CCT396, Fall 2011

Database Design and Implementation

Yuri Takhteyev
University of Toronto

Week 4

Converting an ER Design into a Relational Form

Common Patterns

Hierarchical (1:M)
vs.
Not (Quite) Hierarchical (M:M)

A contains B

1:M

building ≺ room

cd ≺ track, book ≺ chapter

car ≺ part

province ≺ riding

neighborhood ≺ restaurant

session ≺ prediction

invoice ≺ billable item

A contains B

M:M

course $\succ\!\!-\!\!\prec$ student

list $\prec$ restaurant

dish $\succ\!\!-\!\!\prec$ ingredient

A “owns” B

1:M

mother ≺ child

user ≺ comment

restaurant ≺ rating

comment ≺ rating

comment ≺ reply

customer ≺ payment

customer ≺ invoice

customer ≺ session

A “owns” B

M:M

investor ≻—≺ company

instructor ≻—≺ course

Belonging to Different Entities

1:M

user $\prec$ comment $\succ$ restaurant

customer $\prec$ session $\succ$ f. teller

B “instantiates” A

1:M

species ≺ pet

model ≺ vehicle

book ≺ edition

course ≺ course_instance

(“CCT395” vs “CCT395 in Fall 2011”)

M:M

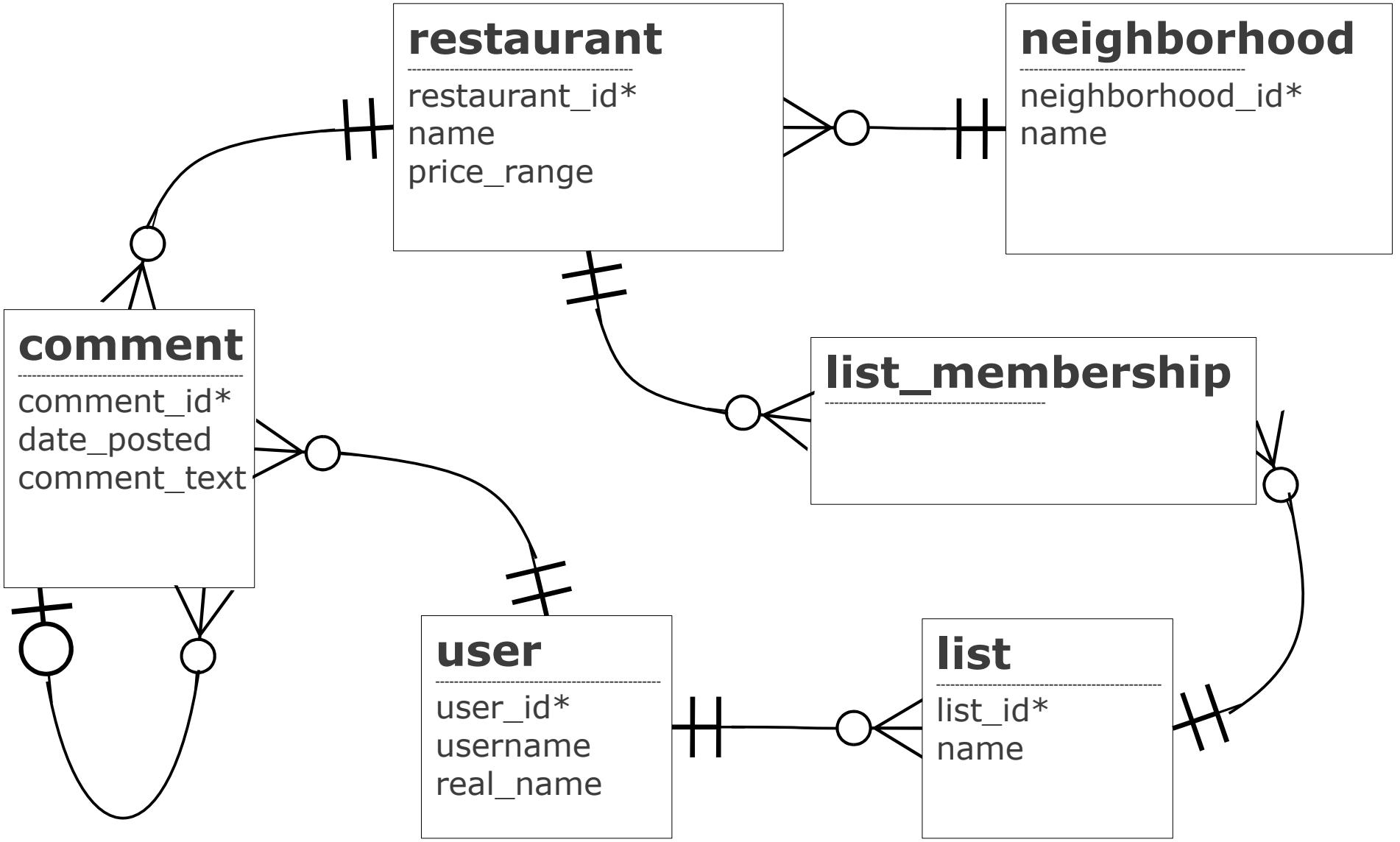
employee ≻≺ role

Drawing Software

Options for software:

- OpenOffice Draw
  - Free / open source
  - Available in the lab
  - You can get “Crow’s Foot” templates at http://www.thinktek.ca/articles/article2.php
  - Alternatively, do UML notation (“n..m”) by hand
- Microsoft Visio
- Your favorite software

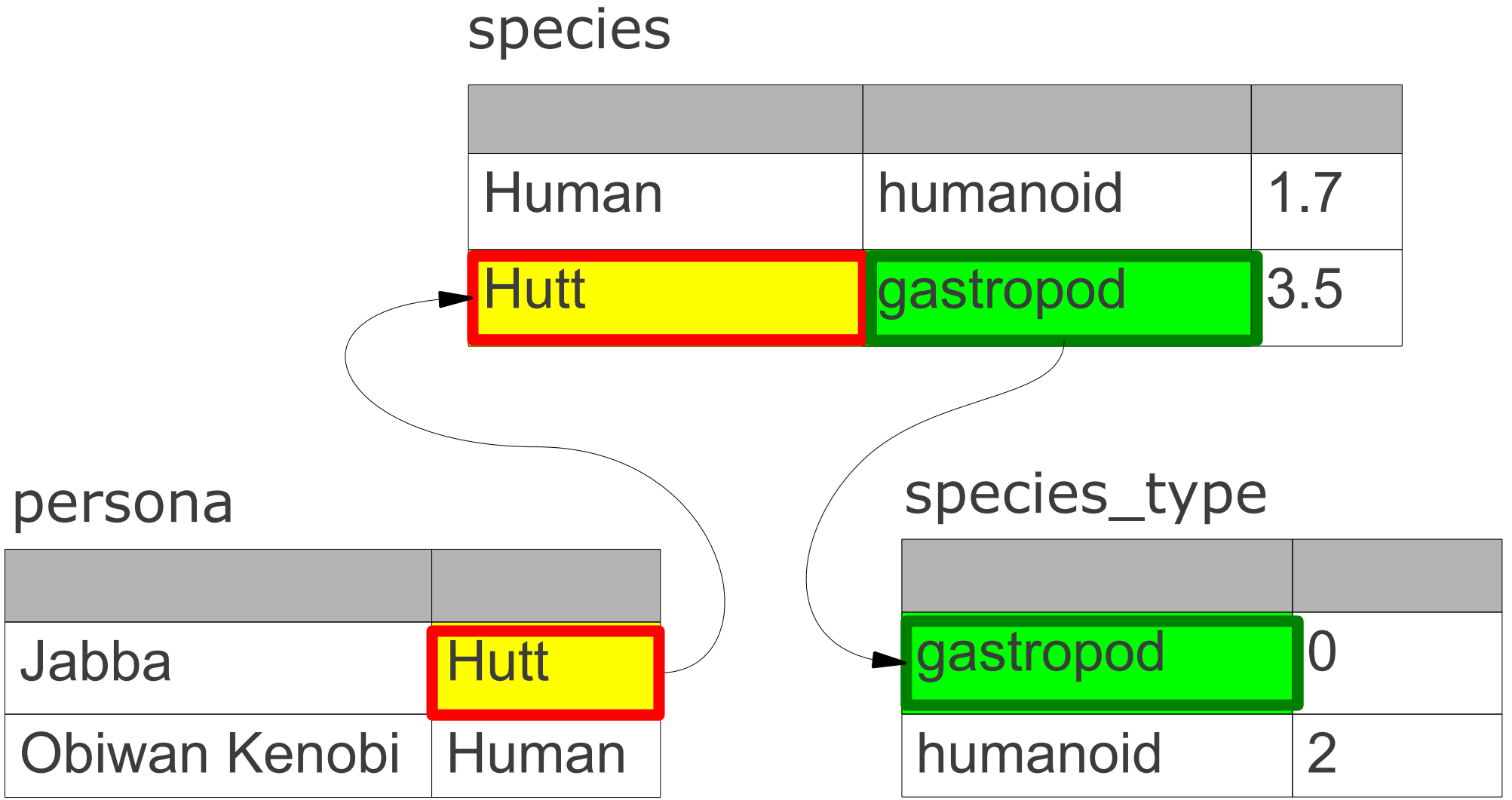
Eatr


restaurant
restaurant_id*
name
price_range
neighborhood
neighborhood_id*
name
comment
comment_id*
date_posted
comment_text
list_membership
user
user_id*
username
real_name
list
list_id*
name

Mapping ER to Rel.

0. Break up M:M entities
1. Each entities becomes a tables
   (attributes become fields / columns)
2. What about the relationships?

Linking the Tables



Keys

A Candidate Key

a set of fields that can uniquely identify a row in a table

The Primary Key (PK)

a candidate key that we decided to use to identify rows in a particular table

Examples of Keys

student

name? student id? Utorid
email? date of birth?

restaurant

name? city? Address?

comment

text? time? user?

Natural vs Surrogate

A "Natural" Key

based on an existing attribute
e.g.: email, existing codes
☺ easy to remember
☹ may have to change

A "Surrogate" Key

an arbitrary identifier
☹ hard to remember
☺ never have to change

Usually a better option

Does Every Table Need a PK?

Strictly speaking, no. But it often helps, and almost never hurts.

So, as a rule of thumb:

add a surrogate PK to each table, except those representing associative entities.

Choosing PKs

restaurant:

```
restaurant_id integer
```

neighborhood:

```
neighborhood_id integer
```

comment:

```
comment_id integer
```

user:

```
user_id integer
```

CREATE TABLE

```
create table restaurant (
   restaurant_id integer,
   name varchar(100),
   price_range integer
);
```

NOT NULL

```
create table restaurant (
   restaurant_id integer
      not null,
   name varchar(100) not null,
   price_range integer
);
```

PRIMARY KEY

```
create table restaurant (
   restaurant_id integer
      not null,
   name varchar(100) not null,
   price_range integer,
   primary key (restaurant_id)
);
```

AUTO_INCREMENT

```
create table restaurant (
   restaurant_id integer
      not null auto_increment,
   name varchar(100) not null,
   price_range integer,
   primary key (restaurant_id)
);
```

Foreign Key

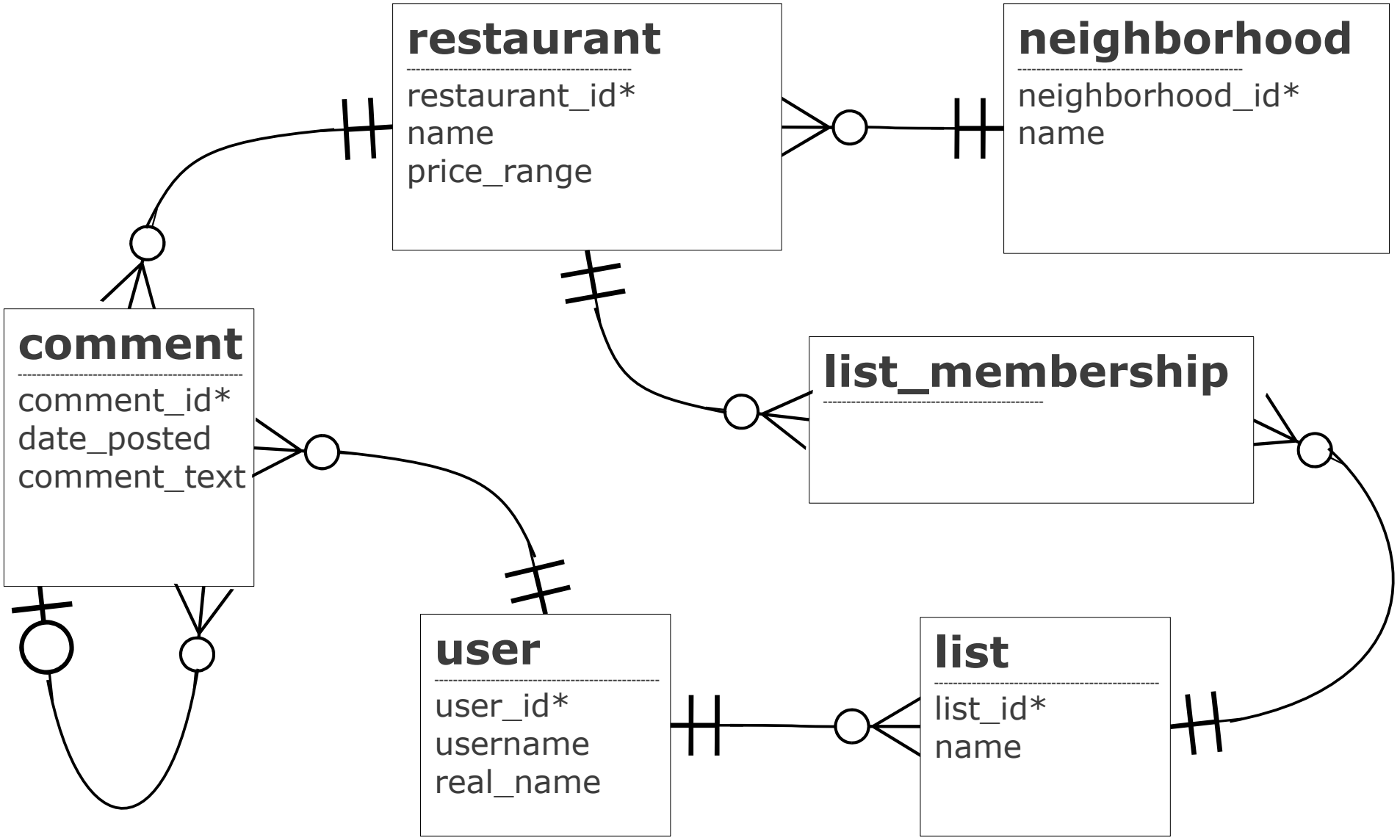
A PK of another table

An attribute that contains a primary key of another table, with a constraint that the corresponding row exists in the other table. (A FK is always itself a “key”.)

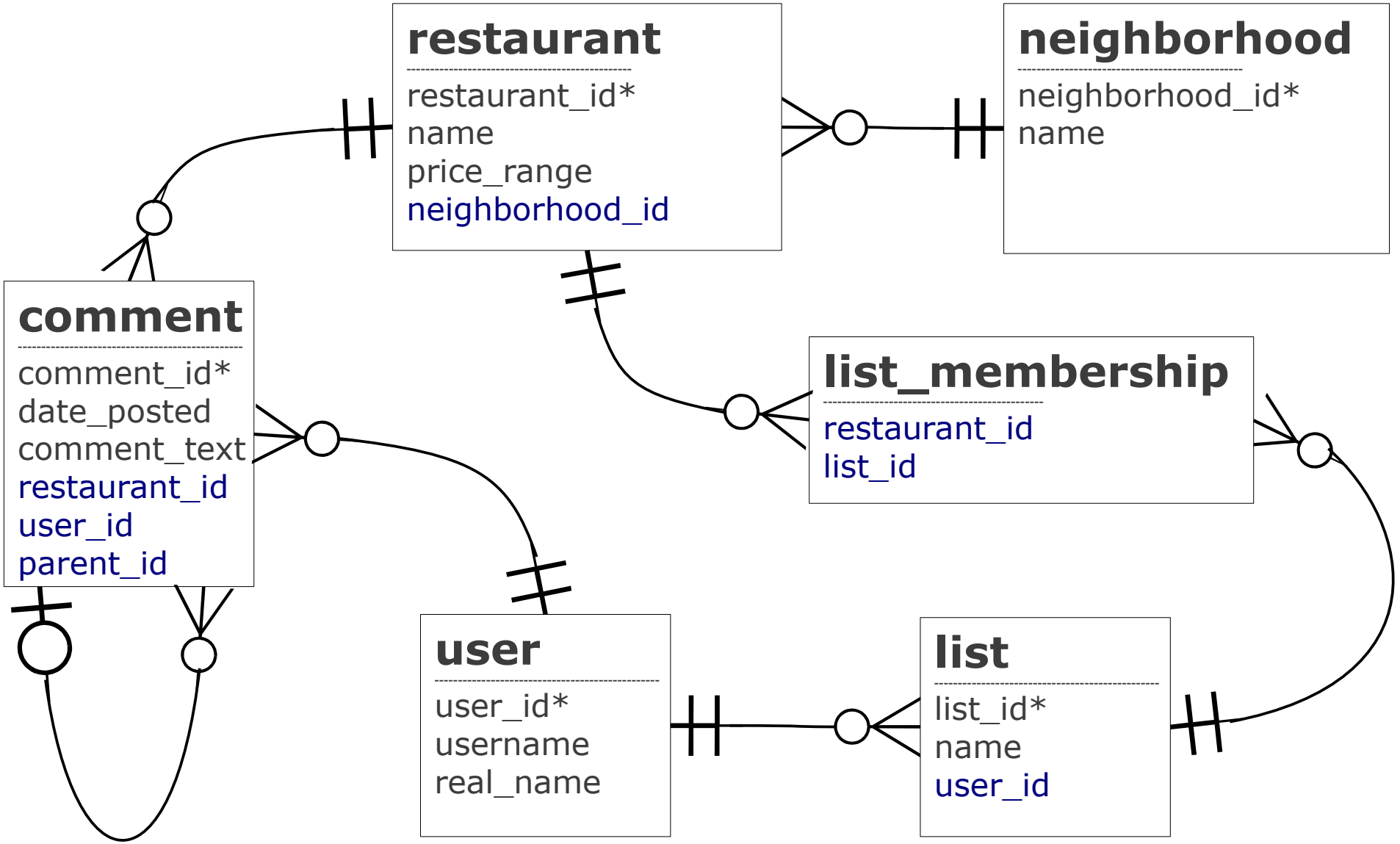
Implementing 1:M

Every table representing an entity on the "M" side of a relationship gets a FK pointing to the PK of the entity on the "1" side of that relationship.

Implementing 1:M



Implementing 1:M



Implementing a FK

```
create table restaurant (
   restaurant_id integer
       not null auto_increment,
   name varchar(100) not null,
   price_range integer,
   neighborhood_id integer,
   primary key (restaurant_id),
   foreign key (neighborhood_id)
   references
     neighborhood(neighborhood_id)
);
```

ON DELETE

```
create table restaurant (
   restaurant_id integer
   ...
   neighborhood_id integer,
   primary key (restaurant_id),
   foreign key (neighborhood_id)
   references
     neighborhood(neighborhood_id)
   on delete cascade
);
```

alternatives: “set null”, “restrict”.

Associative Entities

```
create table list_membership (
   list_id integer not null,
   restaurant_id integer not null,
   primary key
     (list_id, restaurant_id),
   foreign key (list_id)
    references list(list_id),
   foreign key (restaurant_id)
    references
     restaurant(restaurant_id),
);
```

Recursion

```
create table comment (
   comment_id integer not null,
   ...
   parent_id integer,
   primary key (comment_id),
   ...
   foreign key (parent_id)
    references comment(comment_id)
);
```

Questions?

Optional / Mandatory

On the 1 side:

Use “not null” on the FK.

On the M side:

Can’t be mandatory. (It will have to be optional.)

1:1 Relationships

Option 1:

Use the same table.

Option 2:

Use a single-attribute FK
as the PK in one of the tables.

Multivalued Attributes

customer:

- name
- phone number(s)
- email addresse(s)

restaurant:

- name
- address
- tag(s)

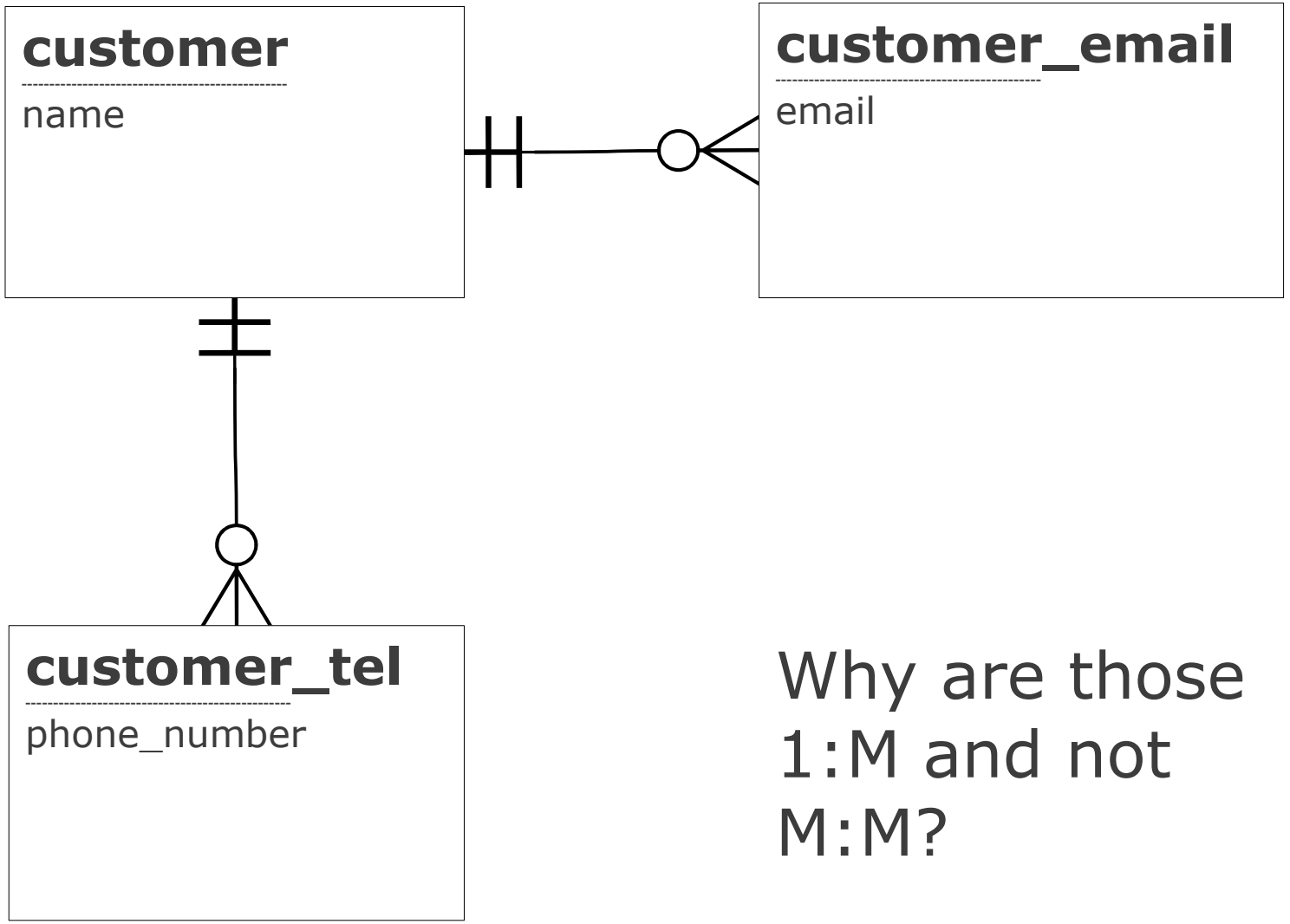
Multivalued Attributes

Problem:

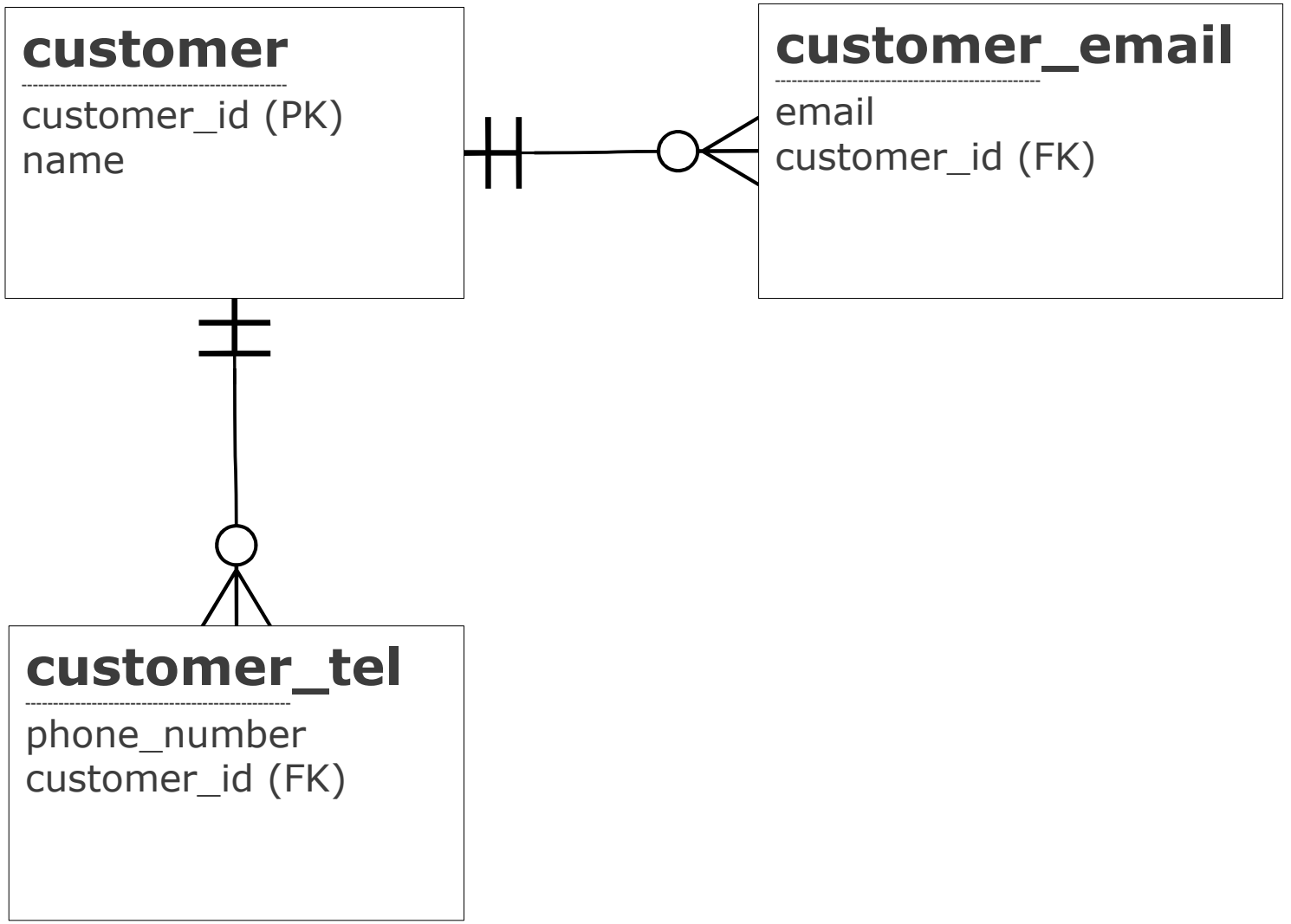
Multivalued attributes may be ok in ER, but definitely not in a relational database.

Solution:

Treat multivalued attributes as simple entities.



Why are those 1:M and not M:M?


customer
customer_id (PK)
name
customer_email
email
customer_id (FK)
customer_tel
phone_number
customer_id (FK)

```
create table customer_email (
   email varchar(100),
   customer_id integer not null,
   primary key
     (customer_id, email),
   foreign key (customer_id)
     references customer(list_id)
);
```

Are we missing anything?

```
create table customer_email (
   email varchar(100),
   customer_id integer not null,
   position integer,
   primary key
     (customer_id, email),
   foreign key (customer_id)
    references customer(list_id)
);
```

ER for M