(Extended) Entity Relationship Modelling and Mappings to the Relational Data Model

Conceptual Data Model Concepts

- “There exist things which have certain properties and which may be related in some way(s) to other things. Data represents specific facts about the things”
- Entity
  - thing or object that exists in its own right and is distinguishable, represented by an Entity Type of which there will be many Entity Instances…
  - physical objects, events, activities, associations
- Relationship
  - an association between several entities represented by a Relationship Type of which there will be many Relationship Instances

Entity Types and Relationship Types

Optional Relationship Types

Conceptual Data Model Concepts

- Attribute
  - fact about an Entity Type or Relationship Type
  - an entity is often expressed as a set of attributes
- Entity Set or Extent
  - Set of all Entity Instances of the same Entity Type
- Relationship Set or Extent
  - Set of all Relationship Instances of the same Relationship Type

Mini World

Simplified phases of Database Design

Data requirements

Conceptual design

Data model design

Physical design

Internal schema (for the same DBMS)

Data Definition Language Statements
**Many:many Relationship Types**

Staff \(\rightarrow\) Teaches \(\rightarrow\) Course

- ST1 \(\rightarrow\) C1
- ST2 \(\rightarrow\) C2
- ST3 \(\rightarrow\) C3
- ST4 \(\rightarrow\) C4
- ... \(\rightarrow\) ...

**Recursive Relationship Types**

Staff \(\rightarrow\) Manages

- ST1 \(\rightarrow\) r1
- ST2 \(\rightarrow\) r2
- ST3 \(\rightarrow\) r3
- ST4 \(\rightarrow\) r4
- ST5 \(\rightarrow\) r5
- ST6 \(\rightarrow\) ...

- 1. Manager
- 2. Employee

**Entity Relationship Model**

- STUDENT
- SCHOOL
- YEAR
- ENROL
- TUTOR
- COURSE

**Attributes in Conceptual Modelling**

- For each and every attribute must define domain, data type, format and whether it can be null
- Every entity type must have a key attribute or set of attributes
- Composite or Atomic
- Single-valued or Multi-valued
- Derived
- Null valued

**Properties of Relationship Types**

- **Degree**
  - The number of participating entity types
- **Cardinality ratios**
  - The number of instances of each of the participating entity types which can partake in a single instance of the relationship type 1:1, 1:many, many:1, many:many
- **Participation (optionality)**
  - The relationship instance doesn’t have to exist
  - Whether an entity instance has to participate in a relationship instance
- **Role**
  - The function that a particular entity type plays in a relationship type

**Semantic Data Models**

- Extended-Entity-Relationship Modelling
- Entity Attribute Relationship Modelling
- Entity Relationship Attribute Modelling
- Entity Modelling
- Object Modelling
- IFO, NIAM etc...

- Extensions for temporal, constraints, rules etc

- Chen 1976
- Entity Relationship Modelling
Composite Keys

Roles & Recursive Relationships
- The function of an entity type in a relationship type

Roles & Association Relationships
- The function of an entity type in a relationship type

Non-binary Relationship

Entity Relationship Model

Mapping Entity Types to Relations
- For every entity type create a relation
  \{ primary_key (E) U \{a1…am\} \}
- Every attribute in an entity becomes a relation attribute
- The relation is a subset of the X of the domains of the attributes
- Composite attributes—just include all the atomic attributes
- Derived attributes are not included but their derivation rules are
Mapping many:many Relationship Types to Relations

- Create a relation:

\[ \bigcup_{i=1}^{n} \text{primary_key}(E_i) \cup \{a_1, \ldots, a_m\} \]

primary keys of each participating entity type in the relationship

\( n \) (degree of relationship)

attributes on the relationship type (if any)

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>given family</th>
<th>tutor slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>fred jones</td>
<td>bush 12B</td>
</tr>
<tr>
<td>s2</td>
<td>mary brown</td>
<td>kahn 12B</td>
</tr>
<tr>
<td>s3</td>
<td>sue smith</td>
<td>goble 10A</td>
</tr>
<tr>
<td>s4</td>
<td>fred bloggs</td>
<td>goble 11A</td>
</tr>
<tr>
<td>s5</td>
<td>peter jones</td>
<td>goble 13B</td>
</tr>
<tr>
<td>s6</td>
<td>jill peters</td>
<td>kahn 12A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAFF</th>
<th>name</th>
<th>roomno</th>
</tr>
</thead>
<tbody>
<tr>
<td>kahn</td>
<td>IT206</td>
<td></td>
</tr>
<tr>
<td>bush</td>
<td>2.26</td>
<td></td>
</tr>
<tr>
<td>goble</td>
<td>2.82</td>
<td></td>
</tr>
<tr>
<td>zobel</td>
<td>2.34</td>
<td></td>
</tr>
<tr>
<td>watson</td>
<td>IT204</td>
<td></td>
</tr>
<tr>
<td>woods</td>
<td>2.125</td>
<td></td>
</tr>
<tr>
<td>capon</td>
<td>A14</td>
<td></td>
</tr>
<tr>
<td>lindsey</td>
<td>2.10</td>
<td></td>
</tr>
<tr>
<td>barringer</td>
<td>2.125</td>
<td></td>
</tr>
</tbody>
</table>

Mapping one:many Relationship Types to Relations

- Mostly: ‘Posting the primary key’
- Given E1 at ‘many’ end of relationship and E2 at ‘one’ end of relationship, add to the relation for E1
- Make the primary key of the entity at the ‘one’ end (the determined entity) a foreign key in the entity at the ‘many’ end (the determining entity). Include any relationship attributes with the foreign key entity

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>studno</th>
<th>given family</th>
<th>tutor slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>fred</td>
<td>jones</td>
<td>bush 12B</td>
</tr>
<tr>
<td>s2</td>
<td>mary</td>
<td>brown</td>
<td>kahn 12B</td>
</tr>
<tr>
<td>s3</td>
<td>sue</td>
<td>smith</td>
<td>goble 10A</td>
</tr>
<tr>
<td>s4</td>
<td>fred</td>
<td>bloggs</td>
<td>goble 11A</td>
</tr>
<tr>
<td>s5</td>
<td>peter</td>
<td>jones</td>
<td>goble 13B</td>
</tr>
<tr>
<td>s6</td>
<td>jill</td>
<td>peters</td>
<td>kahn 12A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAFF</th>
<th>name</th>
<th>roomno</th>
</tr>
</thead>
<tbody>
<tr>
<td>kahn</td>
<td>IT206</td>
<td></td>
</tr>
<tr>
<td>bush</td>
<td>2.26</td>
<td></td>
</tr>
<tr>
<td>goble</td>
<td>2.82</td>
<td></td>
</tr>
<tr>
<td>zobel</td>
<td>2.34</td>
<td></td>
</tr>
<tr>
<td>watson</td>
<td>IT204</td>
<td></td>
</tr>
<tr>
<td>woods</td>
<td>2.125</td>
<td></td>
</tr>
<tr>
<td>capon</td>
<td>A14</td>
<td></td>
</tr>
<tr>
<td>lindsey</td>
<td>2.10</td>
<td></td>
</tr>
<tr>
<td>barringer</td>
<td>2.125</td>
<td></td>
</tr>
</tbody>
</table>

Optional Participation of Determined Entity (‘one end’)

- SCHOOL(hons, faculty)
- STUDENT(studno, givenname, familyname, ???)
Optional Participation of Determined Entity

1. STUDENT (studno, givenname, familyname, tutor, slot)
2. STUDENT (studno, givenname, familyname)
3. same as 2 if lots of attributes on TUTOR

Optional Participation of the Determinant Entity

1. STUDENT (studno, givenname, familyname, tutor, slot)
2. STUDENT (studno, givenname, familyname)
3. same as 2 if lots of attributes on TUTOR

Mapping one:one Relationship Types to Relations

1. Post the primary key of one of the entity types into the other entity type as a foreign key, including any relationship attributes with it or
2. Merge the entity types together

Multi-Valued Attributes

- Create a relation for each multi-valued attribute { primary_key(Ei) U multi-valued attribute }
The primary key is (primary_key(Ei) U multi-valued attribute)
Mapping Roles & Recursive Relationships

- The function of an entity type in a relationship type

```
STAFF(name, roomno, ??? )
```

Multiple Roles between Entity Types

1. Treat each relationship type separately
2. Distinct roles are represented by different foreign keys drawing on the same relation

```
STAFF(name, roomno)
STUDENT(studno, given, family, ??? )
STAFF(name, roomno)
EXAMINER( ??? )
SUPERVISOR( ??? )
EXAM-SUPER( ??? )
```

Non-binary Relationship

```
COURSE(courseno, subject, equip)
STUDENT(studno, givenname, familyname)
STAFF(staffname, roomno)
TUTORS( ??? )
```

Comparative Terms

<table>
<thead>
<tr>
<th>EER</th>
<th>Formal</th>
<th>Relational</th>
<th>Informal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type Schema</td>
<td>Relational Schema</td>
<td>Table description</td>
<td></td>
</tr>
<tr>
<td>Entity Type</td>
<td>Relation</td>
<td>Table</td>
<td></td>
</tr>
<tr>
<td>Entity instance</td>
<td>Tuple</td>
<td>Row</td>
<td></td>
</tr>
<tr>
<td>1:many relationship type</td>
<td>Use foreign keys</td>
<td>Use foreign keys</td>
<td></td>
</tr>
<tr>
<td>1:many relationship instance</td>
<td>Use foreign keys</td>
<td>Use foreign keys</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Attribute</td>
<td>Column</td>
<td></td>
</tr>
<tr>
<td>Domain or Value Set</td>
<td>Domain</td>
<td>Data Type</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Candidate Key</td>
<td>Candidate Key</td>
<td></td>
</tr>
<tr>
<td>No equivalent</td>
<td>Primary Key</td>
<td>Primary Key</td>
<td></td>
</tr>
<tr>
<td>Multivalued attribute</td>
<td>No equivalent</td>
<td>No equivalent</td>
<td></td>
</tr>
<tr>
<td>Composite attribute</td>
<td>No equivalent</td>
<td>No equivalent</td>
<td></td>
</tr>
</tbody>
</table>

Superclasses, Subclasses; Specialisation & Generalisation Relationships

- Subclasses and Superclasses
  - a subclass entity type is a specialised type of superclass entity type
  - a subclass entity type represents a subset or subgrouping of superclass entity type's instances
  - e.g. undergraduates and postgraduates are subclasses of student superclass
- Attribute Inheritance
  - subclasses inherit properties (attributes) of their superclasses

Constraints on Specialisation & Generalisation

- Specialisation
  - the process of defining a set of more specialised entity types of an entity type
- Generalisation
  - the process of defining a generalised entity type from a set of entity types
- Predicate/Condition defined
  - determine the entities that will become members of each subclass by a condition on an attribute value. All member instances of the subclass must satisfy the predicate
  - e.g. first years and second years are subclasses of undergraduates based on their year attribute.
- User defined
  - no condition for determining subclass membership
Constraints on Specialisation & Generalisation

- **Disjointness**
  - **Overlap**
    - the same entity instance may be a member of more than one subclass of the specialisation
  - **Disjoint**
    - the same entity instance may be a member of only one subclass of the specialisation

- **Completeness**
  - **Total**
    - every entity instance in the superclass must be a member of some subclass in the specialisation
  - **Partial**
    - an entity instance in the superclass need not be a member of any subclass in the specialisation

Superclasses, Subclasses

Specialisation & Generalisation Relationships

**Categories and Categorisation**

- a single superclass/subclass relationship with more than one superclass, where the superclasses represent different entity types (sometimes with different keys)

Specialisation & Generalisation Option A

1. Create a relation for superclass
2. Create a relation for each subclass such that:
   - (primary_key of superclass) U (attributes of subclass)
   - key for subclass is (primary_key of superclass)

   \[
   \begin{align*}
   \text{Inclusion dependency:} & \quad \pi_{\text{super}}(\text{superclass}) \supseteq \pi_{\text{sub}}(\text{subclass}) \\
   \text{Covering dependency:} & \quad n \ (\text{number of subclasses}) \quad \pi_{\text{sub}}(\text{subclass}) = \pi_{\text{super}}(\text{superclass}) \\
   \text{Disjoint dependency:} & \quad n \ (\text{number of subclasses}) \quad \bigcap_{i=1}^{n} \pi_{\text{sub}}(\text{subclass}) = \emptyset
   \end{align*}
   \]
Specialisation & Generalisation Option B
1. Create a relation for each subclass such that:
   (primary_key \ U \ {attributes} \ {attributes of superclass}) (subclasses)
key for each relation is (primary_key of superclass)

- Works for total and disjoint constraints
- Partial: lose any entity that is not in a subclass
- Overlapping: redundancy
- To recover the superclass can do an OUTER UNION on the subclass relations

Specialisation & Generalisation Option C
1. Create one relation such that:
   (primary_key \ U \ {attributes} \ {attributes} \ {type of superclass} \ of all subclasses \ attribute)
   key for subclass is (primary_key of superclass)

- Many 'not-applicable' nulls
- Disjoint: one type which indicates which subclass the tuple represents
- Overlap: set of types = number of subclasses
- Partial: type is null: represents superclass

Specialisation & Generalisation Overlapping
1. STAFF(payrollno, name, lengthofservice)
   ACADEMIC(payrollno, level)
   TECHNICAL(payrollno, project)
   ADMIN(payrollno, grade)
2. ACADEMIC(payrollno, name, lengthofservice, level)
   TECHNICAL(payrollno, name, lengthofservice, project)
   ADMIN(payrollno, name, lengthofservice, grade)
3. STAFF(payrollno, name, lengthofservice, level, 
   project, grade, type1, type2, type3)
   STAFF(payrollno, name, lengthofservice, level, 
   project, grade, type)
   \text{type} = \text{powerset of classes}

Specialisation & Generalisation Relationships

Specialisation Lattice with Shared Subclass
- To be a shared subclass the superclasses must have the same key, so any of the options A, B or C stand.

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key

Categories and Categorisation
- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new surrogate key
Entity Constraints

- If an entity instance X depends on the existence of an entity instance Y, then X is existence dependent on Y.
- Entity type Y is dominant.
- Entity type X is subordinate.

<table>
<thead>
<tr>
<th>customer</th>
<th>orderid</th>
<th>date</th>
<th>part</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>RipOff Inc</td>
<td>123</td>
<td>23/6/94</td>
<td>widget</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thingie</td>
<td>24</td>
</tr>
<tr>
<td>RipOff Inc</td>
<td>678</td>
<td>3/10/94</td>
<td>widget</td>
<td>20</td>
</tr>
<tr>
<td>UpYa Ltd</td>
<td>123</td>
<td>27/9/94</td>
<td>wotsits</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>widget</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thingie</td>
<td>24</td>
</tr>
</tbody>
</table>

Strong and Weak Entities (identifier dependency)

- A strong entity type has an identifying primary key.
- A weak entity type does not have a primary key but does have a discriminator.

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER</td>
<td>date</td>
</tr>
</tbody>
</table>

Mapping Weak Entities to Relations

- Create a relation

\[ \bigcup_{i=1}^{n} \text{primary_key}(E_i) \cup \text{partial_key} U (a_i \ldots a_n) \]

- Primary key of each participating identifying entity type.
- Partial key of weak entity (if any).
- Attributes of the weak entity type (if any).

Association Entity Type

- An entity type that represents an association relationship type.
- Useful if:
  - A relationship has lots of attributes.
  - You want a relationship type with a relationship type.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>ENROL</th>
<th>COURSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAFF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Association Entity Type plus Mapping

- An entity type that represents an association relationship type.

<table>
<thead>
<tr>
<th>STAFF</th>
<th>ENROL</th>
<th>COURSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT(studno, givenname, familyname)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COURSE(courseno, subject, equip)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Aggregation

- Aggregation is an abstraction concept for building composite entities from their components
  1. aggregate attribute values to form a whole entity
  2. combining entities that are related by an association relationship into higher-level aggregate entity
- IS-A-PART-OF
- IS-A-COMPONENT-OF
- Sadly, not catered for in EER modelling.

Hints for EER Modelling

- identify entity types by searching for nouns and noun phrases
- assume all entities are strong and check for weak ones on a later pass
- need an identifier for each strong entity
- assume all relationships are partial participation (optional) and check for total (mandatory) ones on a later pass
- expect to keep changing your mind about whether things are entities, relationships or attributes
- keep level of detail relevant and consistent (for example leave out attributes at first)
- approach diagram through different views and merge them

Lets Practice!

- A record company wishes to use a computer database to help with its operations regarding its performers, recordings and song catalogue.
- Songs have a unique song number, a non-unique title and a composition date. A song can be written by a number of composers; the composer’s full name is required. Songs are recorded by recording artists (bands or solo performers). A song is recorded as a track of a CD. A CD has many songs on it, called tracks. CDs have a unique record catalogue number, a title and must have a producer (the full name of the producer is required). Each track must have the recording date and the track number of the CD.
- A song can appear on many (or no) CDs, and be recorded by many different recording artists. The same recording artist might re-record the same song on different CDs. A CD must have only 1 recording artist appearing on it. CDs can be released a number of times, and each time the release date and associated number of sales is required.