attributes
- Identifier Dependency

In the previous lecture we discussed relational data model, its components and properties of a table. We also discussed mathematical and database relations. Now we will discuss the difference in between database and mathematical relations.

Database and Math Relations

We studied six basic properties of tables or database relations. If we compare these properties with those of mathematical relations then we find out that properties of both are the same except the one related to order of the columns. The order of columns in mathematical relations does matter, whereas in database relations it does not matter. There will not be any change in either math or database relations if we change the rows or tuples of any relation. It means that the only difference in between these two is of order of columns or attributes. A math relation is a Cartesian product of two sets. So if we change the order of theses two sets then the outcome of both will not be same. Therefore, the math relation changes by changing the order of columns. For Example , if there is a set A and a set B if we take Cartesian product of A and B then we take Cartesian product of B and A they will not be equal , so

$$\mathbf{A \times B \neq B \times A}$$

Rests of the properties between them are same.

© Copyright Virtual University of Pakistan

Degree of a Relation

We will now discuss the degree of a relation not to be confused with the degree of a relationship. You would be definitely remembering that the relationship is a link or association between one or more entity types and we discussed it in E-R data model. However the degree of a relation is the number of columns in that relation. For Example consider the table given below:

STUDENT

StID	**stName**	**clName**	**Sex**
S001	Suhail	MCS	M
S002	Shahid	BCS	M
S003	Naila	MCS	F
S004	Rubab	MBA	F
S005	Ehsan	BBA	M

Table 1: The STUDENT table

Now in this example the relation STUDENT has four columns, so this relation has degree four.
Cardinality of a Relation
The number of rows present in a relation is called as cardinality of that relation. For example, in STUDENT table above, the number of rows is five, so the cardinality of the relation is five.

Relation Keys

The concept of key and all different types of keys is applicable to relations as well. We will now discuss the concept of foreign key in detail, which will be used quite frequently in the RDM.

Foreign Key
An attribute of a table B that is primary key in another table A is called as foreign key. For Example, consider the following two tables EMP and DEPT:

EMP (empId, empName, qual, depId)
DEPT (depId, depName, numEmp)

In this example there are two relations; EMP is having record of employees, whereas DEPT is having record of different departments of an organization. Now in EMP the primary key is empId, whereas in DEPT the primary key is depId. The depId which is primary key of DEPT is also present in EMP so this is a foreign key.

Requirements/Constraints of Foreign Key
Following are some requirements / constraints of foreign key:
There can be more than zero, one or multiple foreign keys in a table, depending on how many tables a particular table is related with. For example in the above example the EMP table is related with the DEPT table, so there is one foreign key depId,

© Copyright Virtual University of Pakistan

whereas DEPT table does not contain any foreign key. Similarly, the EMP table may also be linked with DESIG table storing designations, in that case EMP will have another foreign key and alike.
The foreign key attribute, which is present as a primary key in another relation is called as home relation of foreign key attribute, so in EMP table the depId is foreign key and its home relation is DEPT.
The foreign key attribute and the one present in another relation as primary key can have different names, but both must have same domains. In DEPT, EMP example, both the PK and FK have the same name; they could have been different, it would not have made any difference however they must have the same domain.

The primary key is represented by underlining with a solid line, whereas foreign key is underlined by dashed or dotted line.
Primary Key : ———
Foreign Key : _ _ _ _ _ _ _
Integrity Constraints
Integrity constraints are very important and they play a vital role in relational data model. They are one of the three components of relational data model. These constraints are basic form of constraints, so basic that they are a part of the data model, due to this fact every DBMS that is based on the RDM must support them.

Entity Integrity Constraint:
It states that in a relation no attribute of a primary key (PK) can have null value. If a PK consists of single attribute, this constraint obviously applies on this attribute, so it cannot have the Null value. However, if a PK consists of multiple attributes, then none of the attributes of this PK can have the Null value in any of the instances.

Referential Integrity Constraint:
This constraint is applied to foreign keys. Foreign key is an attribute or attribute combination of a relation that is the primary key of another relation. This constraint states that if a foreign key exists in a relation, either the foreign key value must match the primary key value of some tuple in its home relation or the foreign key value must be completely null.

Significance of Constraints:
By definition a PK is a minimal identifier that is used to identify tuples uniquely. This means that no subset of the primary key is sufficient to provide unique identification of tuples. If we were to allow a null value for any part of the primary key, we would be demonstrating that not all of the attributes are needed to distinguish between tuples, which would contradict the definition.
Referential integrity constraint plays a vital role in maintaining the correctness, validity or integrity of the database. This means that when we have to ensure the proper enforcement of the referential integrity constraint to ensure the consistency and correctness of database. How? In the DEPT, EMP example above deptId in EMP is foreign key; this is being used as a link between the two tables. Now in every instance of EMP table the attribute deptId will have a value, this value will be used to get the name and other details of the department in which a particular employee works. If the value of deptId in EMP is Null in a row or tuple, it means this particular row is not related with any instance of the DEPT. From real-world scenario it means that this particular employee (whose is being represented by this row/tuple) has not been

© Copyright Virtual University of Pakistan

assigned any department or his/her department has not been specified. These were two possible conditions that are being reflected by a legal value or Null value of the foreign key attribute. Now consider the situation when referential integrity constrains is being violated, that is, EMP.deptId contains a value that does not match with any of the value of DEPT.deptId. In this situation, if we want to know the department of an employee, then ooops, there is no department with this Id, that means, an employee has been assigned a department that does not exist in the organization or an illegal department. A wrong situation, not wanted. This is the significance of the integrity constraints.

Null Constraints:
A Null value of an attribute means that the value of attribute is not yet given, not defined yet. It can be assigned or defined later however. Through Null constraint we can monitor whether an attribute can have Null value or not. This is important and we have to make careful use of this constraint. This constraint is included in the definition of a table (or an attribute more precisely). By default a non-key attribute can have Null value, however, if we declare an attribute as Not Null, then this attribute must be assigned value while entering a record/tuple into the table containing that attribute. The question is, how do we apply or when do we apply this constraint, or why and when, on what basis we declare an attribute Null or Not Null. The answer is, from the system for which we are developing the database; it is generally an organizational constraint. For example, in a bank, a potential customer has to fill in a form that may comprise of many entries, but some of them would be necessary to fill in, like, the residential address, or the national Id card number. There may be some entries that may be optional, like fax number. When defining a database system for such a bank, if we create a CLIENT table then we will declare the must attributes as Not Null, so that a record cannot be successfully entered into the table until at least those attributes are not specified.

Default Value:
This constraint means that if we do not give any value to any particular attribute, it will be given a certain (default) value. This constraint is generally used for the efficiency purpose in the data entry process. Sometimes an attribute has a certain value that is assigned to it in most of the cases. For example, while entering data for the students, one attribute holds the current semester of the student. The value of this attribute is changed as a students passes through different exams or semesters during its degree. However, when a student is registered for the first time, it is generally registered in the first semesters. So in the new records the value of current semester attribute is generally 1. Rather than expecting the person entering the data to enter 1 in every record, we can place a default value of 1 for this attribute. So the person can simply skip the attribute and the attribute will automatically assume its default value.

Domain Constraint:
This is an essential constraint that is applied on every attribute, that is, every attribute has got a domain. Domain means the possible set of values that an attribute can have. For example, some attributes may have numeric values, like salary, age, marks etc. Some attributes may possess text or character values, like, name and address. Yet some others may have the date type value, like date of birth, joining date. Domain specification limits an attribute the nature of values that it can have. Domain is specified by associating a data type to an attribute while defining it. Exact data type name or specification depends on the particular tool that is being used. Domain helps

© Copyright Virtual University of Pakistan

to maintain the integrity of the data by allowing only legal type of values to an attribute. For example, if the age attribute has been assigned a numeric data type then it will not be possible to assign a text or date value to it. As a database designer, this is your job to assign an appropriate data type to an attribute. Another perspective that needs to be considered is that the value assigned to attributes should be stored efficiently. That is, domain should not allocate unnecessary large space for the attribute. For example, age has to be numeric, but then there are different types of numeric data types supported by different tools that permit different range of values and hence require different storage space. Some of more frequently supported numeric data types include Byte, Integer, and Long Integer. Each of these types supports different range of numeric values and takes 1, 4 or 8 bytes to store. Now, if we declare the age attribute as Long Integer, it will definitely serve the purpose, but we will be allocating unnecessarily large space for each attribute. A Byte type would have been sufficient for this purpose since you won't find students or employees of age more than 255, the upper limit supported by Byte data type. Rather we can further restrict the domain of an attribute by applying a check constraint on the attribute. For example, the age attribute although assigned type Byte, still if a person by mistake enters the age of a student as 200, if this is year then it is not a legal age from today's age, yet it is legal from the domain constraint perspective. So we can limit the range supported by a domain by applying the check constraint by limiting it up to say 30 or 40, whatever is the rule of the organization. At the same time, don't be too sensitive about storage efficiency, since attribute domains should be large enough to cater the future enhancement in the possible set of values. So domain should be a bit larger than that is required today. In short, domain is also a very useful constraint and we should use it carefully as per the situation and requirements in the organization.

RDM Components
We have up till now studied following two components of the RDM, which are the Structure and Entity Integrity Constraints. The third part, that is, the Manipulation Language will be discussed in length in the coming lectures.

Designing Logical Database
Logical data base design is obtained from conceptual database design. We have seen that initially we studied the whole system through different means. Then we identified different entities, their attributes and relationship in between them. Then with the help of E-R data model we achieved an E-R diagram through different tools available in this model. This model is semantically rich. This is our conceptual database design. Then as we had to use relational data model so then we came to implementation phase for designing logical database through relational data model.

The process of converting conceptual database into logical database involves transformation of E-R data model into relational data model. We have studied both the data models, now we will see how to perform this transformation.

Transforming Rules

Following are the transforming rules for converting conceptual database into logical database design:
The rules are straightforward , which means that we just have to follow the rules mentioned and the required logical database design would be achieved

© Copyright Virtual University of Pakistan

There are two ways of transforming first one is manually that is we analyze and evaluate and then transform. Second is that we have CASE tools available with us which can automatically convert conceptual database into required logical database design
If we are using CASE tools for transforming then we must evaluate it as there are multiple options available and we must make necessary changes if required.

Mapping Entity Types
Following are the rules for mapping entity types:
Each regular entity type (ET) is transformed straightaway into a relation. It means that whatever entities we had identified they would simply be converted into a relation and will have the same name of relation as kept earlier.
Primary key of the entity is declared as Primary key of relation and underlined.
Simple attributes of ET are included into the relation

For Example, figure 1 below shows the conversion of a strong entity type into equivalent relation:

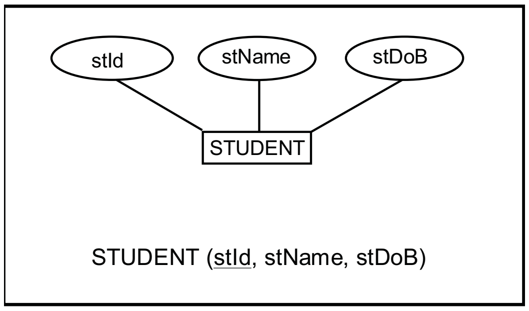
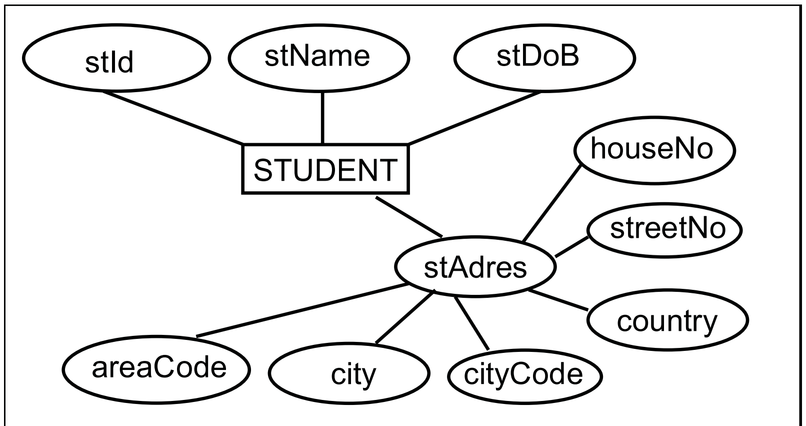


Fig. 1: An example strong entity type

Composite Attributes
These are those attributes which are a combination of two or more than two attributes. For address can be a composite attribute as it can have house no, street no, city code and country , similarly name can be a combination of first and last names. Now in relational data model composite attributes are treated differently. Since tables can contain only atomic values composite attributes need to be represented as a separate relation
For Example in student entity type there is a composite attribute Address, now in E-R model it can be represented with simple attributes but here in relational data model, there is a requirement of another relation like following:

© Copyright Virtual University of Pakistan



STUDENT (<u>stId</u>, stName, stDoB)
STDADRES (<u>stId</u>, hNo, strNo, country, cityCode, city, areaCode)

Fig. 2: Transformation of composite attribute

Figure 2 above presents an example of transforming a composite attribute into RDM, where it is transformed into a table that is linked with the STUDENT table with the primary key
Multi-valued Attributes

These are those attributes which can have more than one value against an attribute. For Example a student can have more than one hobby like riding, reading listening to music etc. So these attributes are treated differently in relational data model. Following are the rules for multi-valued attributes:-

An Entity type with a multi-valued attribute is transformed into two relations
One contains the entity type and other simple attributes whereas the second one has the multi-valued attribute. In this way only single atomic value is stored against every attribute
The Primary key of the second relation is the primary key of first relation and the attribute value itself. So in the second relation the primary key is the combination of two attributes.

© Copyright Virtual University of Pakistan

All values are accessed through reference of the primary key that also serves as

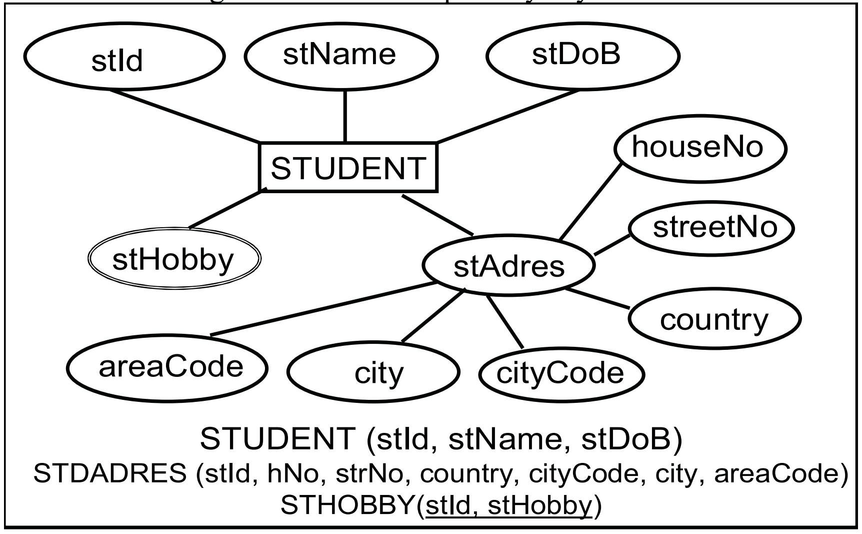


Fig. 3: Transformation of multi-valued attribute

foreign key.

Conclusion

In this lecture we have studied the difference between mathematical and database relations. The concepts of foreign key and especially the integrity constraints are very important and are basic for every database. Then how a conceptual database is transformed into logical database and in our case it is relational data model as it is the most widely used. We have also studied certain transforming rules for converting E-R data model into relational data model. Some other rule for this transformation will be studied in the coming lectures

You will receive exercise at the end of this topic.

© Copyright Virtual University of Pakistan

Lecture No. 16

Reading Material

"Database Systems Principles, Design and Implementation" written by Catherine Ricardo, Maxwell Macmillan.	Page 209

Overview of Lecture:

- o Mapping Relationships
- o Binary Relationships
- o Unary Relationships
- o Data Manipulation Languages

In the previous lecture we discussed the integrity constraints. How conceptual database is converted into logical database design, composite and multi-valued attributes. In this lecture we will discuss different mapping relationships.

Mapping Relationships

We have up till now converted an entity type and its attributes into RDM. Before establishing any relationship in between different relations, it is must to study the cardinality and degree of the relationship. There is a difference in between relation and relationship. Relation is a structure, which is obtained by converting an entity type in E-R model into a relation, whereas a relationship is in between two relations of relational data model. Relationships in relational data model are mapped according to their degree and cardinalities. It means before establishing a relationship there cardinality and degree is important.

Binary Relationships

Binary relationships are those, which are established between two entity type. Following are the three types of cardinalities for binary relationships:

© Copyright Virtual University of Pakistan

- One to One
- One to Many
- Many to Many

In the following treatment in each of these situations is discussed.

One to Many:

In this type of cardinality one instance of a relation or entity type is mapped with many instances of second entity type, and inversely one instance of second entity type is mapped with one instance of first entity type. The participating entity types will be transformed into relations as has been already discussed. The relationship in this particular case will be implemented by placing the PK of the entity type (or corresponding relation) against one side of relationship will be included in the entity type (or corresponding relation) on the many side of the relationship as foreign key (FK). By declaring the PK-FK link between the two relations the referential integrity constraint is implemented automatically, which means that value of foreign key is either null or matches with its value in the home relation.

For Example, consider the binary relationship given in the figure 1 involving two entity types PROJET and EMPLOYEE. Now there is a one to many relationships between these two. On any one project many employees can work and one employee can work on only one project.

© Copyright Virtual University of Pakistan

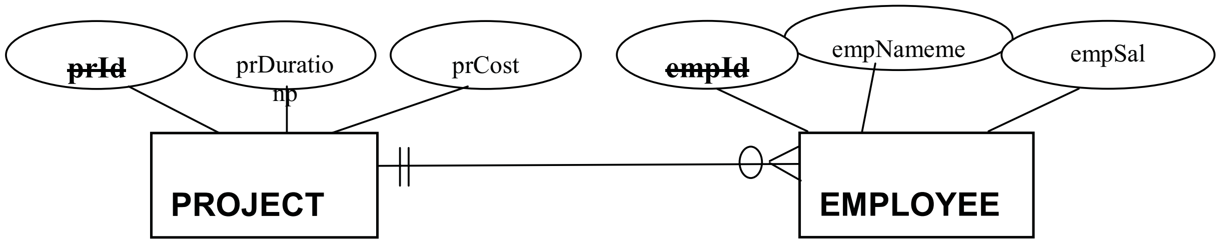


Fig. 1: A one to many relationship

The two participating entity types are transformed into relations and the relationship is implemented by including the PK of PROJECT (prId) into the EMPLOYEE as FK. So the transformation will be:

PROJECT (prId, prDura, prCost)
EMPLOYEE (empId, empName, empSal, prId)

The PK of the PROJECT has been included in EMPLOYEE as FK; both keys do not need to have same name, but they must have the same domain.

Minimum Cardinality:
This is a very important point, as minimum cardinality on one side needs special attention. Like in previous example an employee cannot exist if project is not assigned. So in that case the minimum cardinality has to be one. On the other hand if an instance of EMPLOYEE can exist with out being linked with an instance of the PROJECT then the minimum cardinality has to be zero. If the minimum cardinality is zero, then the FK is defined as normal and it can have the Null value, on the other hand if it is one then we have to declare the FK attribute(s) as Not Null. The Not Null constraint makes it a must to enter the value in the attribute(s) whereas the FK constraint will enforce the value to be a legal one. So you have to see the minimum cardinality while implementing a one to many relationship.

Many to Many Relationship:
In this type of relationship one instance of first entity can be mapped with many instances of second entity. Similarly one instance of second entity can be mapped with many instances of first entity type. In many to many relationship a third table is created for the relationship, which is also called as associative entity type. Generally,

© Copyright Virtual University of Pakistan

the primary keys of the participating entity types are used as primary key of the third table.

For Example, there are two entity types BOOK and STD (student). Now many students can borrow a book and similarly many books can be issued to a student, so in this manner there is a many to many relationship. Now there would be a third relation as well which will have its primary key after combining primary keys of BOOK and STD. We have named that as transaction TRANS. Following are the attributes of these relations: -

- STD (stId, sName, sFname)
- BOOK (bkId, bkTitle, bkAuth)
- TRANS (stId,bkId, isDate,rtDate)

Now here the third relation TRANS has four attributes first two are the primary keys of two entities whereas the last two are issue date and return date.

One to One Relationship:

This is a special form of one to many relationship, in which one instance of first entity type is mapped with one instance of second entity type and also the other way round. In this relationship primary key of one entity type has to be included on other as foreign key. Normally primary key of compulsory side is included in the optional side. For example, there are two entities STD and STAPPLE (student application for scholarship). Now the relationship from STD to STAPPLE is optional whereas STAPPLE to STD is compulsory. That means every instance of STAPPLE must be related with one instance of STD, whereas it is not a must for an instance of STD to be related to an instance of STAPPLE, however, if it is related then it will be related to one instance of STAPPLE, that is, one student can give just one scholarship application. This relationship is shown in the figure below:

© Copyright Virtual University of Pakistan

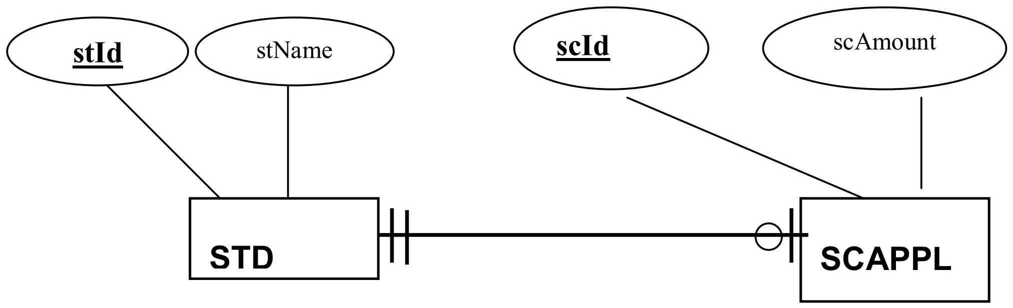


Fig. 2: A one to one relationship

While transforming, two relations will be created, one for STD and HOBBY each. For relationship PK of either one can be included in the other, it will work. But preferably, we should include the PK of STD in HOBBY as FK with Not Null constraint imposed on it.

STD (<u>stId</u>, stName)

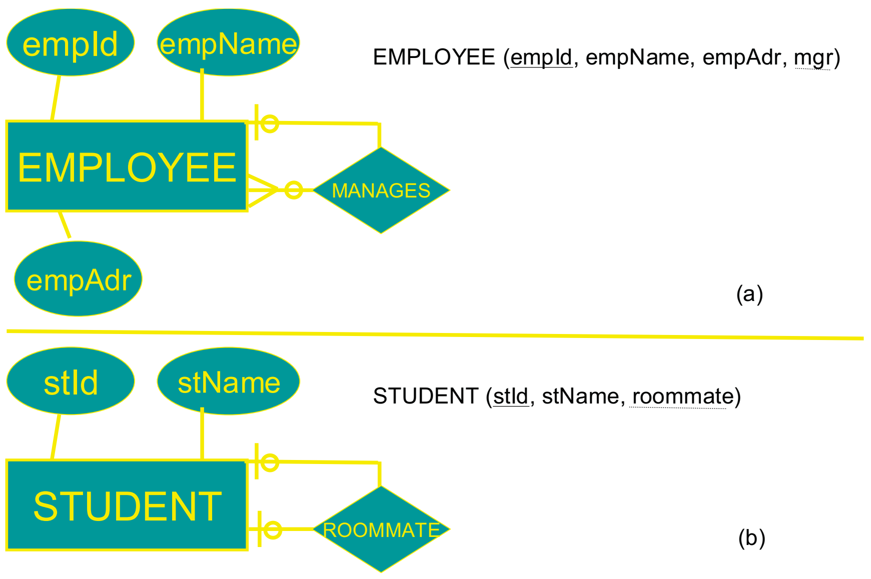
STAPPLE (<u>scId</u>, scAmount, stId)

The advantage of including the PK of STD in STAPPLE as FK is that any instance of STAPPLE will definitely have a value in the FK attribute, that is, stId. Whereas if we do other way round; we include the PK of STAPPLE in STD as FK, then since the relationship is optional from STD side, the instances of STD may have Null value in the FK attribute (scId), causing the wastage of storage. More the number records with Null value more wastage.

Unary Relationship

These are the relationships, which involve a single entity. These are also called recursive relationships. Unary relationships may have one to one, one to many and many to many cardinalities. In unary one to one and one to may relationships, the PK of same entity type is used as foreign key in the same relation and obviously with the different name since same attribute name cannot be used in the same table. The example of one to one relationship is shown in the figure below:

© Copyright Virtual University of Pakistan

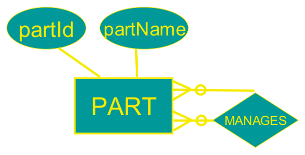


EMPLOYEE (empId, empName, empAdr, mgr)

STUDENT (stId, stName, roommate)

Fig. 3: One to one relationships (a) one to many (b) one to one and their transformation

In many to many relationships another relation is created with composite key. For example there is an entity type PART may have many to many recursive relationships, meaning one part consists of many parts and one part may be used in many parts. So in this case this is a many to many relationship. The treatment of such a relationship is shown in the figure below:



PART (partId, partName)
SUB-PART (partId, component)

Fig. 4: Recursive many to many relationship and transformation

Super / Subtype Relationship:
Separate relations are created for each super type and subtypes. It means if there is one super type and there are three subtypes, so then four relations are to be created. After creating these relations then attributes are assigned. Common attributes are assigned to super type and specialized attributes are assigned to concerned subtypes. Primary key of super type is included in all relations that work for both link and

© Copyright Virtual University of Pakistan

identity. Now to link the super type with concerned subtype there is a requirement of descriptive attribute, which is called as discriminator. It is used to identify which subtype is to be linked. For Example there is an entity type EMP which is a super type, now there are three subtypes, which are salaried, hourly and consultants. So now there is a requirement of a determinant, which can identify that which subtypes to be consulted, so with empId a special character can be added which can be used to identify the concerned subtype.

Summary of Mapping E-R Diagram to Relational DM:
We have up till now studied that how conceptual database design is converted into logical database. E-R data model is semantically rich and it has number of constructs for representing the whole system. Conceptual database is free of any data model, whereas logical database the required data model is chosen; in our case it is relational data model. First we identified the entity types, weak and strong entity types. Then we converted those entities into relations. After converting entities into relations then attributes are identified, different types of attributes are identified. Then relationships were made, in which cardinality and degree was identified. In ternary relationship, where three entities are involved, in this as well another relation is created to establish relationship among them. Then finally we had studied the super and sub types in which primary key of super type was used for both identity and link.

Data Manipulation Languages

This is the third component of relational data model. We have studied structure, which is the relation, integrity constraints both referential and entity integrity constraint. Data manipulation languages are used to carry out different operations like insertion, deletion and updation of data in database. Following are the two types of languages:

© Copyright Virtual University of Pakistan

Procedural Languages:
These are those languages in which what to do and how to do on the database is required. It means whatever operation is to be done on the database that has to be told that how to perform.

Non -Procedural Languages:
These are those languages in which only what to do is required, rest how to do is done by the manipulation language itself.

Structured query language (SQL) is the most widely language used for manipulation of data. But we will first study Relational Algebra and Relational Calculus, which are procedural and non – procedural respectively.

Relational Algebra

Following are few major properties of relational algebra:

- **Relational algebra operations work on one or more relations to define another relation leaving the original intact. It means that the input for relational algebra can be one or more relations and the output would be another relation, but the original participating relations will remain unchanged and intact.** Both operands and results are relations, so output from one operation can become input to another operation. It means that the input and output both are relations so they can be used iteratively in different requirements.
- Allows expressions to be nested, just as in arithmetic. This property is called closure.
- There are five basic operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference.
- These perform most of the data retrieval operations needed.
- It also has Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.

Exercise:

- Consider the example given in Ricardo book on page 216 and transform it into relational data model. Make any necessary assumptions if required.

© Copyright Virtual University of Pakistan