Chapter I
Mapping Generalizations and Specializations and Categories to Relational Databases

Sikha Bagui
University of West Florida, USA

INTRODUCTION

An Entity Relationship (ER) model that includes all the concepts of the original ER model and the additional concepts of generalizations/specializations and categories is often referred to as the Extended ER (EER) model (Elmasri & Navathe, 2007). With the rising complexity of database applications, and also in light of today's web data applications (Necasky, 2006), the basic concepts of the ER model, as originally developed by Chen(1976), are no longer sufficient. Hence the basic ER model has been extended to include generalizations and specializations (Bagui & Earp, 2003; Elmasri & Navathe, 2007), and the concept of categories (Elmasri, et al., 1985). In this short article we shed some light on these relationship concepts, concepts that database designers often find difficult to directly model (Engels et al., 1992/93). We also discuss the mapping rules for generalizations/specializations and categories. Important contributions in this area are also reported in (Elmasri et al., 1985; Gogolla & Hohenstein, 1991; Markowitz & Shoshani, 1992; Dey, et. al., 1999). Dullea, et. al. (2003) discusses the structural validity of modeling structures with ER models.

Due to the short nature of this paper, we will keep the discussion in this paper focused on implementing generalizations and specializations in relational databases; their parallel implementation in objects will not be covered. Also, the discussion of the concept of inheritance will center around generalizations/specializations and categories in EER diagrams, without getting into an almost equivalent notion in Object-oriented (OO) theory, ORM (Object-Role Modeling) and UML (Unified Modeling Language) class diagrams.

BACKGROUND

A generalization/specialization relationship models a superclass/subclass type of relationship. A generalization is an abstraction mechanism that allows for viewing of entity-sets as a single generic entity-set. The attributes and associations which are common to the entity-sets are associated with the generic (generalized) entity-set. The inverse of generalization is called specialization.

GENERALIZATION / SPECIALIZATION RELATIONSHIPS

If we are modeling a hospital database, for example, and we want to store information about the hospital's nurses, technicians, and physician assistants, we could create separate entities such as NURSE, TECHNICIAN and PHYSICIAN ASSISTANT. But, these three entities would also have a lot of fields in common, for example, name, social security number, address, phone, etc. may be common to all three entities. So, it would be a good idea to have an entity set called EMPLOYEE containing these common fields, and entity subsets, NURSE, TECHNICIAN and PHYSICIAN ASSISTANT, that could inherit this information from the EMPLOYEE entity set. In this case the EMPLOYEE entity set would be called the ***superclass***. This superclass is a ***generalization*** of the entity subsets, NURSE, TECHNICIAN and PHYSICIAN ASSISTANT. The NURSE, TECHNICIAN and PHYSICIAN ASSISTANT would be called the ***subclasses***. The subclasses are ***specializations*** of the superclass, EMPLOYEE, and inherit from the superclass. Several specializations can be defined for the same entity type (or superclass).

The subclass, denoted by a separate entity rectangle in the EER diagram, is considered to be a *part* of the superclass entity set, EMPLOYEE. Although it will have attributes that distinguish it from other subclasses, it is considered only a subset of the EMPLOYEE entity set. That is, all nurses are employees, but the reverse is not true - not all employees are nurses. Likewise, all technicians or physician assistants are employees, but all employees are not technicians or physician assistants.

Figure 1 shows this generalization/specialization example. We use Elmasri and Navathe's (2007) diagrammatic notations for the EER diagrams. The subset symbol, "$\subset$", indicates the direction of the superclass/subclass or parent-child, inheritance relationship. This superclass/subclass relationship is also often referred to as a ***IS-A*** or ***IS-PART-OF*** relationship (Sanders, 1995).

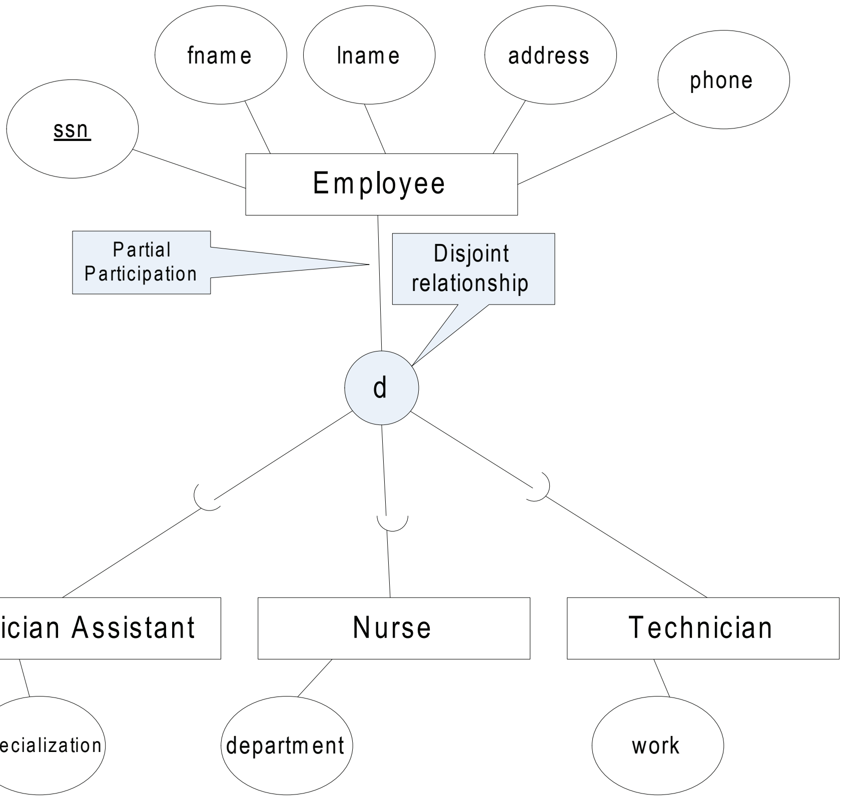
Constraints on Generalization/ Specialization Relationships

Generalizations and specializations can have two types of constraints: (i) the ***disjoint/overlap*** relationship constraint, and, (ii) participation constraints – total or partial. The combinations of these constraints can be: (i) disjoint and total participation; (ii) disjoint and partial participation; (iii) overlap and total participation; (iv) overlap and partial participation. First we will discuss disjoint/overlap relationship constraints and then we will discuss participation constraints, giving examples of combinations of the constraints along the way.

Disjoint/Overlap Relationship Constraints

Generalization/specialization relationships may be ***disjoint*** or they may ***overlap***. A disjoint relationship is shown by a "d" in the circle attaching the superclass to the subclass or subclasses (as shown in Figure 1). A disjoint relationship means that an entity from the superclass can belong to only one of the subclasses (can be of only one specialization). For example, according to figure 1, an EMPLOYEE can be at most a member of only one of the subclasses – PHYSICIAN ASSISTANT,

Figure 1. A generalization/specialization relationship



NURSE, or TECHNICIAN. An employee cannot be a physician assistant as well as a nurse, or, cannot be a nurse as well as a technician.

An overlap relationship is shown by an "o" in the circle attaching the superclass to the subclass or subclasses (as shown in Figure 4). Overlap means that an entity from the superclass can belong to more than one subclass (specialization). For example, according to Figure 4, a computer must be either a laptop or a desktop, or both a laptop and a desktop.

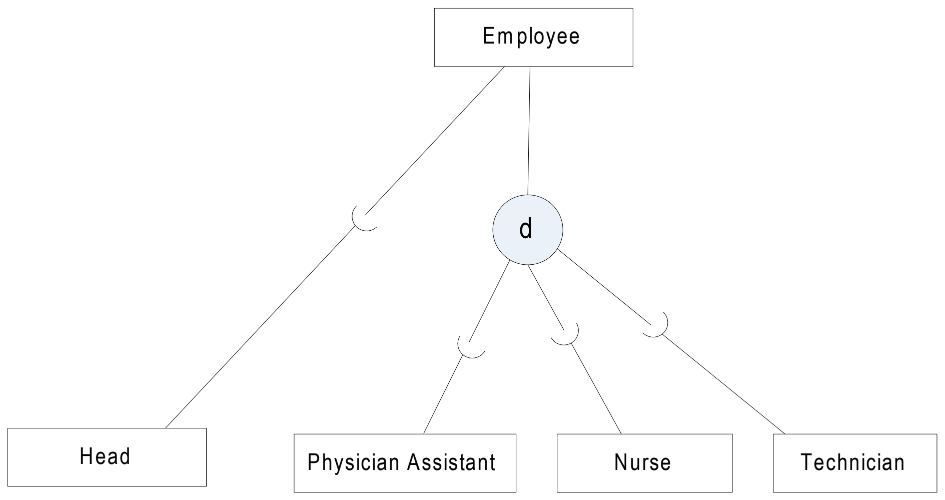
Participation Constraints

The second type of constraint on generalization/specialization relationships is participation constraints, which may be ***total*** (or full) or ***partial***. As in the ER model (Bagui & Earp, 2003; Elmasri & Navathe, 2007), we will show a full or total participation between the superclass and subclass entities by double lines, and a partial participation between the superclass and subclass entities by single lines. Partial participation is shown in Figure 1. Figure 1 can be read as:

An EMPLOYEE ***may*** *either be a PHYSICIAN ASSISTANT, NURSE or TECHNICIAN.*

Figure 1 shows a *disjoint, partial participation* relationship. The "may" means partial participation between the EMPLOYEE superclass and the respective subclasses entities. That is, not all employees of the EMPLOYEE entity set belong one of the subclasses, PHYSICIAN ASSISTANT, NURSE or TECHNICIAN. One may ask, why? Or how? To answer this, we will extend figure 1 to include another subclass, as shown in Figure 2. We now have an Employee from the EMPLOYEE

Figure 2. A disjoint



entity set who may also belong to the HEAD subclass. Once again, the "may" is inferred from the single line between the superclass (or generalization) entity, EMPLOYEE, and the subclass (or specialization) entity, HEAD. Figure 2 can be read as:

*An Employee **may** be a HEAD or PHYSICIAN ASSISTANT or NURSE or TECHNICIAN. Or, an Employee **may** be both a HEAD and a PHSYCIAN ASSISTANT, or both a HEAD and a NURSE, or both a HEAD and a TECHNICIAN.*

An example of total or full participation is shown in Figure 3. We can read Figure 3 as:

*An EMPLOYEE **must** either be a PHYSICIAN ASSISTANT, NURSE or TECHNICIAN.*

The "must" means total participation. So, an EMPLOYEE must belong to one of the subclasses. That is, ***all*** employees of the EMPLOYEE entity set must belong to one of the subclasses. But although there is total participation in Figure 3, the employee cannot belong to more than one subclass because of the "d" or disjoint relationship. Figure 3 shows a *disjoint, full participation* relationship and Figure 4 shows an *overlap, full participation* relationship.

Mapping Generalizations and Specializations to a Relational Database

Rules to map generalizations/specializations to a relational database depend on the constraints on the generalization/specialization relationships. One of the following four rules are generally used (Elmsari & Navathe, 2007) to map generalizations and specializations to a relational database:

Rule 1:

Rule 1 works well for total or partial generalization/specialization relationships as well as disjoint or overlap generalization/specialization relationships. Using this rule, we would create a separate relation for the superclass as well as for each of the subclasses.

For rule 1: (i) Create a relation for the superclass entity. Include all the attributes of this entity in this relation and underline the primary key attribute.

Figure 3. A disjoint and full participation with predicate defined subclasses

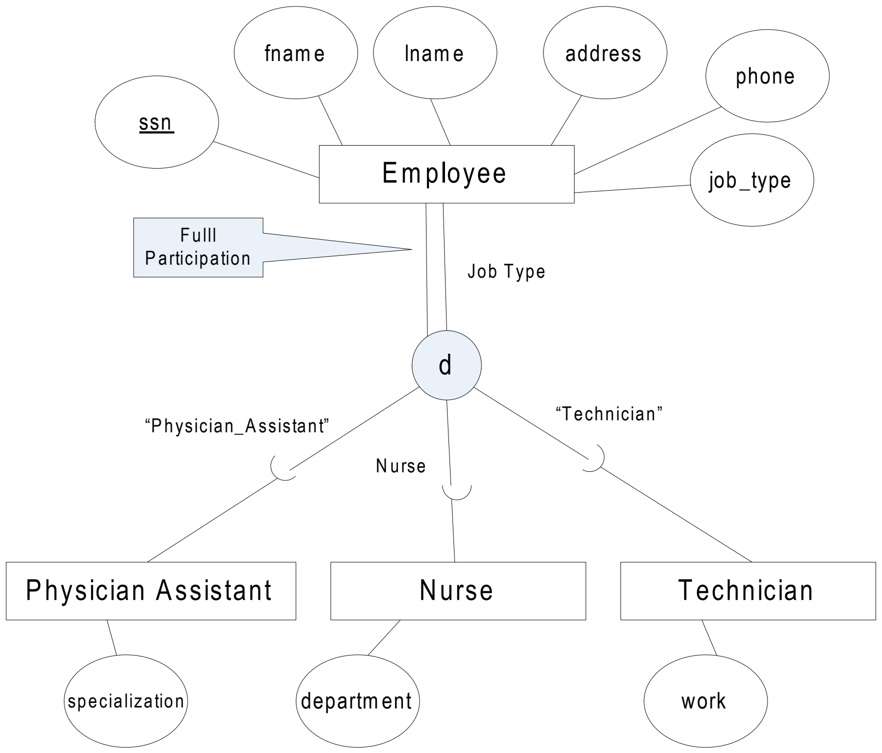
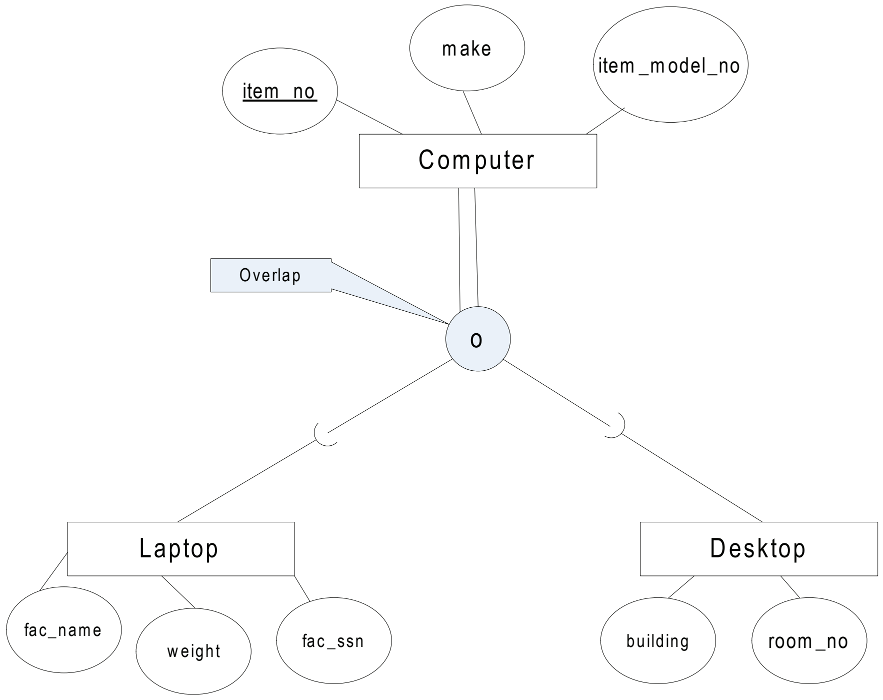


Figure 4. An overlap and full participation



(ii) Create a separate relation for each subclass (specialization) entity. Include the attributes of the respective subclasses in the respective subclass relations. Include the primary key from the superclass entity or relation in the subclass relation as the primary key of the subclass relation (and underline it).

To illustrate this rule we will map Figure 1 as shown below:

EMPLOYEE

ssn	fname	lname	address	phone

PHYSICIAN_ASSISTANT

ssn	specialization

NURSE

ssn	department

TECHNICIAN

ssn	work

Rule 2:

Rule 2 works well if: (a) there is total or full participation between the superclass and the subclasses. That is, if every member of the superclass entity set belongs to at least one of the subclasses; (b) if there is a disjoint relationship – otherwise there will be redundancy if a superclass entity belongs to more than one subclass; and, (c) when there is more emphasis on the subclass (specialization) and it is more important to know about the subclass and it's attributes. By this rule we create a separate relation for each subclass. In this rule you do not have a separate relation for the superclass entity.

For rule 2: Create a separate relation corresponding to each subclass entity. Include the attributes of the respective subclasses in the respective subclass relations. Also include the primary key and other attributes of the superclass entity in all the subclass relations. Underline the primary key brought from the superclass entity (to the subclass relation).

To illustrate this rule we will map figure 1 as shown below (but we are assuming that Figure 1 has full participation – double lines – between the EMPLOYEE superclass and the subclasses):

PHYSICIAN_ASSISTANT

ssn	specialization	fname	lname	address	phone

NURSE

ssn	dept.	fname	lname	address	phone

TECHNICIAN

ssn	work	fname	lname	address	phone

Rule 3:

Rule 3 works well if: (a) there is a disjoint relationship between the superclass and subclasses. It will create redundancy in the database if used with overlap scenarios. And, (b) if the subclasses are predicate defined (condition defined) or attribute defined. A predicate defined subclass is where a condition is placed on the value of some attribute of the superclass to determine the subclass. Figure 3 shows an example of a predicate defined subclass. If, as shown in figure 3, the EMPLOYEE entity has an additional attribute, JobType, and a condition is specified on the condition of membership in the PHYSICIAN ASSISTANT subclass by the condition (jobType=”Physician_Assistant”), this is a defining predicate of the subclass, PHYSICIAN ASSISTANT. A predicate-defined subclass is shown by writing the predicate condition next to the arc that connects the subclass to the circle. Also, the defining attribute name is placed on the arc from the superclass to the circle. This rule is not recommended when subclasses have too many attributes (since this will cause too many null values).

For rule 3: Create a single relation that includes the superclass and its attributes as well as the

EMPLOYEE

ssn	fname	lname	address	phone	jobtype	specialization	department	work

subclasses and it's attributes. For this rule, we will map Figure 3 as shown below:

Rule 4:

Rule 4 works for both overlaps and disjoints, but it works better for overlaps. With disjoints, this rule will create null values when the entity is not a member of a particular subclass. This rule is also not recommended when subclasses have too many attributes (since this will also cause too many null values). If subclasses have few attributes, however, this rule may be preferred to rule 1 or rule 2 since it will eliminate the need for a relational join. In this rule, a flag is created for each superclass tuple that belongs to a subclass.

For rule 4: Create a single relation that includes the attributes of the superclass and the attributes of its subclasses. To illustrate this rule we will map Figure 4 to the COMPUTER relation, as shown at the bottom of this page.

CATEGORIES

The concept of categories extends the concepts of generalization entity types and specialization entity types even further. Categories are created by grouping entity types (generalization entity types or specialization entity types) by the roles they may play within relationships. So, categories can represent superclasses (generalization categories) or subclasses (specialization categories). Important contributions his this area have been made by Elmasri, et. al. (1985), Gogolla, et. al. (1991), Elmasri and Navathe (2007).

In Figure 5, the PAYOR could be inheriting from the PRIMARY INSURANCE, PATIENT, or OTHER RESPONSIBLE PARTY. The PRIMARY INSURANCE, PATIENT, and OTHER RESPONSIBLE PARTY represent a superclass (generalization category) of entity types. Each of the entity types in this generalization or superclass is a different entity type with different keys, but they play a common role – the role of a PAYOR. Here we would refer to the subclass (in this case, PAYOR) as a ***category*** or ***union type*** of two or more superclasses. Hence, a category is a subclass of a union of two or more superclasses that are *different* entity types (Elmasri et al., 1985; Elmasri & Navathe, 2007) playing a common role. A category is diagrammatically shown by a "$\cup$" in the circle that attaches the category to the superclasses, as shown in Figure 5. We can read Figure 5 as:

A PAYOR may be a PRIMARY INSURANCE or PATIENT or OTHER RESPONSIBLE PARTY.

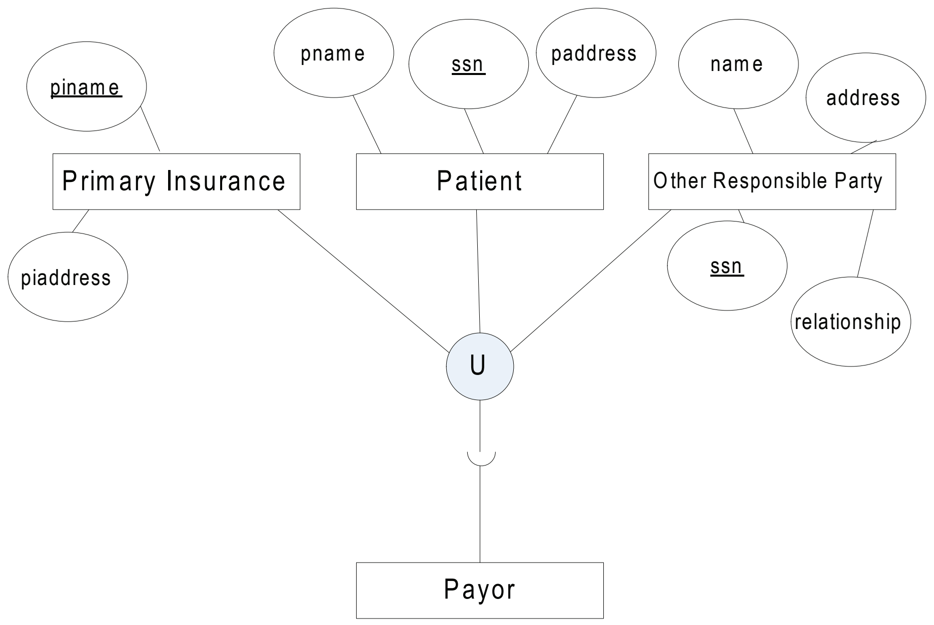
Participation Constraints in Categories

Just like other subclasses, categories can also have total (or full) participation or partial participation. Total participation means that the category holds the union of *all* entities in its superclasses. That is,

COMPUTER

item_no	make	item_model_no	lflag	fac_name	weight	fac_ssn	dflag	building	room_no

Figure 5. A Category



if there were full participation in figure 5 (double lines running from the category, PAYOR, to the circle with the "∪"), then every entity of PRIMARY INSURANCE would exist in PAYOR, and every entity of PATIENT would exist in PAYOR, and every entity of OTHER RESPONSIBLE PARTY would exist in PAYOR.

Partial participation means that a category holds only a subset of the union of all entities in its superclasses. That is, as shown in Figure 5, every entity of PRIMARY INSURANCE does not exist in PAYOR, and every entity of PATIENT does not exist in PAYOR, and every entity of OTHER RESPONSIBLE PARTY does not exist in PAYOR. Once again, partial participation would be shown by single lines from the category to the circle with the "∪".

Mapping Categories

There are two rules to map categories: (Elmasri & Navathe, 2007):

Rule 5:

Rule 5 should be used when the superclasses have different keys. For rule 5:

(i) Create a new relation to correspond to the category.
(ii) Since the keys of each of the superclasses are different, we cannot use any one of them as the key for this new relation. So, we have to specify a new key attribute (called a surrogate key) to correspond to the category in the new relation.
(iii) To be able to join the subclass with the superclass/superclasses, include the surrogate key attribute as the foreign key in each superclass relation.

To illustrate rule 5, we will map Figure 5 as shown below:

PRIMARY_INSURANCE

<u>piname</u>	piaddress	payorid

PATIENT

<u>ssn</u>	paddress	pname	payorid

OTHER_RESPONSIBLE_PARTY

<u>ssn</u>	relationship	name	address	payorid

PAYOR

<u>payorid</u>

Rule 6:

The second rule used to map categories is used when the superclass entities have the same key.

For example, we would map figure 6 as:

DORM

<u>dormnu</u>	dname

ON_CAMPUS

<u>dormnu</u>	bldg	supervisor

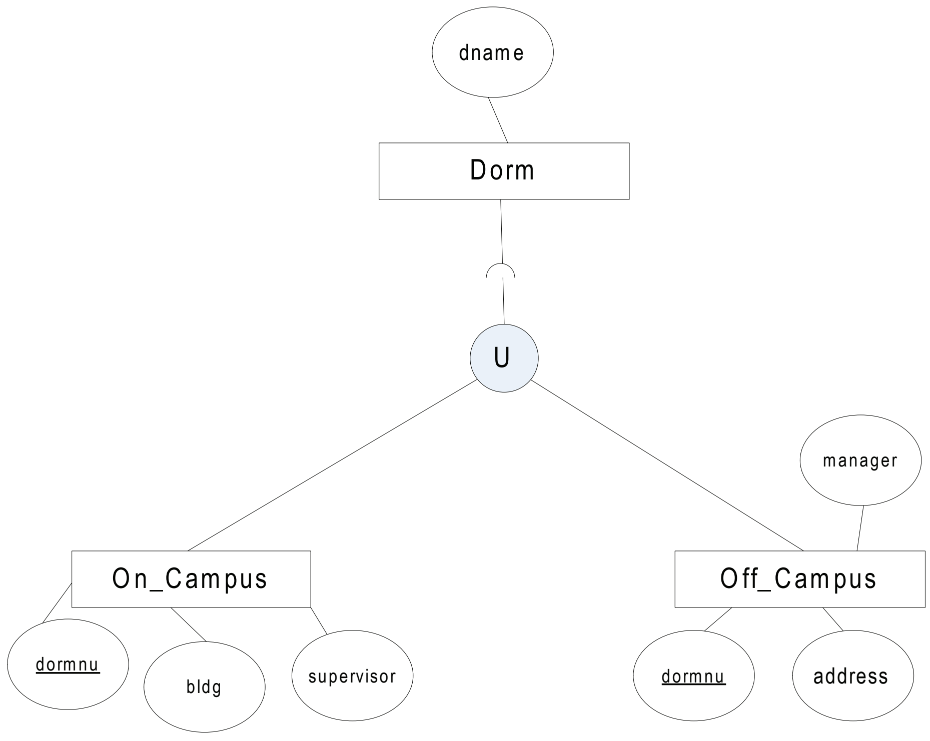
OFF_CAMPUS

<u>dormnu</u>	address	manager

Multiple Inheritance

In this paper we have given examples of how subclasses inherit from superclasses. In reality however, subclasses often inherit from more than one superclass. This concept is known as multiple inheritance. Categories are also not necessarily disjoint, so a given entity may be a member of several different categories. Due to the brief nature of this paper, we will not elaborate on the concept of multiple inheritance.

Figure 6. Category where superclasses have the same key



FUTURE TRENDS

Due to the intuitive nature of the ER and EER approach, and in the light of the next generation of Web applications (Shanmugasundaram, et al., 2000) which will depend on mature relational database technology with respect to storage, retrieval and update (Ceri, et al., 2000; Kappel, et al., 2001a; Kappel, et al., 2001b; Widom, 1999), the ER and EER approach will be heavily used in the future. Also, given the fact that most of the Web data have a hierarchical structure with classes and subclasses, the EER model can lend itself somewhat more naturally to a conceptual structure of web data. Hence we should see a rise the use of the EER model in the future.

CONCLUSION

Databases are becoming increasingly complex today. To deal with these complexities, it is becoming essential for database designers have to an understanding of the extended ER model, which incorporates generalizations/specializations and categories. In this paper we briefly presented generalizations/specializations and categories with the use of examples. We also presented rules to map generalizations/specializations and categories to a relational database.

REFERENCES

Bagui, S., & Earp, R. (2003). *Database Design Using Entity-Relationship Diagrams*. Auerbach Publications, Boca Raton, Florida: CRC Press.

Ceri, S., Fraternali, P., & Paraboschi, S. (2000). XML: Current Developments and Future Challenges for the Database Community. *Proceedings of the 7th International Conference on Extending Database Technology (EDBT),* Springer, LNCS 1777, Konstanz.

Chen, P. P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions of Database Systems, 1*(1), 9-36.

Chen, P., Thalheim, B., & Wong, L. Y. (1999). Future Directions of Conceptual Modeling. *LNCS*, 1565, 287.

Dey, D., Storey, V., & Barron, T., (1999). Improving Database Design through the Analysis of Relationships. *ACM Transactions on Database Systems, 24*(4), 453-486.

Dullea, J., Song, Li-Y., & Lamprou, I. (2003). An analysis of structural validity in entity-relationship modeling. *Data and Knowledge Engineering*, *47*(2), 167-205.

Elmasri, R., & Navathe, S. B. (2007). *Fundamentals of Database Systems*, 5th ed., Addison Wesley.

Elmasri, R., Weeldreyer J., & Hevner, A. (1985). The category concept: An extension to the Entity-Relationship model. *Data and Knowledge Engineering, 1*, 75-116.

Engels, G., Gogolla, M., Hohenstein, U., Hulsmann, K., Lohr-Richter, P., Saake, G., & Ehrich, H-D. (1992/93). Conceptual Modeling of database applications using an extended ER model. *Data and Knowledge Engineering 9*, 157-204.

Gogolla, M., & Hohenstein, U. (1991). Towards a Semantic View of an Extended Entity Relationship Model. *ACM Transactions on Database Systems*, *16*(3), 369-416.

Gogolla, M., Meyer, B., & Westerman, G. D. (1991). Drafting Extended EntityRelationship Schemas with QUEER. In T. J. Teorey, (Ed.), *Proc. 10th Int. Conf. on Entity-Relationship Approach*, (pp. 561-585).

Kappel, G., Kapsammer, E., & Retschitzegger, W. (2001a). Architectural Issues for Integrating XML and Relational Database Systems – The X-Ray Approach. *Proceedings of the Workshop* *on XML Technologies and Software Engineering,* Toronto.

Kappel, G., Kapsammer, E., & Retschitzegger, W. (2001b). XML and Relational Database Systems – A Comparison of Concepts'. *Proceedings of the 2nd International Conference on Internet Computing (IC).* CSREA Press, Las Vegas, USA.

Markowitz, V., & Shoshani, A. (1992). Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach. *ACM Transactions on Database Systems, 17*(3), 423-464.

Necasky, M. (2006). Conceptual Modeling for XML: A Survey. *Proceedings of the DATESO 2006 Annual International Workshop on Databases, Texts, Specifications, and Objects,* Desna-Cerna Ricka, Czech Republic.

Sanders, L. G. (1995). *Data Modeling*. International Thomson Publishing Company.

Shanmugasundaram, J., Shekita, E., Barr, R., Carey, M., Lindsay, B., Pirahesh, H., & Reinwald, B. (2001). Efficiently publishing relational data as XML document. *VLDB Journal 19*(2-3), 133-154.

Widom, J. (1999). Data Management for XML – Research Directions. *IEEE Data Engineering Bulletin. Special Issue on XML*, *22*(3), 44-52.

KEY TERMS

Category: A collection of objects or entities that is a subset of the union of different entity types; entities offering a similar role are grouped into a category.

Entity sets or entity types: Similar entities or objects with common properties are summarized into entity sets or entity types; graphically represented in rectangles.

Generalization: When entities have similar basic attributes, they can be classified into a generalized entity type. Generalization is the process of finding commonalities between entities to be able to abstract to a higher level entity set.

Inheritance: A subclass inherits all the attributes of a superclass and all the relationships that it (the superclass) participates in.

Specialization: A process of conceptual refinement to form specialized subclasses for entity sets. An entity type can have several specializations or categories of specializations defined on the basis of some distinguishing characteristics of the entities in the superclass.

Subclass: Same as specialization; a meaningful sub-grouping of entity-sets that needs to be represented explicitely. These sub-groups are a subset of the entities that belong to the entity set from which they are derived.

Superclass: Same a generalization. A superclass is the main set (entity) from which subsets (sub-classes) are defined based on meaningful criteria needed for a particular database.