Relational Design

ER to relational Mapping
Relational to ER Mapping
Good vrs Bad Relational Design
Normalization Process
ER to Relation Mapping

Strong Entity

Relation:
Employee(ssno, name, salary)

Key:
ssno
ER to Relation Mapping

Weak Entity

account

acct# customer balance

log

Relation:

transaction(acct#, trans#, amount)

Key:

acct# trans#

IND:

transaction[acct#] ⊆ account[acct#]

relational design -- 4
ER to Relational Mapping

**Relation:**
works_on(ssno, proj#, startdate)

**Key:**
ssno, proj#

**IND**
workson[proj#] ⊆ project[proj#]
workson[ssno] ⊆ employee[ssno]
ER to Relational Mapping

**Relation:**
works_on(ssno, proj#, startdate)

**Key:**
ssno

**IND**
workson[proj#] ⊆ project[proj#]
workson[ssno] ⊆ employee[ssno]

relational design -- 6
ER to Relational Mapping

Relation:
works_on(ssno, proj#, startdate)

Key:
ssno, proj#

IND
workson[proj#] ⊆ project[proj#]
workson[ssno] ⊆ employee[ssno]
employee[ssno] ⊆ workson[ssno]
ER to Relational Mapping

**Relation:**

worksonusing(ssno, proj#, toolid, startdate)

**Key:**

ssno, toolid

**IND**

worksonusing[proj#] ⊆ project[proj#]
worksonusing[ssno] ⊆ employee[ssno]
worksonusing[toolid] ⊆ tools[toolid]
employee[ssno] ⊆ worksonusing[ssno]
ER to Relational Mapping

**Relation:**
staff(ssno, name, salary, position)
faculty(ssno,name, salary, rank)
student_assistant(ssno, name, salary, percentge_time)

**Key:** ssno for all the relations
cannot use if partial: cannot represent employees who are neither staff, nor faculty, not student assistants!
cannot use if overlap: if staff could also be a student assistant, then redundancy requires a union to construct list of all employees

if no overlap, and total participation

ssno name salary
employee

staff
position

faculty
rank

student assistant
percentage time
ER to Relational Mapping

Relation:
employee(ssno, name, salary, jobtype, position, rank, percentage-time)

Key: ssno

- jobtype can be used to specify whether an employee is a staff, a faculty, or a student assistant
- lot of null values will be used.
- if an employee does not belong to any subclass, use null value for jobtype
- cannot be used if overlap
- total participation can be represented by preventing null in jobtype
- does not require union to construct the list of employees
ER to Relational Mapping

Relation:
employee(ssno, name, salary)
staff(ssno, position)
faculty(ssno, rank)
student_assistant(ssno, percentage_time)

Key: ssno for all the relations

IND
staff[ssno] ⊆ employee[ssno]
faculty[ssno] ⊆ employee[ssno]
student_assistant[ssno] ⊆ employee[ssno]

cannot represent total constraint.
ER to Relational Mapping

Relation:
employee(ssno, name, salary, Isstaff, position, Isfaculty, rank, Isstudentassistant, percentage-time)

Key: ssno

Isstaff, Isfaculty, Isstudent_assistant are boolean values which are either true or false. The relation will contain lot of null values.

cannot represent total constraint.

relational design -- 12
Relation to ER Mapping

- ER diagrams can be mapped to relational model (except sometimes total participation in a superclass/subclass relationship is difficult to model)
- Recall that at times during the design cycle we wish to do the reverse--that is, map relational schema to ER model.
- Can this always be done?
- So for the mapping to be correct, we should be able to represent the constraints over a relational schema in the ER model.
- Constraints in relational schema--functional dependencies, inclusion dependencies.
- Constraints in ER model: key constraints, cardinality constraints, participation constraints.
- Can we model fds and INDs using the constraints in ER model?
Example

- Consider we wish to build a catalog with three fields:
  street, city, zip
- So least we need to do is create an entity with the three attributes:
  street, city and zip, where the entites satisfy the following:
  - street city uniquely determines zip
  - zip uniquely determines city
- This can be modelled using the following two FDs in the relational model:
  - street city → zip
  - zip → city
- Can the same be modelled in ER using the set of constraints present?
Assume we create a single entity with the three attributes.
The only constraints that can be applied are the key constraints.

{street, city} and {street, zip} are keys

This, however, does not prevent presence of two catalog objects:

(kirby, champaign, 61801)
(florida, urbana, 61801)

which should be prevented since a zip uniquely determines a city!!
Example

Let's try creating an entity for each attribute and a relationship involving each entity.

We can now use cardinality constraints to get the required constraints?

Notice that street city uniquely determine zip, so relationship is functional wrt zip.

Similarly, street zip uniquely determine city, so relationship functional wrt city.

But how can we model a constraint zip determines the city which involves only two entities using a ternary relation?

This schema will also not prevent the catalog objects:

(kirby, champaign, 61801)

(florida, urbana, 61801)

which should be prevented since a zip uniquely determines a city!!
Example

Will this do?
No! since city-of may be an empty relationship and will thus not prevent
(kirby, champaign, 61801)
(florida, urbana, 61801)
which should be prevented since a zip uniquely determines a city!!
Actually, it can be formally shown that no ER schema can be used to represent
the constraints in this example.
(you should try other possibilities at home to convince yourself)
DatabaseDesign Process

- Conceptual Modelling -- ER diagrams
- ER schema transformed to relational schema (**ER to relation mapping**)  
  - Add additional constraints at this stage to reflect real world.  
- Resulting relational schema normalized to generate a good schema (**schema normalization process**)  
  - avoid redundancy  
  - avoid anamolies  
  - semantically equivalent to the original schema  
- Schema is tested over example databases to evaluate its quality  
- correctness results analysed and corrections to schema are made  
- corrections may be translated back to conceptual model to keep the conceptual description of data consistent (**relation to ER mapping**)  
- We have seen the ER to relation mapping. We next study normalization process.