Exercise 2: Relational Model

Solution

The exercises marked with * will be discussed in the exercise session. You can solve the other exercises as practice, ask questions about them in the session, and hand them in for feedback. All exercises may be relevant for the exam.

Ask Zsolt (zsolt.istvan@inf.ethz.ch) for feedback on this week’s exercise sheet or give it to the TA of your session (preferably stapled and with your e-mail address).

1 Keys in the Relational Model*

1. Based on what you learned about the relational model and schemas, mark all statements below that are true.

- Every relation in a relational schema has at least one key.
  - The primary key of a relation is composed of a single attribute.
  - Secondary keys of the relation have to be composed of at least two attributes.
  - In special cases there can be duplicate values among the keys of a relation.

2. Consider the following relational schema of two relations. We underlined the attributes that compose the primary key of each of them:

   Student(FirstName, LastName, Age, LegiNr)
   Attends(LegiNr, LectureName, ...)

   Can you think of a real-world scenario that cannot be represented in this relational schema? Give an example:
**Solution:** In this schema it is not possible to store two students with the same first and last name. While uncommon, it might happen that two people with the same name attend the same university. Therefore choosing the LegiNr as a key might be a better idea.

## 2 Converting relationships*

Consider the following ER model with entities $A$ and $B$ (with attributes $a$ and $b$) connected through a relationship.

![ER model](image)

1. Complete the table below by converting the ER model to relational schema, for all cardinality options. Write down the relations and underline their primary keys. In case a relation has multiple keys (see the lecture slides for examples) underline one, and list the others as “additional keys”.

<table>
<thead>
<tr>
<th>ER model $(X:Y)$</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>M:N</td>
<td>$A(a) \ B(b)\ rel(a,b)$</td>
</tr>
<tr>
<td>1:N</td>
<td>$A(a)\ B(b,a)$</td>
</tr>
<tr>
<td>N:1</td>
<td>$A(a,b)\ B(b)$</td>
</tr>
<tr>
<td>1:1</td>
<td>$rel(a,b)$ additional key: b</td>
</tr>
</tbody>
</table>

2. Suppose we want to add elements to the relations. Mark which tuples from below can be inserted into the relational schemas you created for the M:N relationship:

   - ✔ (a1, b1)
   - ✔ (a1, b2)
   - ✔ (a2, b1)
   - ✔ (a2, b2)
   - ☐ (a1, b1)

3. How about the 1:N case?

   - ✔ (a1, b1)
   - ✔ (a1, b2)
   - ☐ (a2, b1)
   - ☐ (a2, b2)
   - ☐ (a1, b1)
4. Which ones can be inserted if the relationship is 1:1?

- (a1, b1)
- (a1, b2)
- (a2, b1)
- (a2, b2)
- (a1, b1)

3 Finish the Conversion*

1. Which one of the following problem descriptions could have lead to the ER model above? One or more can be possibly correct.

- We are modeling a volleyball tournament. A volleyball game is played by two teams. Multiple referees make sure during a single game that the rules are respected.
- A volleyball game in a tournament is played by two teams. A referee has to be present at each game to make sure the rules are respected.
- Two teams play volleyball in a tournament. In each game, one has the role of “home” team and the other one has the role of “away” team. Each play is supervised by at most one referee.

**Solution:** The first option does not correspond to the ER model: in the ER model each play (combination of home and away team) has to be associated with a single referee. The description however requires multiple referees.

The second option matches the ER model.

The third option is not correct because the model requires exactly one referee. To express an optional referee could be done by adding it as attribute to the relation. In this case it could have a ’null’ value. You will learn about nulls later in the lecture.

2. Based on the ER model above, we came up with the following schema, which is not quite correct:

```sql
Team (Teamname)
Referee (RefereeName)
Plays (RefereeName, Home, Away)
```
Identify the mistake we made and construct an example that is correct in the provided relational schema but violates the ER model and textual description:

**Solution:** The problem in the schema is that we added RefereeName as part of the key for Plays, together with the home team. This means that for each home game of a team there has to be a different referee. An other issue with the schema is that it would allow two teams to play in the same home/away setup any number of times given that they have a different referee.

The correct relation should be: **Plays (RefereeName, Home, Away)**

4 Weak Entities*

The following ER model describes the fact that cities are located in countries. A street is located in a city. A house in a street. And an apartment is located in a house.

1. We started the conversion to a relational model, and got so far:
   - **Country (CoName)**
   - **City (CoName, CiName)**
   - **Street (CoName, CiName, SName)**

   Add the other necessary relations to the schema:

   **Solution:** **House (CoName, CiName, SName, HNumber)**
   **Apartement (CoName, CiName, SName, HNumber, ANumber)**

2. Would it be possible to use less attributes as keys in any of the relations? If yes, which keys could be simplified? If no, give a reason why?

   **Solution:** The keys cannot be reduced because they capture the weak entity relationship in the model. Furthermore, if for instance a city had only its name as key, all city names would have to be unique in the world (in this model they are unique in their countries). The argument applies to all other entities as well.
5 Employees Schema*

During the semester you will work with several databases, one from which represents the database that a human resources department would use at a company. Below you'll find the relational schema. 

- **departments** (dept_no, dept_name)
- **dept_employee** (emp_no, dept_no, from_date, to_date)
- **dept_manager** (emp_no, dept_no, from_date, to_date)
- **employees** (emp_no, birth_date, first_name, last_name, gender, hire_date)
- **salaries** (emp_no, salary, from_date, to_date)

Given this relational schema mark from the following statements the ones that are true (without considering additional constraints).

- √ Two departments with the name "Engineering" could exist at the same time.
- ⃝ No employee can work at a department and manage another one at the same time.
- √ An employee could be employed at two departments at the same time.
- ⃝ For the entries in the dept_manager relation the to_date cannot be before the from_date.
- ⃝ Given an employee we can identify exactly one department where (s)he works.
- √ An employee could collect more than one salary at the same time.
- ⃝ Some entries in salaries could have no from_date and no employee associated to them.

Solution:

The main take-away of this exercise is that without additional constraints defined on the values permitted for each attribute, the relational model can only express a limited set of rules of what is permitted in the database. This means that some schemas are up for interpretation if no additional information is provided (though the ambiguity is much less than in the case of converting text to ER models). Later in the course you will learn about constraints that can be defined on the attributes of relations.

6 Schema Training

Convert the following ER diagrams into relational schemas. Define a primary key for each relation. Invent key attributes for the ER diagrams if they don’t have any.
6.1 Inheritance*

Solution:
1st iteration (one relation per entity or relationship):
- Man (MName)
- Women (WName)
- IsSon (Son, Father, Mother); additional key: Son, Mother
- IsDaughter (Daughter, Mother, Father); additional key: Daughter, Father

6.2 Library System*

Solution:
1st iteration (one relation per entity or relationship):
- Reader (ReaderNr, FamilyName, Firstname, City, Birthdate)
- Copy (CopyNr, Title, Author, NumPages, Category)
- Book (Available, ISBN, PubYear, Title, Author, NumPages)
- Publisher (PubName, PubCity)
- Category (Catname, Contains)
- InCat (InCat, Category, Book)
- Borrow (Borrow, Reader, Copy, ReturnDate, Shelf, Position)
- Publishes (Publishes, Publisher, Book)
**Solution:** Translating the entities leads to the following relations:

- Reader (ReaderNr, FamilyName, Firstname, City, Birthdate) (1)
- Book (ISBN, Title, Author, NumPages, PubYear) (2)
- Publisher (Pubname, Pubcity) (3)
- Category (Catname) (4)
- Copy (ISBN, CopyNr, Shelf, Position) (5)

For the relationships we create these relations:

- Borrows (ReaderNr, ISBN, CopyNr, ReturnDate) (6)
- Available (ISBN, CopyNr) (7)
- Contains (Catname, ContainedIn) (8)
- InCat (ISBN, Catname) (9)
- Publishes (ISBN, Pubname) (10)

Finally, we will combine relations for binary relationships, if they have the same keys and they are of type 1:N, N:1, or 1:1. ( (7) → (5), (8) → (4), (10) → (2) ):

- Reader (ReaderNr, FamilyName, Firstname, City, Birthdate)
- Book (ISBN, Title, Author, NumPages, PubYear, Pubname)
- Publisher (Pubname, Pubcity)
- Category (Catname, ContainedIn)
- Copy (ISBN, CopyNr, Shelf, Position)
- Borrows (ReaderNr, ISBN, CopyNr, ReturnDate)
- InCat (ISBN, Catname)

Note: Weak relationships are often not explicitly shown as relations, because they disappear anyway when combined as shown above (e.g. relation (7) in this example). Take care, however, not to forget attributes when using such shortcuts.
6.3 Trains

Solution: Initially you get the following relations for entities:

- City (Name, Canton) (1)
- Trainstation (Name, NumTracks) (2)
- Train (TrainNr, Length) (3)

For the relationships we get:

- LocatedIn (TrainstationName, CityName, Canton) (4)
- Start (TrainNr, StartTrainstationName) (5)
- Destination (TrainNr, DestTrainstationName) (6)
- Connects (FromTrainstation, ToTrainstation, TrainNr, Departure, Arrival) or
  Connects (FromTrainstation, ToTrainstation, TrainNr, Departure, Arrival) (7)

Note: The or here means that we have the choice about what make the primary key of the relation. In order to preserve the 1:1:N functionalities of the Connects relationship, we need to have both as (secondary) keys though. However, this has been covered only briefly in the lecture and will come to it when we talk about normal forms.

Next we will combine relations for binary relationships, if they have the same keys and they are of type 1:N, N:1, or 1:1. ( (4) → (2), (5) → (3), (6) → (3)):

- City (Name, Canton)
- Trainstation (Name, NumTracks, CityName, Canton)
- Train (TrainNr, Length, StartTrainstationName, DestTrainstationName)
- Connects (FromTrainstation, ToTrainstation, TrainNr, Departure, Arrival) or
  Connects (FromTrainstation, ToTrainstation, TrainNr, Departure, Arrival)
6.4 Hospital

Solution:
Initially we get the following relations for entities:

- Worker (PersNr, Name) (1)
- Station (StationNr, Name) (2)
- Doctor (PersNr, Expertise, Degree) (3)
- Nurse (PersNr, Skills) (4)
- Patient (PatientNr, Name, Illness) (5)
- Room (StationNr, RoomNr, NumBeds) (6)

For the relationships we get:

- Works (StationNr, PersNr) (7)
- Treats (PatientNr, PersNr) (8)
- LocatedAt (StationNr, RoomNr, PatientNr, From, To) (9)
- At (StationNr, RoomNr) (10)

And finally we combine relations with the same key: (7) and (1/3/4), (5) and (9), (6) and (10):

- Worker (PersNr, Name, StationNr)
- Station (StationNr, Name)
- Doctor (PersNr, Name, StationNr, Expertise, Degree)
- Nurse (PersNr, Name, StationNr, Skills)
- Patient (PatientNr, Name, Illness, RoomNr, From, To)
- Room (StationNr, RoomNr, NumBeds)
- Treats (PatientNr, PersNr)

7 Generic ER Model Translation

Consider the ER models 1 to 4 and the relational models (a) to (j) given below. For each ER model, give a relational model that represents it correctly by adding a checkmark (✓) in the
corresponding field in the table. Only add one checkmark per ER model. If an ER model is represented by none of the given relational models, then add a checkmark in the “none” column.

<table>
<thead>
<tr>
<th>ER Model</th>
<th>none</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
<th>(i)</th>
<th>(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER model 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ER model 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ER model 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ER model 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


(a) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,d,r)$  

(b) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,c',r)$  

(c) $B(b)$  
$C(c')$  
$D(c,d,s)$  
$R(b,c,c',d,r)$, extra key: $b,d,c'$  

(d) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,d,r)$, extra key: $b,d$  

(e) $B(b)$  
$D(c,d,s)$  
$R(b,c,d,r)$, extra key: $b,d$  

(f) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,c',d,r)$  

(g) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,d,r)$, extra key: $b,d$  

(h) $B(b)$  
$D(c,d)$  
$R(b,c,d,r)$, extra key: $b,d$  

(i) $B(b)$  
$C(c)$  
$D(c,d,s)$  
$R(b,c,d,r)$  

(j) $B(b)$  
$C(c,d,s)$  
$D(d)$  
$R(b,c,d,r)$, extra key: $b,d$
Solution:

<table>
<thead>
<tr>
<th>ER model</th>
<th>Relational Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER model 1</td>
<td>(d)</td>
</tr>
<tr>
<td>ER model 2</td>
<td>(a)</td>
</tr>
<tr>
<td>ER model 3</td>
<td>none</td>
</tr>
<tr>
<td>ER model 4</td>
<td>(b)</td>
</tr>
</tbody>
</table>

Explanation:

**ER model 1**

We first map each entity to a relation, making the key attribute of the entity the key of the relation:

- \(B(b)\), \(C(c)\), \(D(d)\)

We then map each relationship to a relation, including the attributes of the relationship itself and the keys of the participating entities:

- \(R(b,c,d,r)\), \(S(c,d,s)\)

We now need to ensure each entity with a “1” functionality in a relationship with a key in the relation of the relationship that consists of the keys of all other entities. For \(S\), we ensure the “1” functionality of \(C\) by making \(d\), the key of the other entity, a key of \(S\):

- \(S(c,d,s)\)

For \(R\), we ensure the “1” functionality of \(C\) by making \(b,d\), the keys of the other entities, a key of \(R\), and the “1” functionality of \(B\) by making \(c,d\), the keys of the other entities, another key of \(R\). \(R\) has thus two keys:

- \(R(b,c,d,r)\), extra key: \(b,d\)

Finally, we merge all relations of 1:N and 1:1 relationships with the same key, i.e., we merge \(S\) and \(D\), which both have the key \(d\). With this, we get solution \((d)\):

- \(B(b)\)
- \(C(c)\)
- \(D(c,d,s)\)
- \(R(b,c,d,r)\), extra key: \(b,d\)

**ER model 2**

We perform the same basic steps as with the first ER model. We first create a relation for each entity and one for each relationship.

- \(B(b)\), \(C(c)\), \(D(d)\), \(R(b,c,d,r)\), \(S(c,d,s)\)

The keys of the relations for the entities are straightforward. The “1” functionality of \(C\) in \(S\) makes \(d\) a key of \(S\); the “1” functionality of \(B\) in \(R\) makes \(c,d\) a key of \(R\).

Finally, we merge \(S\) and \(D\), which both have the key \(d\) and come from a 1:N relationship. With this, we get solution \((a)\).
ER model 3
We perform the same basic steps as with the first ER model. We first create a relation for each entity:

\[ \text{B}(b), \text{C}(c), \text{D}(c,d) \]

The weak entity \( D \) has as additional attribute the key of its strong entity \( C \), which becomes part of its key. The keys of the relations for the other entities are straightforward.

Apart from its own attributes, the weak relationship \( S \) has only the key attributes of the weak entity, \( D \) (the instance of \( C \) connected by \( S \) to an instance of \( D \) is the same as the one that the instance of \( D \) inherits the key from). Furthermore, the “1” functionality of \( C \) in \( S \) makes \( d \) a key of \( S \):

\[ S(c,d,s) \]

The relation \( R \) has the key attributes of all participating relations plus its own attributes. We resolve the name clash of \( c \) by renaming it occurrence in \( D \) to \( c' \). The “1” functionality of \( B \) in \( R \) makes the keys of the other entities a key of \( R \). We thus obtain:

\[ R(b,c,c',d,r) \]

Finally, we merge \( S \) and \( D \), which both have the key \( c,d \) and come from a 1:N relationship.

With this, we get the following relations:

\[ \text{B}(b), \text{C}(c), \text{D}(c,d,s) \]

This does not correspond to any of the proposed solutions. In particular is it different from \( [\text{c}] \), which has an additional key.

ER model 4
We perform the same basic steps as with the first ER model. We first create a relation for each non-weak entity:

\[ \text{B}(b), \text{C}(c) \]

We can model the specialization \( D \) of \( C \) either by storing all information of \( D \) in one relation (including the attributes of \( C \)) or only its extra attributes. In both cases, the key of the general relation becomes also the key of the special relation, i.e., the key of \( D \) is \( c \). In ER model 4 both possibilities give the same relations because the general entity, \( C \), has no non-key attributes that could or could not be duplicated in the relation of the special entity. In all cases, we thus get:

\[ \text{D}(c,d,s) \]

The relationship \( R \) consists of its attributes and the keys of the participating relations. We resolve the name clash of \( C \) and \( D \) by renaming \( c \) of \( D \) to \( c' \) (the instance of \( D \), which is an instance of \( C \), may participate in \( R \) with a different instance of \( C \) than itself). \( R \) has thus the form:

\[ R(b,c,c',d,r) \]

The “1” functionality of \( B \) in \( R \) makes the keys of the other entities, namely \( c \) and \( c' \), a key of \( R \).

Since we cannot merge any relations, we obtain solution \([b]\).