


COSC210 Database Management Systems
Lecture 6 - Relational Database Design by ER/EER-to-Relational Mapping
 Dr. Mitchell Welch

1 / 35

Summary

- ER-to-Relational Mapping Algorithms
- SQL Implementation
- EER-to-Relational Mapping Algorithms



Reading

- Chapter 9 from *Fundamentals of Database Systems* by Elmasri and Navathe



ER-to-Relational Mapping Algorithms

- Recall from lecture 6, we constructed an ER model for or COMPANY database:



Figure 9.1
 The ER conceptual schema diagram for the COMPANY database.



ER-to-Relational Mapping Algorithms

- The first step in the mapping process involves the mapping of the regular(Strong) entities to Relations.
- This process is fairly trivial - entity type E is used to create a relation R which includes all of the simple attributes of E
- A primary key must be selected from the key attributes of E - this can be a composite key from the attributes of E
- Key attributes that are not used as the primary key in the relation R should be specified as secondary keys as they may be useful for indexing in the final database implementation.

5 / 35

ER-to-Relational Mapping Algorithms

- The second step in the process involves mapping the weak entity types.
- Weak entity type W is mapped to a relation R
- Attributes of W are mapped to R
- In addition to these, a foreign key for R is created from the primary key of relations that correspond to the owner entity types of W .
- This will map the identifying relationship type of W .
- The primary key of R will be the combination of the owner(s) and the partial key of the weak entity type W .

6 / 35

ER-to-Relational Mapping Algorithms

- Third step is to map the 1:1 relationships.
- For relationship type R , identify the participating relations S and T that correspond to the participating entities.
- The mapping can be achieved in 3 ways:
 - Using foreign keys
 - Merging Relations
 - Through an additional relation

7 / 35

ER-to-Relational Mapping Algorithms

- Using a foreign key involves selecting one of the participating relations (say S) and including a foreign key that points to the primary key of the second participating relations (in this case T)
- It is best to select the entity that has total participation for the role of S
 - i.e. every tuple in S references a tuple in T (but not necessarily every tuple in T is referenced by a tuple in S)
- It is possible to map this the other way (i.e. T references S , however you will end up with NULL values in the foreign key attribute in T if there is not total participation.

8 / 35

ER-to-Relational Mapping Algorithms

- When participation on both sides of the 1:1 relationship is total, it is possible to merge the two relations into a single relation.
- The final approach involves using a third relation that contains tuples that map the keys from S to T .
- This is the approach that is required for M:N relationships, but it can also map 1:1 relationships.

ER-to-Relational Mapping Algorithms

- The fourth step is to map the 1:N relationship types.
- In a binary 1:N relationship R , relation S is on the N -side of the relationship.
- i.e. Multiple tuples of S point to a single tuple of T
- In this situation, a foreign key is placed on the S relation referencing the primary key of the T relation.
- This doesn't work the other way!
- An alternative approach is to use a third relation to map the keys from S to T

ER-to-Relational Mapping Algorithms

- The next step in the process is to map the M:N relationship.
- This is achieved through the use of an additional relation that contains two foreign keys that reference the participating relations (which represent the participating entities)
- In this approach, any UPDATE or DELETE operations should CASCADE through the foreign keys as the relation that represents the relationship-type has an existence dependency on the participating entities.

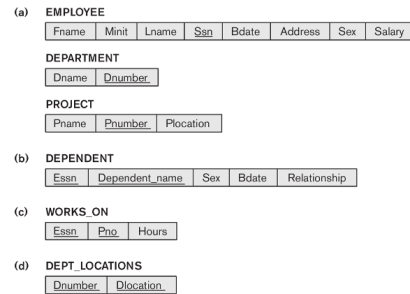
ER-to-Relational Mapping Algorithms

- Step 6 in this procedure involves representing the multi-valued attributes.
- The approach of this is relatively trivial and involves creating an additional relation R for each multivalued attribute A
- The relation will contain a primary key attribute K and an attribute that corresponds to A
- The primary key attribute K is also specified as a foreign key, referencing the relation that contains the multi-valued attribute.
- For example: DEPT_LOCATIONS

ER-to-Relational Mapping Algorithms

Figure 9.3

Illustration of some mapping steps.
a. *Entity* relations after step 1.
b. Additional *weak entity* relation after step 2.
c. *Relationship* relation after step 5.
d. Relation representing multivalued attribute after step 6.



ER-to-Relational Mapping Algorithms

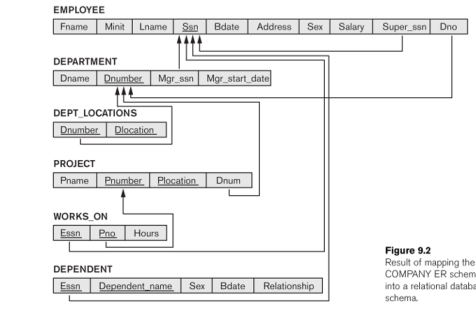


Figure 9.2
Result of mapping the COMPANY ER schema into a relational database schema.

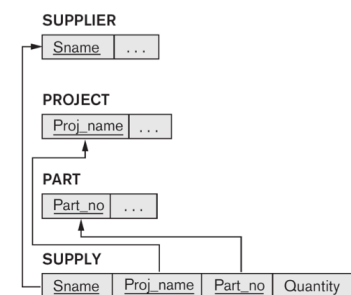
ER-to-Relational Mapping Algorithms

- The final task is to map the N-ary relation types.
- This is achieved through the creation of a new relation, say *S*.
- *S* contains a set of foreign keys that reference the primary keys of the participating relations.
- The primary key of *S* will be the combination of the foreign keys.

ER-to-Relational Mapping Algorithms

Figure 9.4

Mapping the *n*-ary relationship type SUPPLY from Figure 7.17(a).



SQL Implementation

- Once the entity-relationship model has been mapped to a set of relations, we can implement the relations using SQL.
- This is where we need to think carefully about the operations for the referentially triggered actions.
- This can be demonstrated through the implementation of the COMPANY database

17 / 35

SQL Implementation

- The EMPLOYEE entity is represented as an SQL table.
- Notice that the multi-attribute name has been decomposed into three separate columns

```
CREATE TABLE employee (
  fname    varchar(15) NOT NULL,
  minit    varchar(1),
  lname    varchar(15) NOT NULL,
  ssn      char(9) PRIMARY KEY,
  bdate    date,
  address  varchar(50),
  sex      char,
  salary   decimal(10,2),
  superssn char(9),
  dno      integer,
  foreign key (superssn) references employee(ssn)
    ON DELETE SET NULL ON UPDATE CASCADE,
  foreign key (dno) references department(dnumber)
    ON DELETE SET NULL ON UPDATE CASCADE
);
```

18 / 35

SQL Implementation

- The DEPARTMENT and PROJECT entities can be implemented in a similar fashion.

```
CREATE TABLE department (
  dname    varchar(25) not null,
  dnumber  integer primary key,
  mgrssn   char(9) not null,
  mgrstartdate date
);

CREATE TABLE project (
  pname    varchar(25) unique not null,
  pnumber  integer primary key,
  plocation varchar(15),
  dnum     integer not null,
  foreign key (dnum) references department(dnumber)
);
```

19 / 35

SQL Implementation

- The weak entity DEPENDENT is implemented as a table with a foreign key referencing the employee table.
- Notice that the primary key consists of the essn and the dependent_name - this represents the esi

```
CREATE TABLE dependent (
  essn     char(9),
  dependent_name varchar(15),
  sex      char,
  bdate    date,
  relationship varchar(8),
  primary key (essn, dependent_name),
  foreign key (essn) references employee(ssn)
);
```

20 / 35

SQL Implementation

- The WORKS_ON table implements the M:N relationship between EMPLOYEE and PROJECT.
- The DEPT_LOCATION table implements the multi-valued department location attribute. Note the direction of the relationship.

```
CREATE TABLE dept_locations (
  dnumber integer,
  dlocation varchar(15),
  primary key (dnumber,dlocation),
  foreign key (dnumber) references department(dnumber)
);

CREATE TABLE works_on (
  essn char(9),
  pno integer,
  hours decimal(4,1),
  primary key (essn,pno),
  foreign key (essn) references employee(essn),
  foreign key (pno) references project(pnumber)
);
```

21 / 35

EER-to-Relational Mapping Algorithms

- Recall that the EER model is an extension of the ER model, hence all of the algorithms we have looked at so far will be used in an EER model.
- Two extensions are added:
 - An algorithm for mapping Specialisations/Generalisation
 - An algorithm for mapping union types
- These are applied as steps 8 and 9.

22 / 35

EER-to-Relational Mapping Algorithms

- First we will look at mapping the Specialisation/Generalisation (step 8 in the process)
- There are 4 options for representing this relationship:
 - Multiple relations - superclass and subclasses
 - Multiple relations - subclasses only
 - Single relation using a *type* attribute
 - Single relation with multiple *type* attributes

23 / 35

EER-to-Relational Mapping Algorithms

- In the first approach we use a relation for each entity in the superclass/subclass relationship.
- The superclass relation, say L will be given a primary key $PK(L)$ based upon the attributes of the entity it represents.
- We then generate a relation for each subclass (we will denote these as L_i)
- The subclass relations will have a primary key that serves as a foreign key such that $PK(L_i) = PK(L)$
- This link represents the relationship between the entities.
- This approach is suitable for any specialisation (d,o, total and/or partial)

24 / 35

EER-to-Relational Mapping Algorithms

- In the second approach we create a relation for each subclass.
- The attributes from the superclass are simply replicated within the subclass relations.
- This approach only works if the relationship is disjoint and it is not recommended if the specialisation is overlapping - this leads to redundancy across relations for the same entity.
- This approach involves redundancy.

25 / 35

EER-to-Relational Mapping Algorithms

- The third approach uses a single relation with a *type* attribute to indicate the subclass of the entity that the tuple represents.
- This will only work for disjoint specialisations.
- This approach will also generate NULL values within tuples with types for which the attributes are not applicable.

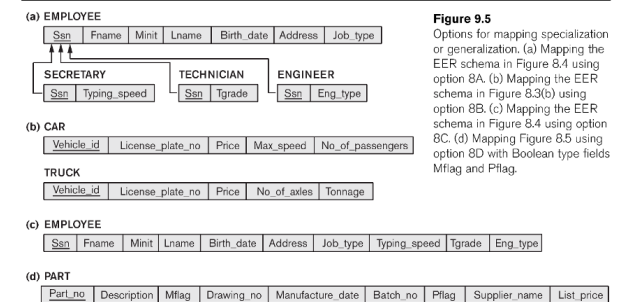
26 / 35

EER-to-Relational Mapping Algorithms

- The final approach involves using a single relation and a boolean-type attribute (i.e. a *flag*) to indicate membership in a class.
- This will work for overlapping specialisations.

27 / 35

EER-to-Relational Mapping Algorithms



28 / 35

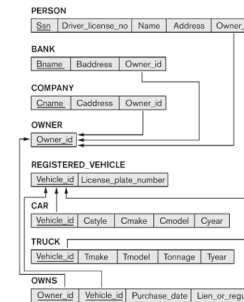
EER-to-Relational Mapping Algorithms

- The final feature we need to address are the union types.
- Recall that in this situation we have a subclass inheriting attributes and relationships from *multiple* superclasses.
- In this situation, the superclasses will be represented by separate relations that will usually have different primary key attributes.
- To work around this, we can generate a *surrogate key* to link the superclass relations.
- We then generate a relation for the subclass using this surrogate key as its primary key.
- This approach allows for all inheritance structures within the union relationship.

29 / 35

EER-to-Relational Mapping Algorithms

Figure 9.7
Mapping the EER categories
(union types) in Figure 8.8 to
relations.



30 / 35

Summary

- ER-to-Relational Mapping Algorithms
- SQL Implementation
- EER-to-Relational Mapping Algorithms

31 / 35

Questions?

32 / 35

Next Lecture

- Relational Algebra

Reading

- Chapters 4 and 9 from *Fundamentals of Database Systems*
- Chapter 7 from *Fundamentals of Database Systems* this week's prac session.