**LIS 5782 – DATABASE MANAGEMENT SYSTEMS**

**FALL 2019**

**Grace Brothers 4: Mapping ERDs to Relations**

**I. Annotated Mapping – Refer to GB 3B (Grace Brothers ERD Version B)**

**(1) Introduction**

Mapping the ERD to a set of relations (also called deriving relations from an ERD) is the major step in the logical phase of DB design. The process involves:

* creating relations,
* normalizing the relations (often results in creating more relations),
* checking that user transactions are supported,
* checking that integrity constraints are not violated,
* checking that the organization’s business rules are supported, and
* reviewing the design with users.

These notes cover creating relations.

**(2) Representing Entities**

Tasks

* Create a relation for each entity in the ERD
* Include all simple attributes
* If composite attributes remain, break them into simple attributes
* Identify primary keys (PK) for strong entities & superclasses. Weak entities & subclasses cannot be assigned PKs until after relationships are represented.
* Create a new relation for multivalued attributes & set keys as for 1:\* relationships (see Representing Relationships below)
* Although not given in the text, I also use this phase for reviewing attribute names (1) to ensure consistency with naming conventions, (2) to make them as descriptive as possible, and (3) to shorten them as much as possible without sacrificing clarity and uniqueness.

Tentative Relations – Representing Entities

Address (adID, street, city, state, zip, adType)

Phone (phID, phNumber, phType)

In the EAR and ERD Version A (GB 3A) there were 8 entities: consultants, customers, employees, and suppliers each were associated with an address and a phone entity. The 8 entities were collapsed into 2 in ERD Version B (GB 3B). The attributes of the original 8 entities were named with prefixes indicating the entity with which they were associated (*e.g.,* **con**Street, **sup**Phone); now that they belong to the same entity, those prefixes are no longer necessary. The xxType attributes were added to distinguish between company and home addresses and among home, office, and cell phone numbers.

Consultant (compName, exprt)

Customer (cID)

Employee (empSSN)

Because these relations represent subclasses of the Person superclass, they do not yet get PKs. For the same reason, all of the shared attributes (*e.g.,* first names, dates of birth) go into the Person relation; only the attributes unique to each subclass stay in the relation representing it.

Person (pID, fName, lName, sex, dob)

This relation represents the Person superclass that contains the attributes common to all its subclasses. As with the Address and Phone attributes, these attributes no longer require their original prefixes (*e.g.,* **cst**FName, **emp**FName).

Dept (deptID, deptName)

Position (posID, posTitle)

Task (taskID, taskName)

Schedule (schdNo, date, sTimeIn, sTimeOut, aTimeIn, aTimeOut)

Stock (itemID, itemDesc, itemSize, itemColor, itemSex)

Supplier (supID, supName)

**(3) Representing Relationships**

Tasks

Some relationships are represented by collapsing the participating entities into a single relation with 2 copies of the PK. Other relationships are represented by copying 1 relation’s PK into another relation as a foreign key (FK). To determine the relation into which to put the FK, identify parent and child entities. A copy of the parent relation’s PK is placed into the child relation as the FK.

* 1:1 recursive relationships: use participation constraints
  + mandatory participation on both sides: combine into 1 relation with 2 copies of the PK (one column should be renamed to indicate the role)
  + mandatory participation on 1 side: combine into 1 relation with 2 copies of the PK (one column should be renamed to indicate the role)
  + optional participation on both sides: create a 2nd relation either the original or the copy can be parent or child
* 1:\* recursive relationships: add another column with the PK and rename it to indicate the role

GB relationship: Supervises

* place a copy of pID into EmpRec and rename it sprID
* \*:\* recursive relationships: see \*:\* and complex relationships below
* 1:1 binary relationships: use participation constraints
  + mandatory participation on both sides: combine into 1 relation and include any attributes on the relationship
  + mandatory participation on 1 side
    - entity having optional participation is parent
    - entity having mandatory participation is child
    - if there are attributes on the relationship, they are placed in the child relation
  + optional participation on both sides: either entity can be parent or child

GB example: Manages

* parent = Employee (inherits its PK from Person)
* child = Dept
* a copy of pID is placed in Dept and renamed mgrID
* 1:\* binary relationships: use cardinality constraints
  + entity on the 1 side is parent
  + entity on the \* side is child
  + if there are attributes on the relationship, they are placed in the child relation

GB example: Supplies

* parent: Supplier
* child: Stock
* a copy of supID and the attributes purDate, poNo, noPur, and costEach are placed in Stock

* \*:\* relationships & complex relationships (ternary & higher): create a new relation (=junction relation) to represent the relationship
  + PKs of the participating entities are placed into the junction relation as FKs
  + attributes on the relationship are placed into junction relation
  + create a surrogate PK for the junction relation

GB example of a \*:\* relationship: IsScheduled

* SchdTask is created as a junction relation containing copies of Schedule’s PK (schdNo) and Task’s PK (taskID)
* stID is created as a surrogate PK

GB example of a complex relationship: Sells

* SalesTrans is created as a junction relation
* both Employee and Customer inherit pID from Person – 2 copies of pID are placed in SalesTrans and renamed to indicate their roles in the sales transaction -- empID & cusID
* a copy of Stock’s PK (itemID) is placed in SalesTrans
* the attributes on the relationship, sellDate, status, noSold and priceEach are placed in SalesTrans
* sltrID is created as a surrogate PK
* Superclass/subclass relationships: use multiplicity constraints
  + superclass is parent entity & subclasses are child entities
  + relations required depends on multiplicity constraints
    - mandatory & nondisjoint: combine all entities into 1 relation
    - optional & nondisjoint: 2 relations – 1 for the superclass & 1 for all subclasses combined
    - mandatory & disjoint: 1 relation for each combination of the superclass with a subclass
    - optional & disjoint: 1 relation for the superclass and 1 for each of the subclasses
  + often useful to create a new relation for an identification code for each subclass in cases in which it is combined with others or with the superclass; this relation may have a \*:\* relationship with the superclass/subclass relation in which case a junction relation is required

GB example: Person/Employee, Customer, Consultant

* + the subclasses are combined with the superclass into a single relation, Person, using its PK, pID
  + PersonType is created to store terms (pTypeDesc) indicating whether a person is an employee, consultant, and/or customer with pTypeID created as a surrogate PK
  + a copy of pTypeID cannot simply be placed into Person as an FK because a person can be more than 1 type (the disjoint constraint on the superclass/subclass relationship is nondisjoint), so the relationship between Person and PersonType is \*:\* PersonRole, a junction relation, is created to contain copies of Person’s and PersonType’s PKs (pID, pTypeID) with prID created as a surrogate PK

**II. Schema (Un-normalized)**

This is an annotated set of the relations derived ERD Version 3B. With the exception of the annotations (in blue), this is an example of Project 2 format.

Address (adID, street, city, state, zip, adType)

Consult (conID, compName, exprt, deptID, pID) Represents Consults

Foreign key: deptID references Dept.deptID

Foreign key: pID references Person.pID

Dept (deptID, deptName, mgrID) Represents Manages

Foreign key: mgrID references Person.pID

EmpRec (erNo, startDate, endDate, salary, posType, deptID, empID, posID, sprID)

Foreign key: deptID references Dept.deptID Represents HiredFor

Foreign key: empID references Person.pID Represents HiredFor

Foreign key: posID references Position.posID Represents HiredFor

Foreign key: sprID references Person.pID Represents Supervises

Person (pID, fName, lName, sex, dob, ssn, cID)

PerAdd (paID, adID, pID) Represents Lists

Foreign key: adID references Address.adID

Foreign key: pID references Person.pID

PerPhone (ppID, pID , phID) Represents Provides

Foreign key: pID references Person.pID

Foreign key: phID references Phone.phID

PersonRole (prID, pID, pTypeID) Junction relation between Person and PersonType

Foreign key: pID references Person.pID

Foreign key: pTypeID references PersonType.pTypeID

PersonType (pTypeID, pTypeDesc) Partially represents superclass/subclass relationship

Phone (phID, phNumber, phType)

Position (posID, posTitle)

SalesTrans (sltrID, sellDate, status, noSold, priceEach, empID, cusID, itemID) Represents Sells

Foreign key: empID references Person.pID

Foreign key: cusID references Person.pD

Foreign key: itemID references Stock.itemID

Schedule (schdNo, Date, sTimeIn, sTimeOut, aTimeIn, aTimeOut, pID) Represents Assigned

Foreign key: pID references Person.pID

SchdTask (stID, schdNo, taskID) Represents IsScheduled

Foreign key: schdNo references Schedule.schdNo

Foreign key: taskID references Task.taskID

Stock (itemID, itemDesc, itemSize, itemColor, itemSex, purDate, poNo, noPur, costEach, supID) Represents Supplies

Foreign key: supID references Supplier.supID Represents Supplies

Supplier (supID, supName, adID, phID)

Foreign key: adID references Address.adID Represents DoesBusiness

Foreign key: phID references Phone.phID Represents IsContacted

Task (taskID, taskName)