

Translating from ERD

24 November 2009
Lecture 6

Topics for Today

- Translating from ERD to Relations
- Source: Ramakrishnan and Gehrke 3.5
- **Reminder:** Quiz next week (December 1) on Relational Model

November 24, 2009

ISE 322: Database Systems

2

From ER to Relations

- Entity Relationship Diagrams are fairly easy to map to relational databases
 - Some (intentional) similarities jump out
 - Some things are expensive to do in SQL
- We'll talk about:
 - Entity sets
 - Relationship sets
 - Relationship sets with key constraints
 - Relationship sets with participation constraints
 - Weak entity sets
 - Class hierarchies
 - Aggregation

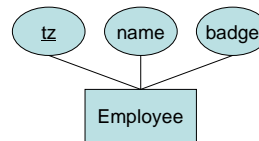
November 24, 2009

ISE 322: Database Systems

3

Entity Sets to Relations

- Straightforward:
 - Attributes → Columns
 - Primary Key is already marked
 - Candidate keys need to be considered



| tz | Name | badge |
|-----|-------|-------|
| 123 | Chaim | C498A |
| 456 | Moshe | 12EE8 |
| 789 | Harel | 988B |

```
CREATE TABLE Employees
(tz INT(9),
name CHAR(20),
badge CHAR(10),
PRIMARY KEY (tz),
UNIQUE (badge))
```

November 24, 2009

ISE 322: Database Systems

4

Relationship Sets (No Constraints)

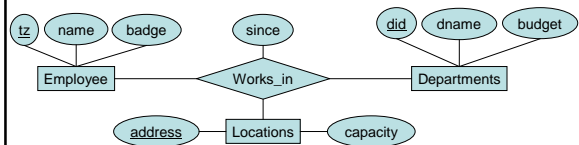
- Each relationship becomes a relation
- One column for each attribute of the relationship
- One column for the primary key of each connected entity set
 - The columns are *foreign keys*
 - Double connections from an entity set → two foreign key
- The primary key of the relation is the set of the foreign keys

November 24, 2009

ISE 322: Database Systems

5

Relationship Sets (No Constraints)



```
CREATE TABLE Departments
(did CHAR(20),
dname CHAR(20),
budget REAL,
PRIMARY KEY (did))
```

```
CREATE TABLE Locations
(address CHAR(40) PRIMARY
KEY,
capacity INTEGER)
```

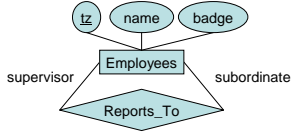
```
CREATE TABLE Works_in
(tz INT(9),
did CHAR(20),
address CHAR(40),
since DATE,
PRIMARY KEY (tz, did, address),
FOREIGN KEY (tz) REFERENCES Employee,
FOREIGN KEY (did) REFERENCES Departments,
FOREIGN KEY (address) REFERENCES Locations)
```

November 24, 2009

ISE 322: Database Systems

6

Reflexive Relations



```
CREATE TABLE Reports_To
(supervisor_tz INT(9),
subordinate_tz INT(9),
PRIMARY KEY (supervisor_tz , subordinate_tz ),
FOREIGN KEY (supervisor_tz ) REFERENCES Employees(tz),
FOREIGN KEY (subordinate_tz ) REFERENCES Employees(tz))
```

November 24, 2009

ISE 322: Database Systems

7

Adding Key Constraints

- When there are key constraints which limit the appearance of one entity in a relationship set, there are two options:
 - Create relation for the relationship set which has a candidate key which is a foreign key (or a combination of foreign keys) from the constrained entity set
 - The candidate key could be chosen as the primary key too
 - Augment the constrained entity set with the information in the relationship set
- Some examples might help...

November 24, 2009

ISE 322: Database Systems

8

Key Constraint Example



- Option A:


```
CREATE TABLE Manages
(tz INT(9),
did CHAR(20),
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (tz) REFERENCES Employees,
FOREIGN KEY (did) REFERENCES Departments)
```

PRIMARY KEY (tz, did),
UNIQUE (did), Super key!

November 24, 2009

ISE 322: Database Systems

9

Key Constraint Example



- Option B:


```
CREATE TABLE Departments2
(did INT(20),
dname CHAR(20),
budget REAL,
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (tz) REFERENCES Employees,
FOREIGN KEY (did) REFERENCES Departments)
```
- Option A:


```
CREATE TABLE Manages
(tz INT(9),
did CHAR(20),
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (tz) REFERENCES Employees,
FOREIGN KEY (did) REFERENCES Departments)
```

November 24, 2009

ISE 322: Database Systems

10

Which is better? It Depends...

- Reasons to use A:
 - If the relationship is sparse:
 - Ex. Most Departments do not have managers
 - Saves space
 - Lets Departments be smaller and run faster
 - If both sides are constrained
 - Ex. One manager per department, only manages one dept
 - If there are a lot of attributes in the relationship
- Reasons to use B:
 - If the relationship is dense on the constrained side
 - Ex. Most Departments have managers
 - If we will be doing a lot of joins on the constrained side plus the relationship
 - Saves on computation time since joins are not cheap

November 24, 2009

ISE 322: Database Systems

11

Relationships with Participation

- Participation constraints are a bit more complicated
 - This is an artifact of SQL, not the relational model
- The only general way to solve it is using an adaptation of Option B above:
 - Augment the constrained relation with the fields of the relationship
 - Force the foreign keys to the other entities to be NOT NULL
- Option A will not work
 - Why not?

November 24, 2009

ISE 322: Database Systems

12

Participation Example

```

CREATE TABLE Departments
  (did CHAR(20),
   dname CHAR(20),
   budget REAL,
   tz INT(9) NOT NULL,
   since DATE,
   PRIMARY KEY (did),
   FOREIGN KEY (tz) REFERENCES Employees ON DELETE NO ACTION)

```

Why does this work?

What does this do?

November 24, 2009 ISE 322: Database Systems 13

Participation Example

```

CREATE TABLE Manages2
  (did CHAR(20) NOT NULL,
   tz INT(9) NOT NULL,
   since DATE,
   PRIMARY KEY (did),
   FOREIGN KEY (tz) REFERENCES Employees ON DELETE NO ACTION
   FOREIGN KEY (did) REFERENCES Departments ON DELETE NO ACTION)

```

Why does this not work?
Any other ideas?

November 24, 2009 ISE 322: Database Systems 14

Participation Example

```

CREATE TABLE Works_In
  (tz INT(9) NOT NULL,
   did CHAR(20) NOT NULL,
   since DATE,
   PRIMARY KEY (tz, did),
   FOREIGN KEY (tz) REFERENCES Employees,
   FOREIGN KEY (did) REFERENCES Departments)

```

Why does this not work?

November 24, 2009 ISE 322: Database Systems 15

Participation Problems

- The only way to enforce multiple participation constraints is to use *assertions* or *table constraints*
 - We'll talk more about them later
 - Or to just let the application program worry about them
- One special case which does work:
 - When all entities have participation and key constraints (i.e. 1-to-1 and covered)
 - Just make all the entities and relationship in to one bit table
 - Every entity primary key is a candidate key

November 24, 2009 ISE 322: Database Systems 16

Weak Entity Sets

- Weak Entities have a partial key
 - Rely on the primary key of another entity
 - Must have total participation and key constraint with the identifying relationship
 - Must be deleted when the identifying entity is deleted
- Approach:
 - Relation for the weak entity
 - Primary key = partial key + foreign key to identifying entity
 - ON DELETE CASCADE

November 24, 2009 ISE 322: Database Systems 17

Weak Entity Example

```

CREATE TABLE Dep_Policy (
  pname CHAR(20),
  age INTEGER,
  cost REAL,
  tz INT(9),
  PRIMARY KEY (pname, tz),
  FOREIGN KEY (tz) REFERENCES Employees ON DELETE CASCADE)

```

Why does this work?

Can't be null

Why?

November 24, 2009 ISE 322: Database Systems 18

Inheritance

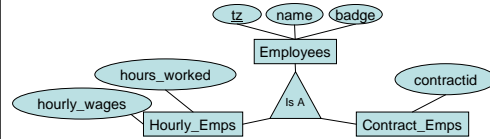
- We talked about IS A inheritance
- There are two general approaches:
 - Create a relation for each superclass and subclass
 - Subclasses have a foreign key to the superclass
 - Subclasses store just their attributes
 - Subclasses do ON DELETE CASCADE
 - Create a relation for each subclass only
 - Subclasses each have all of their attributes, including ones from the superclass
- Some examples...

November 24, 2009

ISE 322: Database Systems

19

Inheritance Example



Option A:

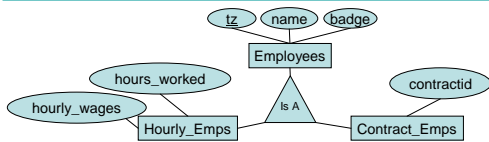
```
CREATE TABLE Employees (tz, name, badge...)
CREATE TABLE Hourly_Emps
(tz INT(9) PRIMARY KEY,
hours_worked INTEGER,
hourly_wages REAL,
FOREIGN KEY (tz) REFERENCES Employees ON DELETE CASCADE)
```

November 24, 2009

ISE 322: Database Systems

20

Inheritance Example



Option B:

```
CREATE TABLE Employees (tz, name, badge...)
CREATE TABLE Hourly_Emps
(tz INT(9) PRIMARY KEY,
name CHAR(20),
badge CHAR(10),
hours_worked INTEGER,
hourly_wages REAL)
```

November 24, 2009

ISE 322: Database Systems

21

Which is better? It depends...

- **Option A: Pluses:**
 - Always works
 - Lets you have some entities which do not belong to any subclass (if no coverage)
 - Lets you easily work just on supertype attributes
 - Lets you work on all supertype elements easily (no joins)
- **Option B: Pluses:**
 - Saves on tables
 - If supertypes are not usually examined separately, save time
 - May be more intuitive for constraints and assertions
- **Minuses:**
 - If there is no coverage, doesn't work
 - Key constraints between subtypes is hard or impossible
 - How would you force two subtypes to have a uniquely identifying key?
 - Cross types create problems
- **Minuses:**
 - Uses more tables
 - If you are working with super and sub attributes, you must do a join each time

November 24, 2009

ISE 322: Database Systems

22

Inheritance Constraints

- What about enforcing coverage and overlap?
 - In general only using assertions and table constraints

November 24, 2009

ISE 322: Database Systems

23

Aggregation

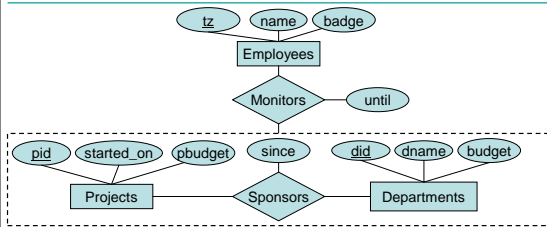
- We combine a relationship and make it into an entity
- General approach – make a relation for the relationship involving aggregation:
 - Add a foreign key for each regular entity connected
 - Add a foreign key for each primary key of each entity in the aggregated relationship
 - Add a column for each attribute of the relationship
- An example...

November 24, 2009

ISE 322: Database Systems

24

Aggregation Example



```
CREATE TABLE Monitors (tz INT(9), until DATE,
pid INTEGER, did CHAR(20),
FOREIGN KEY (tz) REFERENCES Employees,
FOREIGN KEY (pid, did) REFERENCES Sponsors (pid, did))
```

November 24, 2009

ISE 322: Database Systems

25

Aggregation Issues

- We treat the aggregated entity like any other
- In the special case where Sponsors has no attributes and has participation constraints on both sides
 - We may drop the Sponsors relation
 - Put everything in the Monitors relation

November 24, 2009

ISE 322: Database Systems

26

Conclusion

- Translating from ERD to Relations

November 24, 2009

ISE 322: Database Systems

27