Logical Database Design (ER to Relational Model mapping)

Dr Na Yao

Objectives

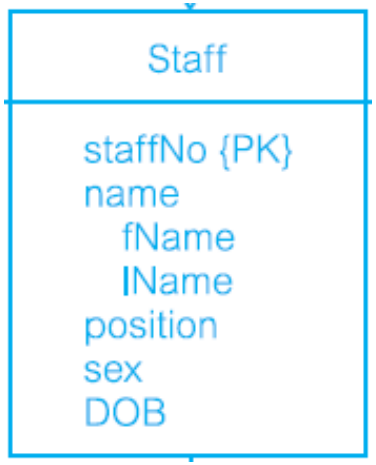
- Understand how to derive a set of relations from a conceptual data model (ER model)
- Able to map an ER model to a set of relations

Mapping ER model concepts to relations

- Entity
- Binary 1-1, 1-N, N-M relationships
- Complex relationships
- Multi-valued attributes

Entity

- For each entity:
  - create a relation that includes all the simple attributes of that entity.
  - For composite attributes, include only the constituent simple attributes (broken into several columns).



Staff (<u>staffNo</u>, fName, lName, position, sex, DoB)

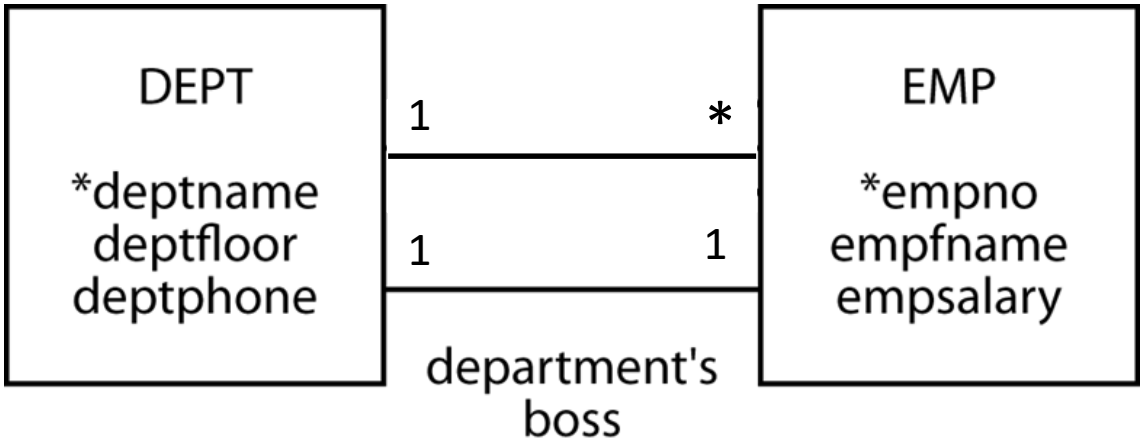
One-to-many (1:*) binary relationship types

- Post a copy of the primary key attribute(s) of one-side entity into the relation representing the many-side, to act as a foreign key.

  Staff (staffNo, fName, lName, position, sex, DoB)

  Client(clientNo, fName, lName, telNo, staffNo)

One-to-one (1:1) binary relationship types



- Mapping 1:1 relationship follows the same rules of placing foreign keys of one side to the other side.
- Three alternatives for this example.

One-to-one (1:1) binary relationship types

- Put the foreign key in dept.

  Doing so means that every instance of *dept* will record *empno* of the employee who is the boss. Since all departments have a boss, the foreign key will always be non-null.

- Put the foreign key in emp.

  This means that every instance of *emp* should record *deptname* of the department this employee manages. Since many employees are not bosses, the value of the foreign key column will generally be null.

- Put a foreign key in both dept and emp.

One-to-one (1:1) binary relationship types

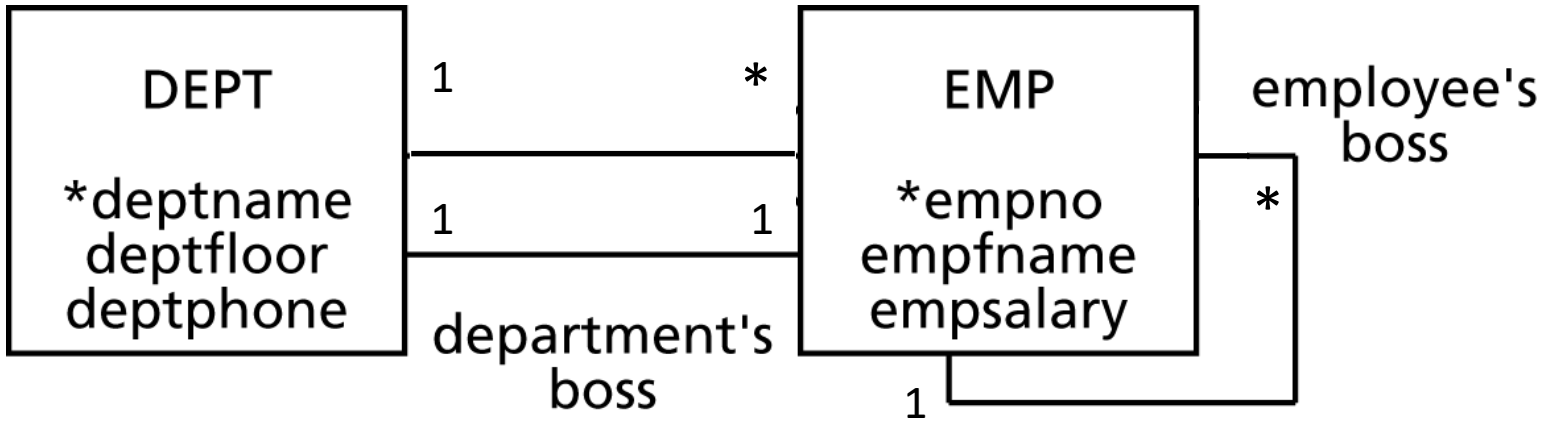
- A sound approach is to select the entity that results in the fewest nulls, since this tends to be less confusing to clients.
- In this example, the simplest approach is to put the foreign key in dept.

dept(<u>deptname</u>, deptfloor, deptphone, *empno*)

emp(<u>empno</u>, empfname, empsalary, *deptname*)

One-to-many (1:*) recursive relationships

- For a 1:* recursive relationship, follow the rules for participation as described above for a 1:m relationship. An additional column for the foreign key, is created for the entity at the “many ” side of the relationship.



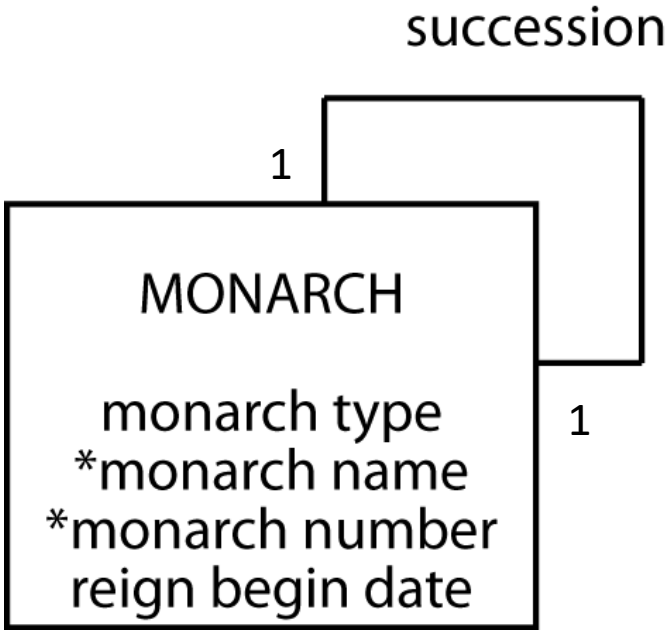
- What’s the new *emp* relation?

One-to-one (1:1) recursive relationships

- For a 1:1 recursive relationship, follow the rules for participation as described above for a 1:1 relationship.
  - *mandatory participation* on both sides, represent the recursive relationship as a single relation with two copies of the primary key.
  - *mandatory participation* on only one side, option to create a single relation with two copies of the primary key, or to create a new relation to represent the relationship. The new relation would only have two attributes, both copies of the primary key. As before, the copies of the primary keys act as foreign keys and have to be renamed to indicate the purpose of each in the relation.
  - *optional participation* on both sides, again create a new relation as described above.

One-to-one (1:1) recursive relationships

- The English monarch



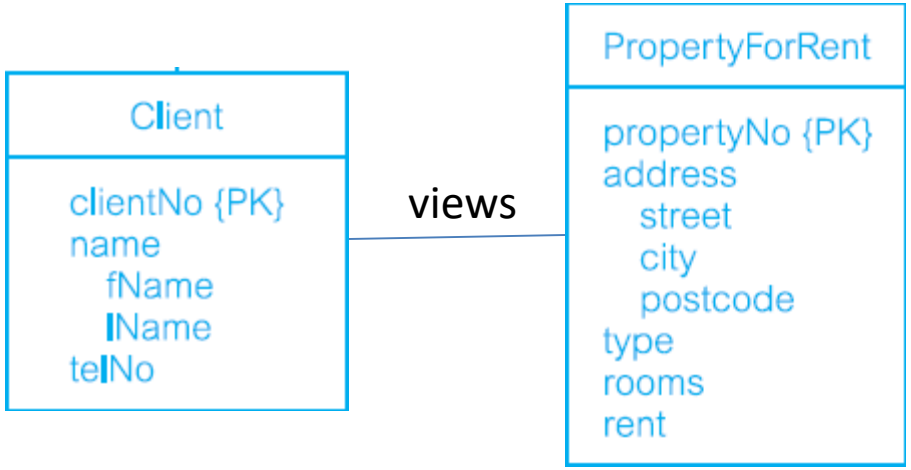
One-to-one (1:1) recursive relationships

- The monarch relation

monarch					
montype	monname	monnum	rgnbeg	*premonname*	*premonnum*
Queen	Victoria	I	1837/6/20	William	IV
King	Edward	VII	1901/1/22	Victoria	I
King	George	V	1910/5/6	Edward	VII
King	Edward	VIII	1936/1/20	George	V
King	George	VI	1936/12/11	Edward	VIII
Queen	Elizabeth	II	1952/2/6	George	VI

Many-to-many (*:*) binary relationship types

- Create a new relation to represent the relationship
- Foreign Key columns pointing to participating entities (These foreign keys will also form the primary key of the new relation)
- Further columns for attributes of the relationship.



- Viewing(clientNo, propertyNo, dateView, comment)

Complex relationship types

- Create a news relation to represent the relationship
- Include any attributes that are part of the relationship.
- Post a copy of the primary key attribute(s) of the participating entities as foreign keys.
- Any foreign keys that represent a ‘many’ relationship (for example, 1..*, 0..*) generally will also form the primary key of this new relation, possibly in combination with some of the attributes of the relationship.

Multi-Valued Attributes

- Create a new relation to represent multi-valued attribute and include primary key of entity in new relation, to act as a foreign key.
- Example:

  Telephone number in Branch relation:

  Telephone(telNo, BranchNo)

Superclass/subclass relationship types

- Identify superclass entity as parent entity and subclass entity as the child entity. There are various options on how to represent such a relationship as one or more relations.
- The selection of the most appropriate option is dependent on a number of factors such as the disjointness and participation constraints on the superclass/subclass relationship, whether the subclasses are involved in distinct relationships, and the number of participants in the superclass/subclass relationship.

Participation constraint	Disjoint constraint	Relations required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And}	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass