2 Conceptual Database Design

2.3 Integrity Constraints

2.3.1 Constraint types

2.3.2 Cardinality constraints

2.4 Extended ER Modeling

2.4.1 Inheritance / Generalization

2.4.2 Modeling historical data

2.4.3 N-ary relationships

Bernstein et al.: chap. 4; Elmasri, Navathe: chap 3 + chap 4;
Kemper, Eickler: 2.7 – 2.13

---

2.3.1 Integrity Constraints

**Def.: An Integrity constraint** is an invariant (assertion, restriction) of the state of a database. ICs are predicates, a database must fulfill during its lifetime.

They result from requirement analysis, context and common sense knowledge.

Formally stated in DB schema

Case study

From requirements

"Names of regions are not necessarily unique"

"A region belongs to exactly one country"

Common sense knowledge

"Population is always >= 0 - or unknown"

"A country has one and only one capital"

---

2.3.2 Cardinality constraints

**Def.: A cardinality constraint** of a relationship R between entity types E1, E2 restricts the number of entities E1, E2 participating R.

UML terminology: multiplicity

---

Conceptual Design: case study

Constraints ??

Constraint types

Attribute constraints

- Attribute value restriction
- Attribute value must / may exist ([not] NULL)

General constraints

- Relations may be symmetric
e.g. neighbor-rel of countries

Cardinality constraints

How many entities of type E may be in relationship R to an entity of type E’?
e.g. to how many countries can a region belong?
How many regions can a country have?

---

Cardinality constraints, N:M notation

Examples

contradicts 1 : N, not allowed

contradicts arbitrary binary relation
**1:N relationship**

*Graphical Notation* with symbolic cardinalities

```
  E1 1 R N E2
```

One E1-entity is related (R) to arbitrary many E2-entities, but one E2-entity is related (R) to only one E1-entity.

Traditional ER-M notation for cardinality constraints:

```
region 1 locatedIn N city
```

Formally: locatedIn:: city -> region is a function.

**More relationships**

*M:N-Relationships*

every instance of E1 may be related according to R to every instance of E2.

```
E1 M R N E2
```

R is M:N means: no restriction on the pairs of R.

**1:1-Relationships**

every instance of E1 may be related according to R to exactly one instance of E2 and vice versa.

```
E1 1 R 1 E2
```

**((min,max))-Notation**

More precise cardinality restrictions by specifying minimal and maximal number of entities.

Many cities locatedIn one country, at least one

\[ \Rightarrow \min=1, \max=\ast \]

A country has zero, one or many neighbors

\[ \Rightarrow \min=0, \max=\ast \]

\((\min,\max)\)-Cardinality constraint (multiplicity) notation also used in UML associations.

**CAVEAT: Misleading Notation**

Traditional ER-Model, \((\min,\max)\)-Notation does not conform to N:M-Notation.

You find this in many text books, 1:N and (min,max) interchanged.

UML-multiplicity conformant to 1:N notation:

```
Customer min1 max1 0* account
```

\[ \min=1, \max=\ast \]

Use \((\min,\max)\) annotation which conforms to UML, \[ \min,\max \in \{0,1,*\} \]

Cardinality of weak entities

\[ e \] is existentially dependent on \(e'\)

Cardinality:

\[ \min1 = \max1 = 1 \]

\[ \min2 = 0 \mid 1 \]

\[ \max2 = 1 \mid \ast \]
2 Conceptual Database Design

2.3 Integrity Constraints

2.3.1 Constraint types

2.3.2 Cardinality constraints

2.4 Extended ER Modeling

2.4.1 Inheritance / Generalization

2.4.2 Modeling historical data

2.4.3 N-ary relationships

Bernstein et al.: chap. 4; Elmasri, Navathe: chap 3 + chap 4; Kemper, Eickler: 2.7 – 2.13

Weak entity: example

Countries and regions

Orders and order items

Sometimes an artificial key makes sense, not in this case

Cardinality constraints: semantics

Let \( R \subseteq E_1 \times E_2 \) be a relationship between entity sets \( E_1 \) and \( E_2 \)

\[
\text{E1-R has (min1, max1) cardinality} \iff \text{for all extensions of } R \text{ and for all } y_0 \in E_2 \text{ min1} \leq |\{x | x \in E_1 \land (x, y_0) \in R\}| \leq \text{max1}
\]

\[
\text{E2-R has (min2, max2) cardinality} \iff \text{for all extensions of } R \text{ and for all } x_0 \in E_1 \text{ min2} \leq |\{y | y \in E_2 \land (x_0, y) \in R\}| \leq \text{max2}
\]

Cardinality constraints notations

<table>
<thead>
<tr>
<th>mandatory/ multiple</th>
<th>optional/ multiple</th>
<th>optional/ single</th>
<th>mandatory/ single</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM / (UML)</td>
<td>(1,*)</td>
<td>(0,*)</td>
<td>(0,1)</td>
</tr>
<tr>
<td>1:N</td>
<td>N or M</td>
<td>N or M</td>
<td>1</td>
</tr>
<tr>
<td>UML^</td>
<td>(k)</td>
<td>(0..*)</td>
<td>(0..1)</td>
</tr>
</tbody>
</table>

+ : k and j are natural numbers; n, N,M in the ERM are literals

Many more notations in use!, eg. Oracle ‘crow’s feet’-Notation

Conceptual Design: case study

UML class diagram

No operations in conceptual DB model

from Wikipedia (Klassendiagramm)
2.4 Extended ER (EER)

Example:
Suppose two types of customers of a video-shop:
- frequent customers
- regular customers

Redundant: relationships of Customer has to be duplicated for FreqCustomer

employ object oriented principle of generalization/inheritance

Generalization: Example

Customer           FreqCustomer
memberShip: Number  membership: Number
name: Name         name: Name...
first_name: Name   credit: Money
address: A_type    address: A_type
(phone: Phone_Type) (phone: Phone_Type)

Generalization / specialization

Factorize common attributes of different entities

Publication
internalID: String
issn: String
(editor: Name)
(author: Name)

Book
isbn: String
publisher: Name
edition: Number

Journal
volume: Number
issue: Number

is-a

Standard relationship is-a between subtypes and super types

Semantics of generalization: type versus set
Instances of A, B and C are different but share some attributes (OO-interpretation)
All instances of B and of C are also instances of A (DB interpretation)

B ⊆ A and C ⊆ A

B ⊆ T1

A ⊆ T2

C

Def.: Specialization is called

- disjoint iff C ∩ B = ∅
- complete
ifu A = B ∪ C, and every tuple is either B or C

more general definition: m≥2 specializations

No overwriting... why not?

2.4.1 Modeling historical data

Time invariant:
a particular relationship between e1 and e2 will never change.

Time variant:
a particular relationship (c1, v1) may disappear, a new one may be established

In many cases:
History of time variant relationships has to be recorded
Case study and historical data

Keeping track of changes...

Use case: Bike rental

A bike may be rented by many customers...

... but not at the same time

Conceptual Modeling: historical data

Solution:

Introduce a weak entity which keeps track of related entities over time (here: rental of each particular bike over time)

Question: Why ‘Rental’ existentially dependent on bike, not customer?

2.4.2 N-ary relationships

Motivation example

Represent the following facts in a database:
supplier X delivers part Y to project Z
supplier A delivers part P to project Z
supplier B delivers part Q to project S

Wrong: Conceptual model does NOT represent the information given above

N-ary relationships

Def.: A relationship is call n-ary relationship \( R \), if more than 2 entity sets are involved in the \( R \)

N-ary relations and cardinalities

Def.: \( E1 \)-R has (min1, max1) cardinality

\( \Rightarrow \) for all extensions of \( R \) and for all \( \{ y,z \} \in E2 \times E3 \)

\( \min1 \leq |\{ x \in E1 \wedge \{ x,y,z \} \in R \}| \leq \max1 \)

E2-R, E3-R correspondingly.

N-ary relationships

Example:

- Employees assigned to a project, work at one location for this project.
- Employees work for one project at a particular location
- At each location several employees may work for a particular project

Question: May an employee work for different projects?
Which constraints cannot be expressed?
**N-ary relationships by N binary relationships**

Introduce a weak entity type for the relationship and binary relationships to the other entity types.

**Extended ER: Aggregation**

Different constraints expressed than n-ary relationship

**Conceptual Design**

Def.: View integration is the process of integrating conceptual models, which are related but have been designed separately, into one single model.

For big projects different "views" of the application make sense: model different, more or less independent parts of the "real world". (compare "partitioning approach" to "top down approach")

Important: model data and processes the data are used for e.g. student administration, exams, teachers and human resources

**View integration**

Integrate different partial designs into the conceptual design of the overall DB

Running example:
- a) countries, cities...
- b) Organizations (Government, national / internat. organization
- c) geography: lakes, mountains, rivers...

Not as easy as it sounds…

**DB design and constraints**

Constraints
- Restrict the state of the database
- Database should always be coherent with real world

Types of constraints
- Value restriction
- Cardinality restriction

1:N notation imprecise but sufficient in many situations

Uniform modeling "patterns"
- Historical / time related data
- N-ary relationships: model with binary relationships and a another entity type
- Generalization