Conceptual Database Design

Jan Chomicki
University at Buffalo

Outline

- Entity-Relationship Data Model
- Mapping E-R schemas to relations
- Description logics
Entity-Relationship (E-R) Data Model

Proposed by Peter Chen in 1976.

**Features**
- used for the description of the conceptual schema of the database
- not used for database implementation
- formal notation
- close to natural language

**Can be mapped to various data models**
- relational
- object-oriented, object-relational
- XML
- description logics

**Basic ER model concepts**

<table>
<thead>
<tr>
<th>Schema level</th>
<th>Instance level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Domain element (value)</td>
</tr>
<tr>
<td>Entity type</td>
<td>Entity</td>
</tr>
<tr>
<td>Relationship type</td>
<td>Relationship (instance)</td>
</tr>
<tr>
<td>Cardinality constraints</td>
<td>Valid relationships</td>
</tr>
<tr>
<td>Attribute</td>
<td>Attribute value</td>
</tr>
<tr>
<td>Key</td>
<td>Unique key value</td>
</tr>
</tbody>
</table>
Entities

**Entity**
Something that *exists* and can be *distinguished* from other entities.

**Examples**
A person, an account, a course.

**Entity type**
A set of entities with similar *properties*. Entity types can overlap.

**Examples**
Persons, employees, Citibank accounts, UB courses.

**Entity type extension**
The set of entities of a given type in a given database instance.

**Notation**
- entities: $e_1, e_2, \ldots$
- “entity $e$ is of type $T$”: $T(e)$.

Attributes

**Domain**
A predefined set of primitive, atomic values (entity types are *not* domains!).

**Examples**
Integers, character strings, decimals.

**Attribute**
A (partial) function from an entity type to a domain, representing a property of the entities of that type.

**Examples**
Name: Person $\rightarrow$ String
Balance: Account $\rightarrow$ Decimal

**Notation**
- $A(e)$: “the value of the attribute $A$ for the entity $e$”.

**Example**
Name($e_1$) = ’Brown’
Keys

**Key**
A (minimal) set of attributes that uniquely identifies every entity in an entity type.

**Examples**

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Americans</td>
<td>SSN</td>
</tr>
<tr>
<td>ATT accounts</td>
<td>Phone number</td>
</tr>
<tr>
<td>NY vehicles</td>
<td>License plate number</td>
</tr>
<tr>
<td>US vehicles</td>
<td>(License plate number, State)</td>
</tr>
</tbody>
</table>

- an entity type can have **multiple keys**
- one key is selected as the **primary key**.

Relationships

**Relationship type of arity k**
A subset of the Cartesian product of some entity types $E_1, \ldots, E_k$, representing an association between the entity types. Relationship types can have attributes.

**Examples**

- Teaches(Employee, Class)
- Sells(Vendor, Customer, Product)
- Parent(Person, Person)

**Relationship instance of arity k**
A $k$-tuple of entities of the appropriate types.

**Example**

Teaches($e_1, c_1$) where Employee($e_1$) and Class($c_1$) and Name($e_1$)= ’Brown’. 
Cardinality constraints

Binary relationship type $R(A, B)$ is:

- $\boxed{1 : 1}$ if for every entity $e_1$ in $A$ there is at most one entity $e_2$ in $B$ such that $R(e_1, e_2)$ and vice versa.
- $\boxed{N : 1}$ if for every entity $e_1$ in $A$ there is at most one entity $e_2$ in $B$ such that $R(e_1, e_2)$.
- $\boxed{N : M}$ otherwise.

Advanced schema-level concepts

- isa relationships
- weak entity types
- complex attributes
- roles.
**Isa relationships**

**Definition**

An entity type \( A \) is *isa* an entity type \( B \) if every entity in the entity type \( A \) is also in the entity type \( B \).

**Example**

Faculty *isa* Employee.

If \( A \) *isa* \( B \), then:

- \( \text{Attrs}(B) \subseteq \text{Attrs}(A) \) (inheritance of attributes),
- \( \text{Key}(A) = \text{Key}(B) \) (inheritance of key).

**Example**

Rank : Faculty \( \rightarrow \) \{’Assistant’, ’Associate’, …\}

Rank is not defined for non-faculty employees (or defined differently).

---

**Weak entity types**

**Definition**

An entity type \( A \) is a weak entity type if:

- \( A \) does not have a key.
- the entities in \( A \) can be identified through an identifying relationship type \( R(A, B) \) with another entity type \( B \).

The entities in \( A \) can be identified by the combination of:

- the *borrowed* key of \( B \).
- some *partial* key of \( A \).

**Example**

Entity types: Account, Check.
Identifying relationship type: Issued.
Borrowed key (of Account): AccNo.
Partial key (of Check): CheckNo.
Complex attributes

Attribute values
- sets (multivalued attributes).
- tuples (composite attributes).

Multivalued attribute
Degrees : Faculty → 2\{′B.A.′,′B.S.′,...,′Ph.D.′,...\}

Composite attribute
Address : Employee → Street × City × Zipcode

Multivalued and composite attributes can be expressed using other constructs of the E-R model.

Roles

Roles are necessary in a relationship type that relates an entity type to itself. Different occurrences of the same entity type are distinguished by different role names.

Example
In the relationship type ParentOf(Person, Person) the introduction of role names gives ParentOf(Parent:Person, Child:Person)
ER design

General guidelines
- schema: stable information, instance: changing information.
- avoid redundancy (each fact should be represented once).
- no need to store information that can be computed.
- keys should be as small as possible.
- introduce artificial keys only if no simple, natural keys available.

How to choose entity types
- things that have properties of their own, or
- things that are used in navigating through the database.
- avoid null attribute values if possible by introducing extra entity types.

isa relationship design

Generalization (bottom-up)
- generalize a number of different entity types (with the same key) to a single type.
- factor out common attributes.

Example
Student isa Person
Teacher isa Person
Name : Person → String

Specialization (top-down)
- specialize an entity type to one or more specific types.
- add attributes in more specific entity types.

Example
Salary : Teacher → Decimal
Mapping E-R schemas to relations

Assumption
No complex attributes.

Multiple stages
- creating relation schemas from entity types.
- creating relation schemas from relationship types.
- identifying keys.
- identifying foreign keys.
- schema optimization.

Mapping entity types to relations

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Relation schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1$ such that $E_1$ isa $E_2$</td>
<td>$Key(E_2)$</td>
</tr>
<tr>
<td></td>
<td>$\cup (\text{Attr}(E_1) - \text{Attr}(E_2))$</td>
</tr>
<tr>
<td>$E_1$ is a weak entity type identified by $R(E_1, E_2)$</td>
<td>$Key(E_2)$</td>
</tr>
<tr>
<td></td>
<td>$\cup (\text{Attr}(E_1) - \text{Attr}(E_2))$</td>
</tr>
<tr>
<td>$E_1$ is none of the above</td>
<td>$\text{Attr}(E_1)$</td>
</tr>
</tbody>
</table>
Mapping relationship types to relations

<table>
<thead>
<tr>
<th>Relationship type</th>
<th>Relation schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(E_1, \ldots, E_n)$</td>
<td>$\text{Key}(E_1) \cup \cdots \text{Key}(E_n)$ $\cup \text{Attrs}(R)$</td>
</tr>
</tbody>
</table>

No relations are created from isa or identifying relationships.

Different occurrences of the same attribute name should be named differently.

Identifying keys

Relation schema $W$ is the result of mapping an entity type $E_1$ or a relationship type $R(E_1, E_2)$.

<table>
<thead>
<tr>
<th>Source of $W$</th>
<th>Key of $W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type $E_1$</td>
<td>$\text{Key}(E_1)$</td>
</tr>
<tr>
<td>Weak entity type $E_1$</td>
<td>Union of borrowed and partial keys of $E_1$</td>
</tr>
<tr>
<td>$R(E_1, E_2)$ is 1:1</td>
<td>$\text{Key}(E_1)$ or $\text{Key}(E_2)$</td>
</tr>
<tr>
<td>$R(E_1, E_2)$ is N:1</td>
<td>$\text{Key}(E_1)$</td>
</tr>
<tr>
<td>$R(E_1, E_2)$ is N:M</td>
<td>$\text{Key}(E_1) \cup \text{Key}(E_2)$</td>
</tr>
</tbody>
</table>

These rules can be generalized to arbitrary relationship types $R(E_1, \ldots, E_n)$. 
Identifying foreign keys

Relation schema $W$ is the result of mapping an entity type $E_1$ or a relationship type $R(E_1, E_2)$.

<table>
<thead>
<tr>
<th>Source of $W$</th>
<th>Foreign keys of $W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type $E_1$</td>
<td>No foreign keys</td>
</tr>
<tr>
<td>Weak entity type $E_1$</td>
<td>Borrowed key of $E_1$</td>
</tr>
<tr>
<td>Entity type $E_1$</td>
<td>Key($E_1$)</td>
</tr>
<tr>
<td>such that $E_1$ isa $E_2$</td>
<td></td>
</tr>
<tr>
<td>$R(E_1, E_2)$</td>
<td>Key($E_1$), Key($E_2$)</td>
</tr>
</tbody>
</table>

Schema optimization

Combine relation schemas with *identical* keys coming from *the same* entity type.

Student($SName$, Address) can be combined with Advising($SName$, Faculty) to yield Student($SName$, Address, Faculty).

Different keys

Student($SName$, Address) **should not** be combined with Grades($SName$, Course, Grade).

Different entity types

Student($SName$, Address) **should not** be combined with Graduate($SName$).
Description logics knowledgebases

Description logics
- a family of variable-free logics developed in AI
- used to define ontologies for the Semantic Web (OWL DL)

Terminological box (TBox)
- corresponds to database conceptual schema
- vocabulary: atomic concepts and roles
- containment and transitivity assertions, definitions

Assertional box (ABox)
- corresponds to database instance
- named individuals
- assertions stating membership of individuals in concepts and roles

Concepts

Atomic concepts
- correspond to entity types

Singleton concepts
- the concept consists of a single individual: \{a\}

Boolean concepts
- intersection of concepts: \( C \cap D \)
- union of concepts: \( C \cup D \)
- negation of a concept: \( \neg C \)
- top concept: \( \top = A \cup \neg A \)
- bottom concept: \( \bot = A \cap \neg A \)
Further concepts

Quantification and number restriction

- $C$ is a concept, $R$ a role
- *individuals associated with some individual in $C$ through $R$: $\exists R.C$
- *individuals associated only with individuals in $C$ through $R$: $\forall R.C$
- *individuals associated with at most $k$ individuals through $R$: $\leq k R$
- *individuals associated with at least $k$ individuals through $R$: $\geq k R$

Datatypes

In $\exists R.C$ and $\forall R.C$, $C$ can be a datatype (Integer, String,...).

Roles

Atomic roles

- correspond to relationship types

Inverse roles

- an individual $a$ is associated with an individual $b$ through $R^-$ if and only if $b$ is associated with $a$ through $R$. 
Assertions

Definition
- atomic concept $A$ is defined as concept $C$: $A \equiv C$

Containment
- concept $C$ is contained in concept $D$: $C \subseteq D$
- role $R$ is contained in role $S$: $R \subseteq S$

Transitivity
- role $R$ is transitive: $R^+ \subseteq R$

Membership
- individual $a$ is a member of concept $C$: $a \in C$
- pair $(a, b)$ belongs to role $R$: $(a, b) \in R$

E-R constructs in description logics

Integer attribute $A$ for entity type $E$
- $E$ is a concept, $A$ is a role
- assertion:
  $$E \sqsubseteq \forall A.\text{Integer} \sqcap \leq 1 \sqsubseteq A$$

Relationship $R$ is between entity types $E_1$ and $E_2$
- $E_1$ and $E_2$ are concepts, $R$ is a role
- assertions:
  $$E_1 \sqsubseteq \forall R.E_2$$
  $$E_2 \sqsubseteq \forall R^- .E_1$$
Further E-R constructs

Relationship $R$ is $n : 1$

- assertion:
  \[ E_1 \sqsubseteq \leq 1 \ R \]

$E_1$ isa $E_2$

- assertion:
  \[ E_1 \sqsubseteq E_2 \]

Problematic constructs

- keys
- $n$-ary relationships for $N > 2$ (but can be simulated)

Beyond E-R: ontologies

Concepts $C_1$ and $C_2$ are disjoint

- assertion:
  \[ C_1 \cap C_2 \sqsubseteq \bot \]

Single parents

- assertion:
  \[ SingleParent \equiv Person \cap (\forall Parent. \leq 1Parent^-) \]

Typical ontology reasoning tasks

- correctness of knowledge: does the knowledgebase imply a given containment assertion?
- querying ontologies: does the knowledgebase imply a given membership assertion?