The Entity/Relationship (ER) Model & DB Design

Manos Papagelis

Overview

- The Entity/Relationship (ER) Model
- Designing a database schema
  - Restructuring of an E/R model
  - Translation of an E/R model into the logical model (DB Schema)

THE ENTITY/RELATIONSHIP (ER) MODEL

Conceptualizing the real-world

Modeling is about mapping entities and relationships of the world into the concepts of a database

application
(the “real world”)

Mapping is Not Deterministic

The Relational Model uses relations to represent entities, relationships, or combinations thereof

The mapping process is not always clear

Entity/Relationship Model

- Visual data model (diagram-based)
  - Quickly "chart out" a database design
  - Easier to "see" big picture
  - Comparable to class diagrams in UML
- Basic concept: *entities* and their *relationships*, along with the *attributes* describing them

Entity Sets

- An entity set represents a class of objects that have properties in common and an autonomous existence (e.g., City, Department, Employee, Sale)
- An entity is an instance of an entity set (e.g., Stockholm is a City; Peterson is an Employee)

Relationship Sets

- A relationship set is an association between 2+ entity sets (e.g., Residence is a relationship set between entity sets City and Employee)
- A relationship is an instance of a n-ary relationship set (e.g., the pair <Johanssen, Stockholm> is a relationship instance of Residence)

Recursive Relationships

- Recursive relationships relate an entity to itself
- Note in the second example that the relationship is not symmetric
  - In this case, it is necessary to indicate the two roles that the entity plays in the relationship

Ternary Relationships

Attributes

- Describe elementary properties of entities or relationships (e.g., Surname, Salary and Age are attributes of Employee)
- May be single-valued, or multi-valued

Composite Attributes

- composite attributes are grouped attributes of the same entity or relationship that have closely connected meaning or uses

Example Schema with Attributes

Cardinalities

- Each entity set participates in a relationship set with a minimum (min) and a maximum (max) cardinality
- Cardinalities constrain how entity instances participate in relationship instances
- Graphical representation in E/R Diagrams: pairs of (min, max) values for each entity set

An entity might not participate in any relationship

Cardinalities (cont.)

- In principle, cardinalities are pairs of non-negative integers (n, N) such that $n \leq N$, where n represents minimum and N represents maximum cardinality
- minimum cardinality n:
  - If 0, entity participation in a relationship is optional
  - If 1, entity participation in a relationship is mandatory
- maximum cardinality N:
  - If 1, each instance of the entity is associated at most with a single instance of the relationship
  - If N, then each instance of the entity is associated with many instances of the relationship

Cardinality Examples

Multiplicity of relationships

If entities E1 and E2 participate in relationship R with cardinalities (n1, N1) and (n2, N2) then the multiplicity of R is N1-N2 (= N2-N1)

1-1 ⊂ N-1 OR 1-N ⊂ N-N

Cardinalities of Attributes

- Describe min/max number of values an attribute can have
- When the cardinality of an attribute is (1, 1) it can be omitted (single-valued attributes)
- The value of an attribute, may also be null, or have several values (multi-valued attributes)

Cardinalities of Attributes (cont.)

- Multi-valued attributes often represent situations that can be modeled with additional entities. E.g., the ER schema of the previous slide can be revised into:

Keys in E/R

- Keys consist of minimal sets of attributes which identify uniquely instances of an entity set
  - socialInsurance# may be a key for Person
  - firstName, middleName, lastName, address may be a key for Person
- In most cases, a key is formed by one or more attributes of the entity itself (*internal* keys)
- Sometimes, other entities are involved in the identification (*foreign keys*, *weak entities*)
- A key for a relationship consists of keys of entities it relates

Examples of Keys in E/R

Schema with Keys

Challenge: modeling the "real world"

- Life is arbitrarily complex
  - Directors who are also actors? Actors who play multiple roles in one movie? Animal actors?
- Design choices: Should a concept be modeled as an entity, an attribute, or a relationship?
- Constraints on the ER Model: A lot of data semantics can be captured but some cannot
- Key to successful model: parsimony
  - As complex as necessary, but no more
  - Choose to represent only "relevant" things

EXAMPLE

From real world to E/R Model

We wish to create a database for a company that runs training courses. For this, we must store data about trainees and instructors. For each course participant (about 5,000 in all), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for each course. We need also to represent the seminars that each participant is attending at present and, for each day, the places and times the classes are held.

Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants. If a trainee is self-employed, we need to know her area of expertise, and, if appropriate, her title. For somebody who works for a company, we store the level and position held. For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or freelance.

From real world to E/R Model

We wish to create a database for a company that runs training courses. For this, we must store data about the *trainees* and the *instructors*. For each *course participant* (about 5,000), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for each course. We need also to represent the *seminars* that each participant is attending at present and, for each day, the places and times the classes are held.

Each *course* has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants. If a trainee is self-employed, we need to know her area of expertise, and, if appropriate, her title. For somebody who works for a company, we store the level and position held. For each *instructor* (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the *tutor* is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or freelance.

More Annotations

We wish to create a database for a company that runs training courses. For this, we must store data about *trainees* and *instructors*. For each *course participant* (about 5,000), identified by a code, we want to store her *social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed)*, courses attended (there are about 200 courses) and the final assessment for each course. We need also to represent *seminars* that each participant is attending at present and, *for each day, the places and times the classes are held.*

Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For *each edition, we represent the start date, the end date, and the number of participants.* If a trainee is self-employed, we need to know her area of expertise, and, if appropriate, her title. For somebody who works for a company, we store the level and position held. For each *instructor* (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the *tutor* is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or freelance.

… the E/R model result

DESIGNING A DATABASE SCHEMA

Designing a Database Schema

The "real world"

The E/R Model (Conceptual Model)

The Relational Schema

Part (Name,Description,Part#)
Supplier (Name, Addr)
Customer (Name, Addr)
Supplies (Name,Part#, Date)
Orders (Name,Part#)

Restructuring of E/R Schema

Restructuring parts:

- Analysis of Redundancies
- Removing Generalizations (Subclasses)
- Partitioning/Merging of Entities and Relations
- Limit the Use of Weak Entity Sets
- Selection of Primary Identifiers (Keys)

Translation into a Logical Schema

Input: E/R Schema

Output: Relational Schema

- Starting from an E-R schema, an equivalent relational schema is constructed
  - "equivalent": a schema capable of representing the same information
- We will deal with the translation problem systematically

Many-to-Many Relationships

Employee(<u>Number</u>, Surname, Salary)

Project(<u>Code</u>, Name, Budget)

Participation(**<u>Number</u>**, **<u>Code</u>**, StartDate)

Red Bold: Foreign Key

Many-to-Many Recursive Relationsh.

Product(<u>Code</u>, Name, Cost)

Composition(**<u>Part</u>**, **<u>SubPart</u>**, Quantity)

Ternary Relationships

Supplier(<u>SupplierID</u>, SupplierName)

Product(<u>Code</u>, Type)

Department(<u>Name</u>, Telephone)

Supply(**<u>SupplierID</u>**, **<u>Code</u>**, **<u>Name</u>**, Quantity)

One-to-Many Relationships

Player(<u>Surname</u>,<u>DateOfBirth</u>, Position)
Team(<u>Name</u>, Town, TeamColours)
Contract(<u>PlayerSurname</u>, <u>PlayerDateOfBirth</u>, Salary, **Name**)

OR (preferred)

Player(<u>Surname</u>,<u>DateOfBirth</u>, Position, Salary, **Name**)
Team(<u>Name</u>, Town, TeamColours)

Weak Entities

University(<u>Name</u>, Town, Address)

Student(<u>RegistrationNumber</u>, **<u>Name</u>**, Surname, EnrolmentYear)

One-to-One Relationships

Head(<u>Number</u>, Name, Salary, StartDate, **DepartmentName**)

Department(<u>Name</u>, Telephone, Branch)

Or

Head(<u>Number</u>, Name, Salary)

Department(<u>Name</u>, Telephone, Branch, StartDate, **HeadNumber**)

Optional One-to-One Relationships

Employee(<u>Number</u>, Name, Salary)
Department(<u>Name</u>, Telephone, Branch, StartDate, **Head**)

Or, if both entities are optional

Employee(<u>Number</u>, Name, Salary)
Department(<u>Name</u>, Telephone, Branch)
Management(<u>**Head**</u>, **Department**, StartDate)

A Sample ER Schema

Entities with Internal Identifiers

E3(<u>A31</u>, A32)

E4(<u>A41</u>, A42)

E5(<u>A51</u>, A52)

E6(<u>A61</u>, <u>A62</u>, A63)

1-N, 1-1 and Opt. 1-1 Relationships

E5(<u>A51</u>, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

Due to R3: A61R3, A62R3, AR3

Due to R4: A61R4, A62R4

Due to R5: A61R5, A62R5, AR5

Weak Entities

E1(<u>A11</u>, **<u>A51</u>**, A12)

E2(<u>A21</u>, **<u>A11</u>**, **<u>A51</u>**, A22)

Many-to-Many Relationships

R2(**A21**, **A11**, **A51**, **A31**, **A41**, AR21, AR22)

Result of the Translation

E1(A11, **A51**, A12)

E2(A21, **A11**, **A51**, A22)

E3(A31, A32)

E4(A41,A42)

E5(A51, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

E6(A61, A62, A63)

R2(**A21**, **A11**, **A51**, **A31**, **A41**, AR21, AR22)

We have a Database Schema!

Ready to create our tables in the DBMS!

Summary of Transformation Rules

Type	Initial schema	Possible translation
Binary many-to-many relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Ternary many-to-many relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$
One-to-many relationship with mandatory participation		$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

...More Rules...

Type	Initial schema	Possible translation
One-to-many relationship with optional participation		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E21}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Relationship with external identifiers		$E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

_ : Primary keys
...: Alternative primary keys
*: NULL values are allowed

…Even More Rules…

Type	Initial schema	Possible translation
One-to-one relationship with mandatory participation for both entities		$E_1(\underline{A_{E11}}, A_{E12}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_2(\underline{A_{E21}}, A_{E22}, \underline{A_{E11}}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$
One-to-one relationship with optional participation for one entity		$E_1(\underline{A_{E11}}, A_{E12}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

_ : Primary keys
…: Alternative primary keys
*: NULL values are allowed

…and the Last One…

Type	Initial schema	Possible translation
One-to-one relationship with optional participation for both entities		$E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A^*_{E11}, A^*_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$

_ : Primary keys
…: Alternative primary keys
*: NULL values are allowed