Conceptual Database Design

Jan Chomicki
University at Buffalo

Outline

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

Schema level	*Instance level*
Domain	Domain element (value)
Entity type	Entity
Relationship type	Relationship (instance)
Cardinality constraints	Valid relationships
Attribute	Attribute value
Key	Unique key value

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` $\longrightarrow$ `String`
`Balance` : `Account` $\longrightarrow$ `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

Entity type	**Key**
Americans	SSN
ATT accounts	Phone number
NY vehicles	License plate number
US vehicles	(License plate number,State)

- 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:

- 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*.
- N : 1 if for every entity e_1 in A there is at most one entity e_2 in B such that $R(e_1, e_2)$.
- N : M otherwise.

Advanced schema-level concepts

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

isa relationships

Definition

A isa 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:

- $Attrs(B) \subseteq Attrs(A)$ (*inheritance* of attributes),
- $Key(A) = Key(B)$ (*inheritance* of key).

Example

`Rank` : `Faculty` $\rightarrow$ {'Assistant','Associate',...}

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

Weak entity types

Definition

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

$\texttt{Degrees}: \texttt{Faculty} \longrightarrow 2^{\{'\mathrm{B.A.}','\mathrm{B.S.}',\ldots,'\mathrm{Ph.D.}',\ldots\}}$

Composite attribute

$\texttt{Address}: \texttt{Employee} \longrightarrow \texttt{Street} \times \texttt{City} \times \texttt{Zipcode}$

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 $\longrightarrow$ String

Specialization (top-down)

- specialize an entity type to one or more specific types.
- add attributes in more specific entity types.

Example

Salary : Teacher $\longrightarrow$ Decimal

Mapping E-R schemas to relations

Assumption

No complex attributes.

Multiple stages

1. creating relation schemas from entity types.
2. creating relation schemas from relationship types.
3. identifying keys.
4. identifying foreign keys.
5. schema optimization.

Mapping entity types to relations

Entity type	*Relation schema*
E_1 such that E_1 **isa** E_2	$Key(E_2)$ $\cup(Attrs(E_1) - Attrs(E_2))$
E_1 is a weak entity type identified by $R(E_1, E_2)$	$Key(E_2)$ $\cup(Attrs(E_1) - Attrs(E_2))$
E_1 is none of the above	$Attrs(E_1)$

Mapping relationship types to relations

Relationship type	*Relation schema*
$R(E_1, \ldots, E_n)$	$Key(E_1) \cup \cdots Key(E_n)$ $\cup Attrs(R)$

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 $R(E_1, E_2)$.

Source of W	*Key of W*
Entity type E_1	$Key(E_1)$
Weak entity type E_1	Union of borrowed and partial keys of E_1
$R(E_1, E_2)$ is $1:1$	$Key(E_1)$ or $Key(E_2)$
$R(E_1, E_2)$ is $N:1$	$Key(E_1)$
$R(E_1, E_2)$ is $N:M$	$Key(E_1) \cup Key(E_2)$

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 $R(E_1, E_2)$.

Source of W	*Foreign keys of W*
Entity type E_1	No foreign keys
Weak entity type E_1	Borrowed key of E_1
Entity type E_1 such that E_1 **isa** E_2	$Key(E_1)$
$R(E_1, E_2)$	$Key(E_1)$, $Key(E_2)$

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: $C \sqcap D$
- union of concepts: $C \sqcup D$
- negation of a concept: $\neg C$
- top concept: $\top = A \sqcup \neg A$
- bottom concept: $\bot = A \sqcap \neg A$

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*: $C \sqsubseteq D$
- *role R is contained in role S*: $R \sqsubseteq S$

Transitivity

- role R is transitive: $R^+ \sqsubseteq R$

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:

$$E \sqsubseteq \forall A.Integer \sqcap \leq 1\ A$$

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:

$$C_1 \sqcap C_2 \sqsubseteq \bot$$

Single parents

- assertion:

$$SingleParent \equiv Person \sqcap (\forall Parent. \leq 1 Parent^{-})$$

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?*