The Entity/Relationship (E/R) Model & DB Design

csc343, Introduction to Databases
Renée J. Miller and Fatemeh Nargesian and Sina Meraji
Winter 2018
Overview

• Using the Entity/Relationship (ER) Model to model the real world
• From there, designing a database schema
  • Restructuring of an E/R model
  • Translating an E/R model into a logical model (DB Schema)
Conceptualizing the real-world

- DB design begins with a boss or client who wants a database.
- We must map the entities and relationships of the world into the concepts of a database. This is called modeling.
- Sketching the key components is an efficient way to develop a design.
  - Sketch out (and debug) schema designs
  - Express as many constraints as possible
  - Convert to relational DB once the client is happy
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 concepts:
  - *entities*
  - *relationships* among them
  - *attributes* describing the entities and relationships
Entity Sets

- An **entity set** represents a *category* 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 an instance of relationship Residence)
Example of Instances for Exam
Recursive Relationships

- Recursive relationships relate an entity set to itself.
- The relationship may be asymmetric.
  - If so, we indicate the two roles that the entity plays in the relationship.

![Diagram of recursive relationships]

Recursive relationships relate an entity set to itself. The relationship may be asymmetric, and if so, we indicate the two roles that the entity plays in the relationship.
Ternary Relationships

SUPPLIER

SUPPLY

PRODUCT

DEPARTMENT

Department

Supplier

Product

S_1
S_2
S_3
S_4

P_1
P_2
P_3
P_4
P_5

D_1
D_2
D_3
D_4

s_1
s_2
s_3
s_4
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
Keys in E/R

- Notation: solid circle
- If multi-attribute, connect with a line and a “knob”
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

• Notation: pairs of (min, max) values for each

An entity might not participate in any relationship
Cardinalities (cont.)

- In principle, cardinalities are pairs of non-negative integers \((\text{min}, \text{max})\) such that \(\text{min} \leq \text{max}\).

  - **minimum cardinality \(\text{min}\):**
    - If 0, entity participation in a relationship is *optional*.
    - If 1, entity participation in a relationship is *mandatory*.
    - Other values are possible.

  - **maximum cardinality \(\text{max}\):**
    - If 1, each instance of the entity is associated at most with a single instance of the relationship.
    - If \(> 1\), then each instance of the entity can be associated *multiple* instances of the relationship.
    - We write \(N\) to indicate no upper limit.
    - Other values are possible.
Cardinality Examples

ORDER \( (0,1) \) SALE \( (1,1) \) INVOICE

PERSON \( (1,1) \) RESIDENCE \( (0,N) \) CITY

TOURIST \( (1,N) \) RESERVATION \( (0,N) \) VOYAGE
Multiplicity of relationships

If entities E1 and E2 participate in relationship R with cardinalities \((n_1, N_1)\) and \((n_2, N_2)\) then the multiplicity of R is \(N_1\)-to-\(N_2\) (which is the same as saying \(N_2\)-to-\(N_1\))
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)

```
Person
  Surname (0,N)
  License Number (0,1)
  CarRegistration# (0,N)
```
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:

![Diagram showing the revised ER schema]

- Person
  - Surname: (0,1)
  - License Number: (0,1)
  - Owns: (0,N) to Car
  - Owns: (1,1) to CarRegistration#
Keys in E/R

- **Keys** consist of minimal sets of attributes which uniquely identify instances of an entity set.
- Usually, a key is formed by one or more attributes of the entity itself (*internal* keys).
- Sometimes, an entity doesn’t have a key among its attributes. This is called a *weak entity*. Solution: the keys of related entities brought in to help with identification (becoming *foreign keys*).
- A key for a relationship consists of the keys of the entities it relates
Examples of Keys in E/R

- **internal, single-attribute**
  - Registration

- **internal, multi-attribute**
  - Surname
  - FirstName
  - Address

- **foreign, multi-attribute**
  - DateOfBirth

- **Weak entity**
  - Registration
  - Year
  - Surname

- **(1,1)** relationship
  - Student
  - Enrollment

- **(1,N)** relationship
  - Enrollment
  - University

- **Name**
  - University

- **City**
  - University

- **Address**
  - University
General Observations about Keys

- A key may consist of one or more attributes, provided that each of the attributes has (1,1) cardinality.
- A foreign key can involve one or more entities, provided that each of them is member of a relationship to which the entity to be identified participates in the relationship with cardinality equal to (1,1).
- A foreign key may involve an entity that has itself a foreign key, as long as cycles are not generated.
- Each entity set must have at least one (internal or foreign) key.
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?
- **Limitations** of 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
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.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Synonym</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trainee</td>
<td>Participant in a course. Can be an employee or self-employed.</td>
<td>Participant</td>
<td>Course, Company</td>
</tr>
<tr>
<td>Instructor</td>
<td>Course tutor. Can be freelance.</td>
<td>Tutor</td>
<td>Course</td>
</tr>
<tr>
<td>Course</td>
<td>Course offered. Can have various editions.</td>
<td>Seminar</td>
<td>Instructor, Trainee</td>
</tr>
<tr>
<td>Company</td>
<td>Company by which a trainee is employed or has been employed.</td>
<td></td>
<td>Trainee</td>
</tr>
</tbody>
</table>
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
From E/R MODEL TO database schema
Two Steps

- **Restructure** the ER schema to improve it, based on criteria
- **Translate** the schema into the relational model
1. Restructuring an e/r model
Restructuring Overview

**Input:** E/R Schema  
**Output:** Restructured E/R Schema

Restructuring includes:
- Analysis of redundancies
- Choosing entity set vs attribute
- Limiting the use of weak entity sets
- Selection of keys
- Creating entity sets to replace attributes with cardinality greater than one
Example: no redundancy

It is not redundant to have Name twice.
Example: redundancy

What is redundant here?

Part (1,1) Made By (1,N) Manufacturer

Name
Manf Name

Part Number

Name
Address
Example: redundancy

What is redundant here?

Manf Name  Manf Address

Name

Part

Part Number
Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute.
  - It is the “many” in a many-one or many-many relationship.

- Rules of thumb
  - A “thing” in its own right => Entity Set
  - A “detail” about some other “thing” => Attribute

Really this is just about avoiding redundancy
E.S. vs. attributes: examples

Domain fact change: *A part can have more than one manufacturer.*

![Diagram showing the relationship between a part and its manufacturers.](image)
E.S. vs. attributes: examples

Domain fact change: Not representing Manufacturer address ...
E.S. vs. attributes: examples

New domain

Student

Mentored by

Mentor

Name

Student number

Mentor email

Mentor name

Name

Student number
E.S. vs. attributes: examples

Domain fact change: A mentor can have more than one mentee ...

\[ (0,1) \text{ Student} \quad \text{Mentored by} \quad (1,N) \text{ Mentor} \]
When to use weak entity sets?

• The usual reason is that there is no global authority capable of creating unique ID’s

• Example: it is unlikely that there could be an agreement to assign unique student numbers across all students in the world
Don’t Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself
  - They make all entity sets weak, supported by all other entity sets to which they are linked
- It is usually better to create unique IDs
  - Social insurance number, automobile VIN, etc.
  - Useful for many reasons (next slide)
Selecting a Primary Key

• Every relation must have a primary key
• The criteria for this decision are as follows:
  • Attributes with null values cannot form primary keys
  • One/few attributes is preferable to many attributes
  • Internal keys preferable to external ones (weak entities depend for their existence on other entities)
  • A key that is used by many operations to access instances of an entity is preferable to others
Keeping keys simple

Multi-attribute and/or string keys...

- ... are wasteful
  - e.g. Movies(title, year, ...): 2 attributes, ~16 bytes
  - Number of movies ever made $< 2^{32}$ (4 bytes)
    $\Rightarrow$ Integer movieID key saves 75% space and a lot of typing

- ... break encapsulation
  - e.g. Patient(firstName, lastName, phone, ...)
  - Security/privacy hole
    $\Rightarrow$ Integer patientID prevents information leaks

- ... are brittle (nasty interaction of above two points)
  - Name or phone number change? Parent and child with same name?
  - Patient with no phone? Two movies with same title and year?
    $\Rightarrow$ Internal ID always exist, are immutable, unique

Also: computers are really good at integers...
Attributes with cardinality > 1

- The relational model doesn’t allow multi-valued attributes. We must convert these to entity sets.
2. Translating an e/r model into A DB Schema
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

- A good translation should also:
  - not allow redundancy
  - not invite unnecessary null values
The general idea

• Each entity set becomes a relation. Its attributes are
  • the attributes of the entity set.

• Each relationship becomes a relation. It’s attributes are
  • the keys of the entity sets that it connects, plus
  • the attributes of the relationship itself.

• We’ll see opportunities to simplify.
Many-to-Many Binary Relationships
Many-to-Many Binary Relationships

Participation[Code] ⊆ Project[Code]
Many-to-Many Recursive Relationships

![Diagram showing a many-to-many recursive relationship between PRODUCT and COMPOSITION with attributes Quantity, Cost, Name, Code, Part, and Subpart.](image)

53
Many-to-Many Recursive Relationships

Product(\texttt{Code}, \texttt{Name}, \texttt{Cost})

Composition(\texttt{Part}, \texttt{SubPart}, \texttt{Quantity})

Composition[Part] \subseteq Product[Code]

Composition[SubPart] \subseteq Product[Code]
Many-to-Many Ternary Relationships
Many-to-Many Ternary Relationships

Supplier(SupplierID, SupplierName)
Product(Code, Type)
Department(Name, Telephone)
Supply(Supplier, Product, Department, Quantity)

\[
\text{Supply}[\text{Supplier}] \subseteq \text{Supplier}[\text{SupplierID}]
\]
\[
\text{Supply}[\text{Product}] \subseteq \text{Product}[\text{Code}]
\]
\[
\text{Supply}[\text{Department}] \subseteq \text{Department}[\text{Name}]
\]
One-to-Many Relationships
with mandatory participation on the “one” side
One-to-Many Relationships
with mandatory participation on the “one” side

Player(Surname, DOB, Position)
Team(Name, Town, TeamColours)
Contract(PlayerSurname, PlayerDOB, Team, Salary)

Contract[PlayerSurname, PlayerDOB] ⊆ Player[Surname, DOB]
Contract[Team] ⊆ Team[Name]

OR

Player(Surname, DOB, Position, TeamName, Salary)
Team(Name, Town, TeamColors)
Player[TeamName] ⊆ Team[Name]
One-to-One Relationships
with mandatory participation for both

- Head
  - Salary
  - Name
  - Number
  - StartDate

- Management
  - (1,1)

- Department
  - Name
  - Telephone
  - Branch
One-to-One Relationships
with mandatory participation for both

The standard translation, with 3 relations (what is key of mngmt)?

Or

Head(Number, Name, Salary, Department, StartDate)
Department(Name, Telephone, Branch)
    Head[Department] ⊆ Department[Name]

Or

Head(Number, Name, Salary)
Department(Name, Telephone, Branch, HeadNumber, StartDate)
    Department[HeadNumber] ⊆ Head[Number]
One-to-One Relationships
with mandatory participation for both

Or

Head(\textbf{Number}, Name, Salary, StartDate)
Department(\textbf{Name}, Telephone, HeadNumber, Branch)

\text{Department}[\text{HeadNumber}] \subseteq \text{Head}[\text{Number}]
One-to-One Relationships
with optional participation for one
One-to-One Relationships
with optional participation for one

Employee(Number, Name, Salary)
Department(Name, Telephone, Branch)
Management(Head, Department, StartDate)

Management[Head] ⊆ Employee[Number]
Management[Department] ⊆ Department[Name]

Or

Employee(Number, Name, Salary)
Department(Name, Telephone, Branch, Head, StartDate)
Department[Head] ⊆ Employee[Number]
Summary of Types of Relationship

- many-to-many (binary or ternary)
- one-to-many
  - mandatory: (1,1) on the “one” side
  - optional: (0,1) on the “one” side
- one-to-one
  - both mandatory: (1,1) on both sides
  - one mandatory, one optional: (1,1) on one side and (0,1) on other side
  - both optional: (0,1) on both sides
Will the schema be “good”?

• If we use this process, will the schema we get be a good one?
• The process should ensure that there is no redundancy.
• But only with respect to what the E/R diagram represents.
• Crucial thing we are missing: functional dependencies.
  (We only have keys, not other FDs.)
• So we still need FD theory.
Redundancy can be *desirable*

- **Disadvantages** of redundancy:
  - More storage (but usually at negligible cost)
  - Additional operations to keep the data consistent

- **Advantages** of redundancy:
  - Speed: Fewer accesses necessary to obtain information

- **How to decide to maintain or eliminate a redundancy?**

  Examine:
  - the cost of operations that involve the redundant information and
  - the storage needed for the redundant information with and without the redundancy.

*Performance analysis is required to decide about redundancy.*