Database Systems

*ER to Relational Model Mapping*

Feifei Li

University of Utah
Logical DB Design: ER to Relational

- Entity sets to tables.

```
CREATE TABLE Employees
(
    ssn CHAR(11),
    name CHAR(20),
    lot INTEGER,
    PRIMARY KEY (ssn)
)
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
</tr>
</tbody>
</table>

CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
In translating a many-to-many relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys).
  - This set of attributes forms a superkey for the relation.
- All descriptive attributes.

```
CREATE TABLE Works_In(
  ssn  CHAR(1),
  did  INTEGER,
  since DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments
)
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>did</th>
<th>since</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>51</td>
<td>1/1/91</td>
</tr>
<tr>
<td>123-22-3666</td>
<td>56</td>
<td>3/3/93</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>51</td>
<td>2/2/92</td>
</tr>
</tbody>
</table>
Review: Key Constraints

- Each dept has at most one manager, according to the key constraint on Manages.

Translation to relational model?
Translating ER Diagrams with Key Constraints

- Map relationship set to a table:
  - Note that `did` is the key now!
  - Separate tables for Employees and Departments.

- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```
Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a **participation constraint**: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)
We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11) NOT NULL,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE NO ACTION)
```
As in C++, or other PLs, attributes are inherited.

If we declare A ISA B, every A entity is also considered to be a B entity.

- **Overlap constraints:** Can Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
- **Covering constraints:** Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
Translating ISA Hierarchies to Relations

General approach:
- 3 relations: Employees, Hourly_Emps and Contract_Emps.
  - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages, hours_worked, ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
  - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.

Alternative: Just Hourly_Emps and Contract_Emps.
- *Hourly_Emps*: *ssn, name, lot, hourly_wages, hours_worked*.
- Each employee must be in one of these two subclasses.
E/R to Relations

E/R diagram

Relational schema, e.g.
account=(bname, acct_no, bal)

E = (a₁, ..., aₙ)

E₁

R₁

E₂

R₁ = (a₁, b₁, c₁, ..., cₖ)
More on relationships

- What about:

![Diagram of relationships]

- Could have:

\[ R1 = (a_1, b_1, c_1, \ldots, c_k) \]

- put \( b_1 \) as the key for \( R1 \), it is also the key for \( E2 = (b_1, \ldots, b_n) \)

- Usual strategy:
  - ignore \( R1 \)
  - Add \( a_1, c_1, \ldots, c_k \) to \( E2 \) instead, i.e.
  - \( E2 = (b_1, \ldots, b_n, a_1, c_1, \ldots, c_k) \)
More

<table>
<thead>
<tr>
<th>Diagram</th>
<th>E1 = ((a_1, \ldots, a_n)) E2 = ((b_1, \ldots, b_m))</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Diagram 1" /></td>
<td>R1 = ((a_1, b_1, c_1, \ldots, c_k))</td>
</tr>
<tr>
<td><img src="" alt="Diagram 2" /></td>
<td>E1 = ((a_1, \ldots, a_n)) E2 = ((b_1, \ldots, b_m, a_1, c_1, \ldots, c_k))</td>
</tr>
<tr>
<td><img src="" alt="Diagram 3" /></td>
<td>E1 = ((a_1, \ldots, a_n, b_1, c_1, \ldots, c_k)) E2 = ((b_1, \ldots, b_m))</td>
</tr>
<tr>
<td><img src="" alt="Diagram 4" /></td>
<td>Treat as n:1 or 1:m</td>
</tr>
<tr>
<td><img src="" alt="Diagram 5" /></td>
<td>Foreign key also needs to be marked as a candidate key!</td>
</tr>
</tbody>
</table>

Treat as n:1 or 1:m
Foreign key also needs to be marked as a candidate key!
E/R to Relational

Method 1:
\[ E = (a_1, \ldots, a_n) \]
\[ S_1 = (a_1, b_1, \ldots, b_m) \]
\[ S_2 = (a_1, c_1, \ldots, c_k) \]

Method 2:
\[ S_1 = (a_1, \ldots, a_n, b_1, \ldots, b_m) \]
\[ S_2 = (a_1, \ldots, a_n, c_1, \ldots, c_k) \]

Q: When is method 2 not possible?
Tenary relationshipset:

- What about tenary:

![Diagram of tenary relationshipset]

- Strategy:
  - $E1(a_1, ..., a_n) E2(b_1, ..., b_m) E3(d_1, ..., d_l)$
  - $R1(a_1, b_1, d_1, c_1, ..., c_k)$
E/R to Relational

- Aggregation

E1, R1, E2, E3 as before
R2 = (c_1, a_1, b_1, d_1, ..., d_j)
ER Model to Relation Model Mapping Summary

- Usually easier to understand than Relational
- Expresses relationships clearly
- Rules to convert ER-diagrams to Relational Schema
- Some systems use ER-model for schema design
- Many use ER-model as a step before creating relational tables