6.6 Weak entity sets

They are sets w/o sufficient attributes to form a PK

Scenario:
- Customer makes one payment for several of their existing loans.
- Teller splits total payment into the individual amounts …
- …but the same payment number is assigned to each amount.

The weak entity set must be in a total and many-to-one relationship with its identifying entity set.
Alternative for modeling weak entity set: make it a multi-valued attributes of the identifying entity set.

- Draw the E-R diagram:

![E-R Diagram](image)

- Do you see problems with this approach?

Another alternative: make it a strong entity-set by including the PK of its identifying entity-set.

- Draw the E-R diagram:

![E-R Diagram](image)

- Do you see problems with this approach?
How do we decide?

→ Does the weak entity set participate in other relationships?

Another example:

- In a university, a *course* is a strong entity and a *course_offering* can be modeled as a weak entity.
- The discriminator of *course_offering* would be *semester* (including year) and *section_number* (if there is more than one section).
- If we model *course_offering* as a strong entity we would model *course_number* as an attribute.
6.7 Extended E-R features

**Hierarchy:** specialization, generalization, attribute inheritance

**Constraints on generalization:**
- Condition-defined vs. user-defined
- Disjoint vs. overlapping
- Total (use double line) vs. partial
Food for thought: What requirements does each of the constraints place on:

- Deletion?
- Insertion?
- Updating?

(Similar to FK constraints)
Aggregation:

Q: How do we represent relationships between/among relationships in E-R?

Example: We have the ternary `works_on` relationship and want to represent which manager is responsible for each task (combination of `employee`, `job`, `branch`). Only possible solution with what we have so far is a quartary relationship `manages`:

![Diagram of E-R model showing `works_on` relationship between `employee`, `job`, and `branch`, and `manages` relationship between `job` and `manager`.

Problem: redundancy!
Better solution: Treat relationships as higher-level entities:

Manages is now binary!
Read over lightly: 6.7.6 Alternative E-R notations

Read and understand: 6.8 DB design for the banking enterprise
6.9 Reduction of E-R models to Relational Schemas

Entity-sets

- A strong entity set reduces to a schema with the same attributes.

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

```plaintext
payment =
  ( loan_number, payment_number, payment_date, payment_amount )
```
Relationship-sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

- Example: schema for relationship set borrower
  
  \[ \text{borrower} = (\text{customer_id}, \text{loan_number}) \]

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the “one” side.

- Example: Instead of creating a schema for relationship set \text{account\_branch}, add an attribute \text{branch\_name} to the schema arising from entity set \text{account}. 

![Data model diagram showing relationships between account, account-branch, branch, account-number, balance, assets, branch-name, and branch-city.](image-url)
For one-to-one relationship sets, either side can be chosen to act as the “many” side.

- That is, extra attribute can be added to either of the tables corresponding to the two entity sets.

If participation is partial on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values.

The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

- Example: The payment schema already contains the attributes that would appear in the loan_payment schema (i.e., loan_number and payment_number).

Conclusion: We don’t have to represent it as a schema!
Composite attributes are flattened out by creating a separate attribute for each component attribute.

- Example: given entity set `customer` with composite attribute `name` with component attributes `first_name` and `last_name` the schema corresponding to the entity set has two attributes `name.first_name` and `name.last_name`.

A multivalued attribute $M$ of an entity $E$ is represented by a separate schema $EM$.

- Schema $EM$ has attributes corresponding to the primary key of $E$ and an attribute corresponding to multivalued attribute $M$.

- Example: Multivalued attribute `dependent_names` of `employee` is represented by a schema:
  
  `employee_dependent_names = (employee_id, dname)`

- Each value of the multivalued attribute maps to a separate tuple of the relation on schema $EM$.
  
  For example, an employee entity with primary key 123-45-6789 and dependents Jack and Jane maps to two tuples: (123-45-6789, Jack) and (123-45-6789, Jane).
Write the schema implementation of the following E-R model involving a multi-valued attribute:

![Diagram of a multi-valued attribute E-R model]

Write the schema implementation of the following E-R model involving a ternary relationship set:

![Diagram of a ternary relationship E-R model]
CREATE TABLE Sold  (
    Price  INTEGER,    -- rel. attribute
    Date   DATE,       -- rel. attribute
    ProjId  INTEGER,  -- entity PK
    SupplierId  INTEGER, -- entity PK
    PartNumber  INTEGER, -- entity PK

    PRIMARY KEY (ProjId, SupplierId, PartNumber, Date),

    FOREIGN KEY (ProjId) REFERENCES Project,
    FOREIGN KEY (SupplierId) REFERENCES Supplier (Id),
    FOREIGN KEY (PartNumber) REFERENCES Part (Number)  )
**Specialization/generalization** -- Method 1:

- Form a schema for the higher-level entity
- Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, credit_rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, salary</td>
</tr>
</tbody>
</table>

Note: There’s a similar example in the text, but using `person_id` as PK

**Specialization/generalization** -- Method 2:

- Form a schema for each entity set with all local and inherited attributes

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>person</strong></td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, street, city, credit_rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, street, city, salary</td>
</tr>
</tbody>
</table>

Only if specialization is complete!

What if overlapping? What if FK points to person?
**Aggregation**: create a schema containing
- primary key of the aggregated relationship,
- the primary key of the associated entity set
- any descriptive attributes

`manages (employee_id, branch_name, title, manager_name, project)`
Read 6.9.7 – Schemas for banking enterprise
6.10 Other aspects of DB design

Classical trade-off in DB systems: throughput vs. response time
- Example: to sort or not to sort?

Read the entire Section 6.10
SKIP 6.11 – UML

Homework:
- End of chapter 1, 5, 8, 9, 12, 26, 27
- Due Wed, March 30

No class and no office hours this Friday!