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

Introduction to databases
CSC343, Fall 2015
Based on slides by Manos Papagelis

Thanks to Diane Horton, Ryan Johnson, John Mylopoulos, Arnold Rosenbloom and Renee Miller for material in these slides
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)
THE ENTITY/RELATIONSHIP (E/R) MODEL
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 concept: entities and their relationships, along with the attributes describing them

Diagram:
- Actors
  - name
  - address
- StarsIn
  - role
- Movies
  - title
  - year
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

A student can’t take more than one exam for a particular course
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

\[ \text{Supplier} \xrightarrow{\text{Supply}} \text{Department} \xrightarrow{\text{Product}} \]

Diagram showing relationships between Supplier, Department, and Product.
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 \(N\) means "any number"

- **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

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 \( E_1 \) and \( E_2 \) 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-\( N_1 \)).

\[
\begin{align*}
\begin{array}{c}
\text{1-to-1} \\
(0,1)
\end{array} & \subset & \begin{array}{c}
\text{N-to-1 OR 1-to-N} \\
(1,1) & \text{or} & (0,N)
\end{array} & \subset & \begin{array}{c}
\text{N-to-N} \\
(1,N) & \text{or} & (0,N)
\end{array}
\end{align*}
\]
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:

![ER Diagram]

- **Person**
  - **Surname**
  - **License Number**

- **Owns**
  - (0,N)

- **Car**
  - **CarRegistration#**
  - (1,1)
Keys in E/R

- **Keys** consist of minimal sets of attributes which uniquely identify 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, 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**
  - AUTOMOBILE
    - Registration
    - Model
    - Colour

- **internal, multi-attribute**
  - PERSON
    - DateOf Birth
    - Surname
    - FirstName
    - Address

- **foreign, multi-attribute**
  - STUDENT
    - Registration
    - Year
    - Surname

- **Weak entity**
  - ENROLMENT
    - (1,1)
  - UNIVERSITY
    - Name
    - City
    - Address
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

- EMPLOYEE
  - Code
  - Surname
  - Salary
  - Age
  - Participation
    - StartDate
  - Project
    - Name
    - Budget
    - ReleaseDate

- MANAGEMENT
  - (0,1)

- MEMBERSHIP
  - (0,1)
  - (1,N)
  - StartDate

- DEPARTMENT
  - (1,N)
  - Phone
  - Name
  - Composition
    - (1,N)

- BRANCH
  - City
  - Address
    - Number
    - Street
    - PostCode
Challenge: modelling 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
... the E/R model result
FROM E/R MODEL TO DATABASE SCHEMA
(Relational) Database Design

• Given a conceptual schema (ER, but could also be UML), generate a logical (relational) schema

• It is helpful to divide the design into two steps:
  - *Restructuring of the ER schema*, based on criteria for the optimization of the schema
  - *Translation into the logical model*, based on the features of the logical model (in our case, 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

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

- **Part Number**
- **Name**
- **Address**
Example: redundancy

![Diagram of entity relationship model showing Part, Made By, and Manufacturer entities with their attributes and relationships:]

- **Part** with attributes: Name, Part Number, and Manf Name
- **Made By** relationship with cardinality (1,1) to (1,N) between Part and Manufacturer
- **Manufacturer** with attributes: Address, Name, and Manf Name

This diagram illustrates the concept of redundancy in databases, where multiple parts can be made by the same manufacturer, and each part can be associated with a single manufacturer.
Example: redundancy

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 nonkey 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
  - A “detail” correlated among many “things” => Entity Set

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

Domain fact change: A part can have more than one manufacturer ...
E.S. vs. attributes: examples

Name

Part

Part Number

(1,N)

Made By

(1,N)

Manufacturer

Name

Manf Name

Name

(1,N)

Part

Part Number

(No address attribute)
E.S. vs. attributes: examples

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 ...

![Entity-Relationship Diagram]

- **Student**
  - Name
  - Student number

- **Mentor**
  - Name
  - Mentor name
  - Mentor email

- **Mentored by**
  - (0,1) from Student to Mentored by
  - (1,N) from Mentored by to 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 redundant
  - e.g. Movies(title, year, ...): 2 attributes, ~16 bytes
  - Number of movies ever made << $2^{32}$ (4 bytes)
  => Integer movieID key saves 75% space and a lot of typing

• ... break encapsulation
  - e.g. Patient(firstName, lastName, phone, ...)
  - Security/privacy hole
  => 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?
  => Internal ID always exists, 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.
Many-to-Many Binary Relationships

Employee\((\text{Number}, \text{Surname}, \text{Salary})\)
Project\((\text{Code}, \text{Name}, \text{Budget})\)
Participation\((\text{Number}, \text{Code}, \text{StartDate})\)
Many-to-Many Recursive Relationships

Product(Code, Name, Cost)
Composition(Part, SubPart, Quantity)
Ternary Many-to-Many Relationships

Supplier(SupplierID, SupplierName)
Product(Code, Type)
Department(Name, Telephone)
Supply(Supplier, Product, Department, Quantity)
One-to-Many Relationships
with mandatory participation for one

Player(Surname, DateOfBirth, Position)
Team(Name, Town, TeamColours)
Contract(PlayerSurname, PlayerDateOfBirth, TeamName, Salary)

OR

Player(Surname, DateOfBirth, Position, TeamName, Salary)
Team(Name, Town, TeamColours)
One-to-One Relationships with mandatory participation for both

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

Or
Head(Number, Name, Salary, StartDate)
Department(Name, Telephone, HeadNumber, Branch)
One-to-One Relationships with optional participation for one

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

Or, if both entities are optional
Employee(Number, Name, Salary)
Department(Name, Telephone, Branch)
Management(Head, Department, StartDate)
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
# Summary of Transformation Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
</table>
| **Binary many-to-many relationship**      | ![Diagram 1](image1.png) | $E_1(A_{E_{11}}, A_{E_{12}})$  
                                        | $E_2(A_{E_{21}}, A_{E_{22}})$  
                                        | $R(A_{E_{11}}, A_{E_{21}}, A_R)$ |
| **Ternary many-to-many relationship**     | ![Diagram 2](image2.png) | $E_1(A_{E_{11}}, A_{E_{12}})$  
                                        | $E_2(A_{E_{21}}, A_{E_{22}})$  
                                        | $E_3(A_{E_{31}}, A_{E_{32}})$  
                                        | $R(A_{E_{11}}, A_{E_{21}}, A_{E_{31}}, A_R)$ |
| **One-to-many relationship with mandatory participation** | ![Diagram 3](image3.png) | $E_1(A_{E_{11}}, A_{E_{12}}, A_{E_{21}}, A_R)$  
                                        | $E_2(A_{E_{21}}, A_{E_{22}})$ |
...More Rules...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-many relationship with optional participation</td>
<td><img src="image" alt="Diagram" /></td>
<td>$E_1(A_{E11}, A_{E12})$&lt;br&gt;$E_2(A_{E21}, A_{E22})$&lt;br&gt;$R(A_{E11}, A_{E21}, A_R)$&lt;br&gt;Alternatively:&lt;br&gt;$E_1(A_{E11}^<em>, A_{E12}, A_{E21}^</em>, A_R^*)$&lt;br&gt;$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td>Relationship with external identifiers</td>
<td><img src="image" alt="Diagram" /></td>
<td>$E_1(A_{E12}, A_{E21}, A_{E11}, A_R)$&lt;br&gt;$E_2(A_{E21}, A_{E22})$</td>
</tr>
</tbody>
</table>
### Even More Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
</table>
| One-to-one relationship with mandatory participation for both entities | ![Diagram](image)                                                              | $E_1(A_{E11}, A_{E12}, A_{E21}, A_{E22})$  
$E_2(A_{E21}, A_{E22})$  
Alternatively:  
$E_2(A_{E21}, A_{E22}, A_{E11}, A_{E12})$  
$E_1(A_{E11}, A_{E12})$ |
| One-to-one relationship with optional participation for one entity    | ![Diagram](image)                                                              | $E_1(A_{E11}, A_{E12}, A_{E21}, A_{E22})$  
$E_2(A_{E21}, A_{E22})$  
$E_1(A_{E11}, A_{E12})$ |
...and the Last One...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
</table>
| One-to-one relationship with optional participation for both entities | ![Diagram](image) | $E_1(A_{E11}, A_{E21})$
$E_2(A_{E21}, A_{E22}, A_{E11}^*, A_{R}^*)$
Alternatively:
$E_1(A_{E11}, A_{E12}, A_{E21}^*, A_{R}^*)$
$E_2(A_{E21}, A_{E22})$
Alternatively:
$E_1(A_{E11}, A_{E12})$
$E_2(A_{E21}, A_{E22})$
$R(A_{E11}, A_{E21}, A_{R})$ |
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*

- We’ve talked a lot about avoiding redundancy.  
- It’s a *disadvantage*: because of larger storage requirements, (but, usually at negligible cost) and the necessity to carry out additional operations in order to keep the derived data consistent
- But it’s also an *advantage*: a reduction in the number of accesses necessary to obtain derived information
- The decision to maintain or eliminate a redundancy is made by comparing the *cost of operations* that involve the redundant information and the *storage needed*, in the case of presence or absence of redundancy.

*Performance analysis is required to decide about redundancy*
Exercise

1. A car sale cannot involve more than one salesperson. **True**
2. There can be two cars with the same VIN as long as the model and year are different. **True**
3. A salesperson can work at any number of dealerships. **True**
4. There can't be more salespeople than dealerships. **False**
5. There can be multiple sales on the same date. **True**
6. Two salespeople can have the same sID as long as they work at different dealerships. **True**
7. This model contains a weak entity set. **Yes**
8. The works at relationship is a one-to-many relationship. **True**

**Entity-Relationship Diagram**

- car(VIN, model, year)
- customer(name, email, phone)
- salesperson(sID, sName, dID, salary)
- dealership(dID, dName)
- sale(name, email, sID, VIN, date, amount)
- phone(name, email, phone)
Exercise

1. A car sale cannot involve more than one salesperson.
   - True False
2. There can be two cars with the same VIN as long as the model and year are different.
   - True False
3. A salesperson can work at any number of dealerships.
   - True False
4. There can't be more salespeople than dealerships.
   - True False
5. There can be multiple sales on the same date.
   - True False
6. Two salespeople can have the same sID as long as they work at different dealerships.
   - True False
7. This model contains a weak entity set.
   - True False
8. The works at relationship is a one-to-many relationship.
   - True False