TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

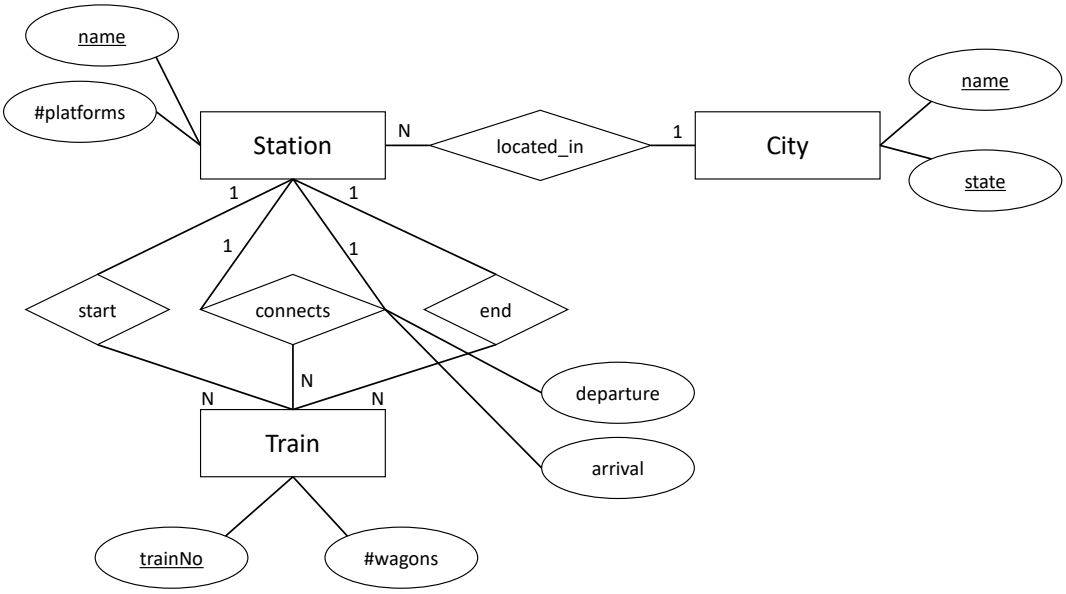
***Database System Concepts for Non-Computer Scientist* - WiSe 24/25**

Alice Rey (rey@in.tum.de)
http://db.in.tum.de/teaching/ws2425/DBSandere/?lang=en

**Sheet 03**

## Exercise 1

Consider the entity relationship diagram from exercise sheet 2:




Refine the relational schema that you created in sheet 2 from the ER-Diagram. Underline keys and find appropriate data types. As a reminder, here is the un-refined schema:

$$\text{City} : \{[\underline{\text{name} : \text{string}, \text{state} : \text{string}}]\} \quad (1)$$
$$\text{Station} : \{[\underline{\text{name} : \text{string}}, \#\text{platforms} : \text{integer}]\} \quad (2)$$
$$\text{Train} : \{[\underline{\text{trainNo} : \text{integer}}, \#\text{wagons} : \text{integer}]\} \quad (3)$$

For the relationships in the model, we create the following relations:

$$\text{located\_in} : \{[\underline{\text{stationName} : \text{string}}, \text{cityName} : \text{string}, \text{cityState} : \text{string}]\} \quad (4)$$
$$\text{start} : \{[\underline{\text{trainNo} : \text{integer}}, \text{stationName} : \text{string}]\} \quad (5)$$
$$\text{end} : \{[\underline{\text{trainNo} : \text{integer}}, \text{stationName} : \text{string}]\} \quad (6)$$
$$\text{connects} : \{[\underline{\text{fromStationName} : \text{string}}, \text{toStationName} : \text{string}, \underline{\text{trainNo} : \text{integer}}, \text{departure} : \text{date}, \text{arrival} : \text{date}]\} \quad (7)$$

**Solution:**

During refinement, we merge relations for binary relationships into relations for entities, if the relations have the same key and it was a 1:N, N:1 or 1:1 relationship in the ER-model. Note: A binary 1:N relationship can be merged into the entity with the $N$ next to it.

Doing so we can merge the (4) relation into (2). (5) gets merged into (3). And same for the *end* relation, which also gets merged into *train*.

$$(4) \mapsto (2),\ (5) \mapsto (3),\ (6) \mapsto (3)$$

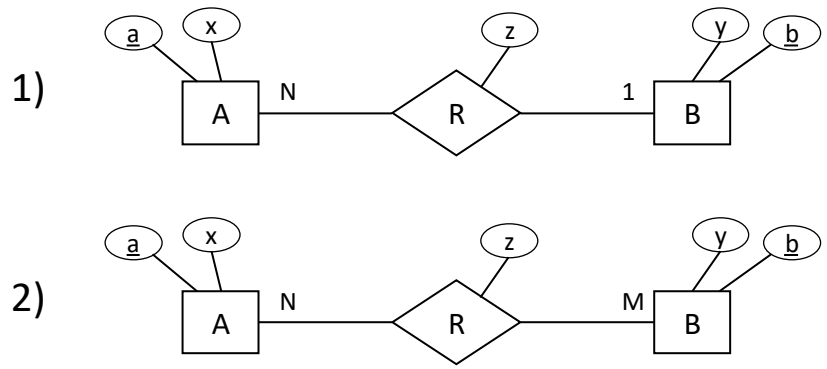
Thus, we end up with the following schema:

$$\begin{aligned}
\text{City} &: \{[\underline{\text{name} : \text{string}}, \text{state} : \text{string}]\} \\
\text{Station} &: \{[\underline{\text{name} : \text{string}}, \#\text{platforms} : \text{integer}, \\
&\quad \text{cityName} : \text{string}, \text{state} : \text{string}]\} \\
\text{Train} &: \{[\underline{\text{trainNo} : \text{integer}}, \#\text{wagons} : \text{integer}, \\
&\quad \text{startStationName} : \text{string}, \text{endStationName} : \text{string}]\} \\
\text{connects} &: \{[\underline{\text{fromStationName} : \text{string}}, \text{toStationName} : \text{string}, \\
&\quad \underline{\text{trainNo} : \text{integer}}, \text{departure} : \text{date}, \text{arrival} : \text{date}]\}
\end{aligned}$$

In our model the train number is uniquely identifying a connection between two cities (possibly involving serveral stations). An ICE starting in Munich (*startStationName*) and going to Berlin (*endStationName*) has a unique train number. When the train returns it has a different train number. Therefore, in the *connects* relation, the (*trainNo*, *fromStationName*)-pair and the (*trainNo*, *toStationName*)-pair are both valid keys (as they are both uniquely identifying a tuple in the relation).

## Exercise 2

Consider the following ER-diagram:




Refine and transform this diagram into a database schema (SQL DDL). You can assume that each attribute is an integer. Use **not null**, **primary key**, **references**, **unique** and **cascade** when possible/necessary.

**Solution:**

**1)**

```
create table A (
    a int not null primary key,
    x int
);

create table B (
    b int not null primary key,
    y int
);

create table R (
    a int not null references A primary key,
    b int not null references B,
    z int
);
```

Alternatively, we can merge the $R$ relation into the $A$ relation. Remember, a relationship can be merged into the entity with the same key or (graphically) the one on the "$N$"-side.

```
create table A (
    a int not null primary key,
    x int,
    b int references B,
    z int
);
```

The downside of this is that we now have information about the relation $R$ inside of the $A$ relation (namely, the $z$ and $b$ attribute). If $R$ is a sparse relationship (not many tuples in $A$ are connected to $B$), then we end up with a lot of null values.

Also take note of how the foreign keys in $R$ (that is, $R.a$ and $R.b$) are marked as *not null*. Once $R$ is merged into $A$, the reference to $B$ (that is, $A.b$) becomes *nullable*. The reason for this is that the entity relationship diagram specifies that each tuple in $A$ has zero or one partners in $B$. We need the *nullable* for $A.b$, otherwise zero partners would not work. This is not necessary if we translate $R$ as its own relation because in this case the "zero partner"-case is expressed by simply not having a tuple in $R$.

**2)**

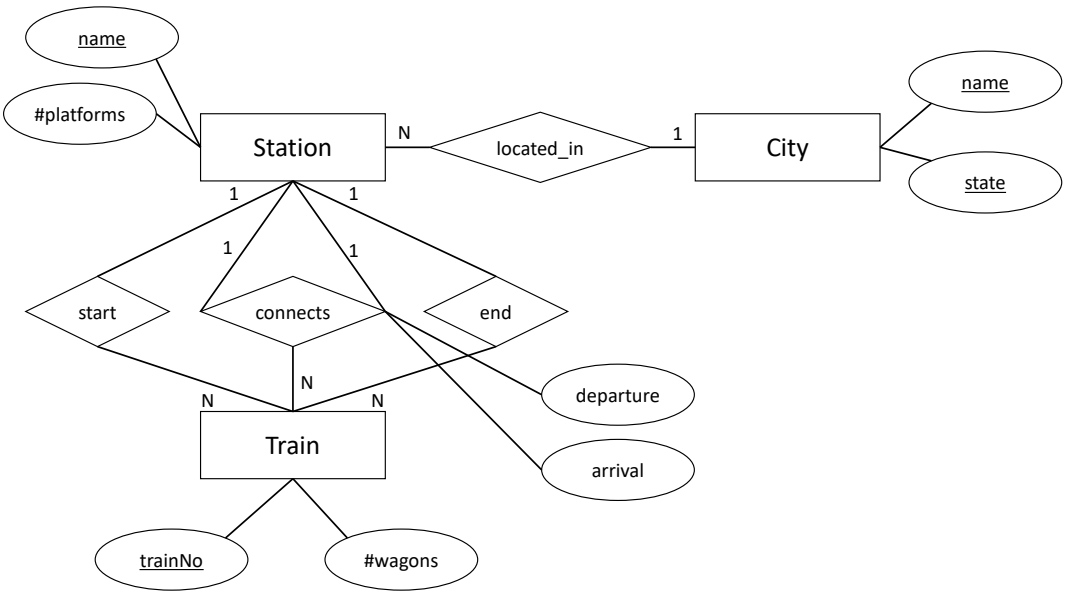
```
create table A (
    a int not null primary key,
    x int
);

create table B (
    b int not null primary key,
    y int
);

create table R (
    a int not null references A,
    b int not null references B,
    z int,
    primary key(a, b)
```

```
);
```

Here we can not merge $R$ into any of the relations. Remember, N-to-M relationships can not be refined. Note that we have to use $R.a$ and $R.b$ together as the primary key because neither attribute is unique on its own. In addition, we actually have to use this primary key for a correct translation of the entity relationship model. Otherwise, we have the following problem: Assume a relation $A$ with values (1, 1) and $B$ with (2, 2). Without a primary key on $a$ and $b$ in $R$, we could have the entries (a=1, b=2, z1) and (a=1, b=2, z2). This would mean that one entry in $A$ can map to the same entry in $B$ multiple times, which is not allowed in entity relationship models. Therefore, we have to exclude this and use the primary key constraint in $R$.

## Homework 3

Look at the following (familiar) ER-diagram and create SQL DDL statements to create the respective tables.




**Lösung:**

```
create table city (name varchar(50) not null,
                   state varchar(50) not null,
                   primary key(name, state)
);

create table station (name varchar not null primary key,
                      num_platforms int,
                      cityName varchar(50) not null,
                      state varchar(50) not null,
                      foreign key(cityName, state)
                             references city(name, state)
);
```

```
create table train (trainNo int not null primary key,
                    num_wagons int,
                    start varchar not null references station,
                    end varchar not null references station
);

create table connects (from varchar not null references station,
                       to varchar not null references station,
                       trainNo int not null references train,
                       departure date,
                       arrival date,
                       primary key(from, trainNo)
);
```