

INF1343, Winter 2012

Data Modeling and Database Design

Yuri Takhteyev

Faculty of Information
University of Toronto



This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

Week 4

Converting an ER Design into a Relational Form

Drawing Software

Options for software:

- OpenOffice Draw
 - Free / open source
 - Available in the lab
 - **You can get “Crow’s Foot” templates at http://takhteyev.org/courses/12W/inf1343/Crows_Feet_Arrows.odg**
 - Alternatively, do UML notation (“n..m”) by hand
- Microsoft Visio
- Your favorite software

Common Patterns

Hierarchical (1:M)

vs.

Not (Quite) Hierarchical (M:M)

A contains B

1:M

building \rightarrow room

cd \rightarrow track, book \rightarrow chapter

car \rightarrow part

province \rightarrow riding

neighborhood \rightarrow restaurant

session \rightarrow prediction

invoice \rightarrow billable item

A contains B

M:M

course $\succ\leftarrow$ student

list \leftarrow restaurant

dish $\succ\leftarrow$ 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 \rightsquigarrow company

instructor \rightsquigarrow course

Belonging to Different Entities

1:M

user \leftarrow comment \succ restaurant
customer \leftarrow session \succ f. teller

B “instantiates” A

1:M

species \rightarrow pet

model \rightarrow vehicle

book \rightarrow edition

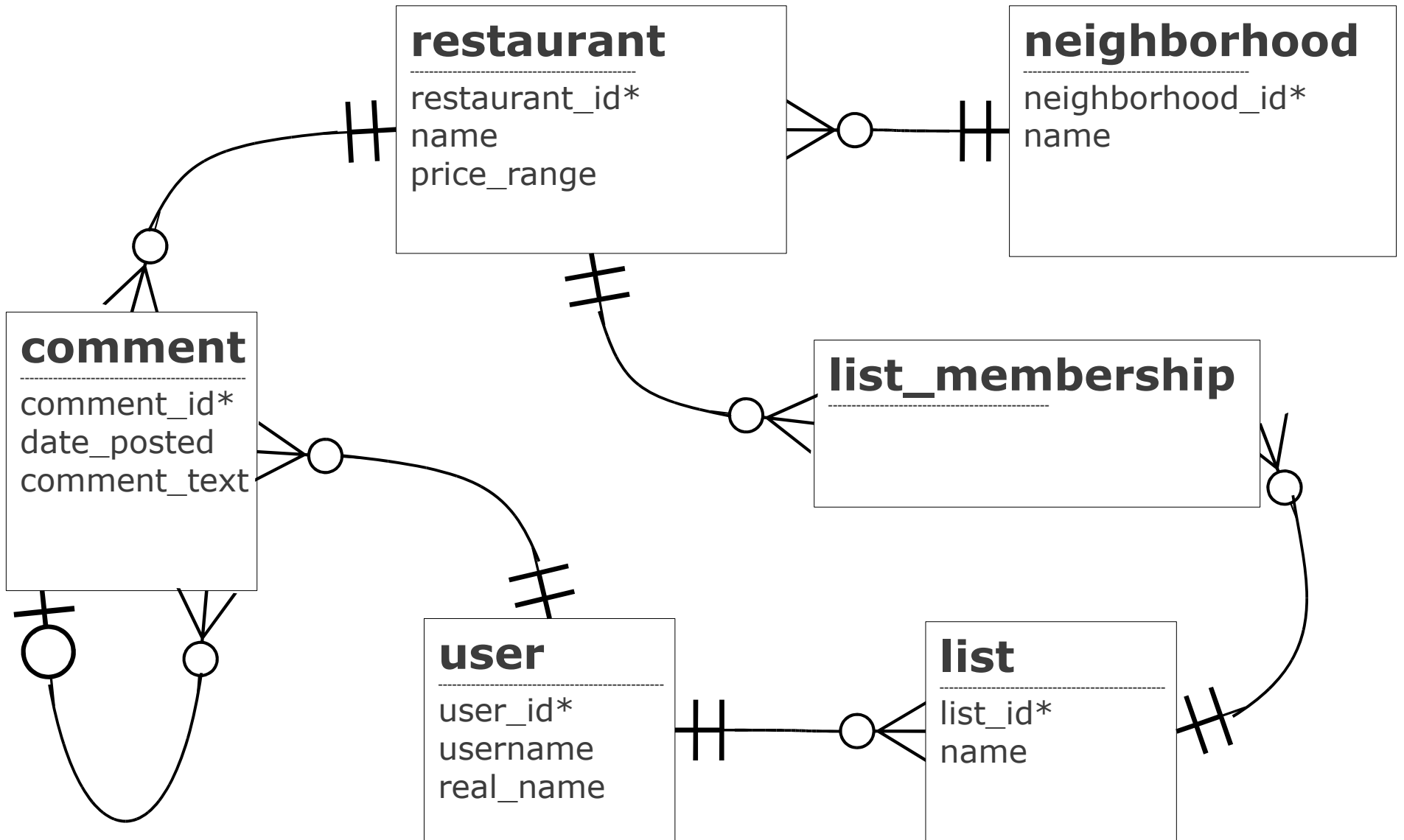
course \rightarrow course_instance

(“CCT395” vs “CCT395 in Fall 2011”)

M:M

employee \rightarrow \rightarrow role

Eatr



Mapping ER to Rel.

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

Linking the Tables

species

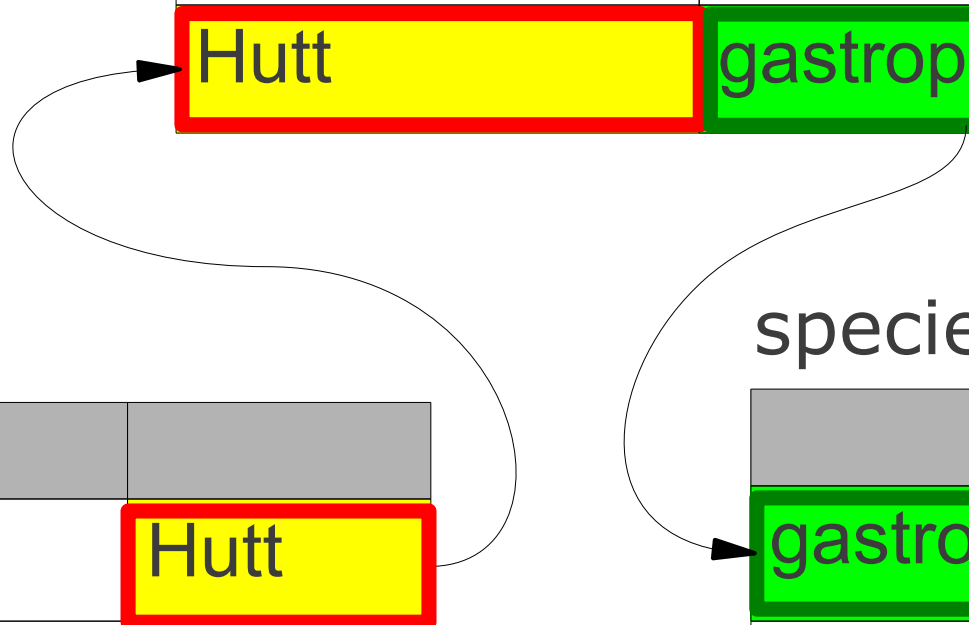
Human	humanoid	1.7
Hutt	gastropod	3.5

persona

Jabba	Hutt
Obiwan Kenobi	Human

species_type

gastropod	0
humanoid	2



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 as the primary method for identifying 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

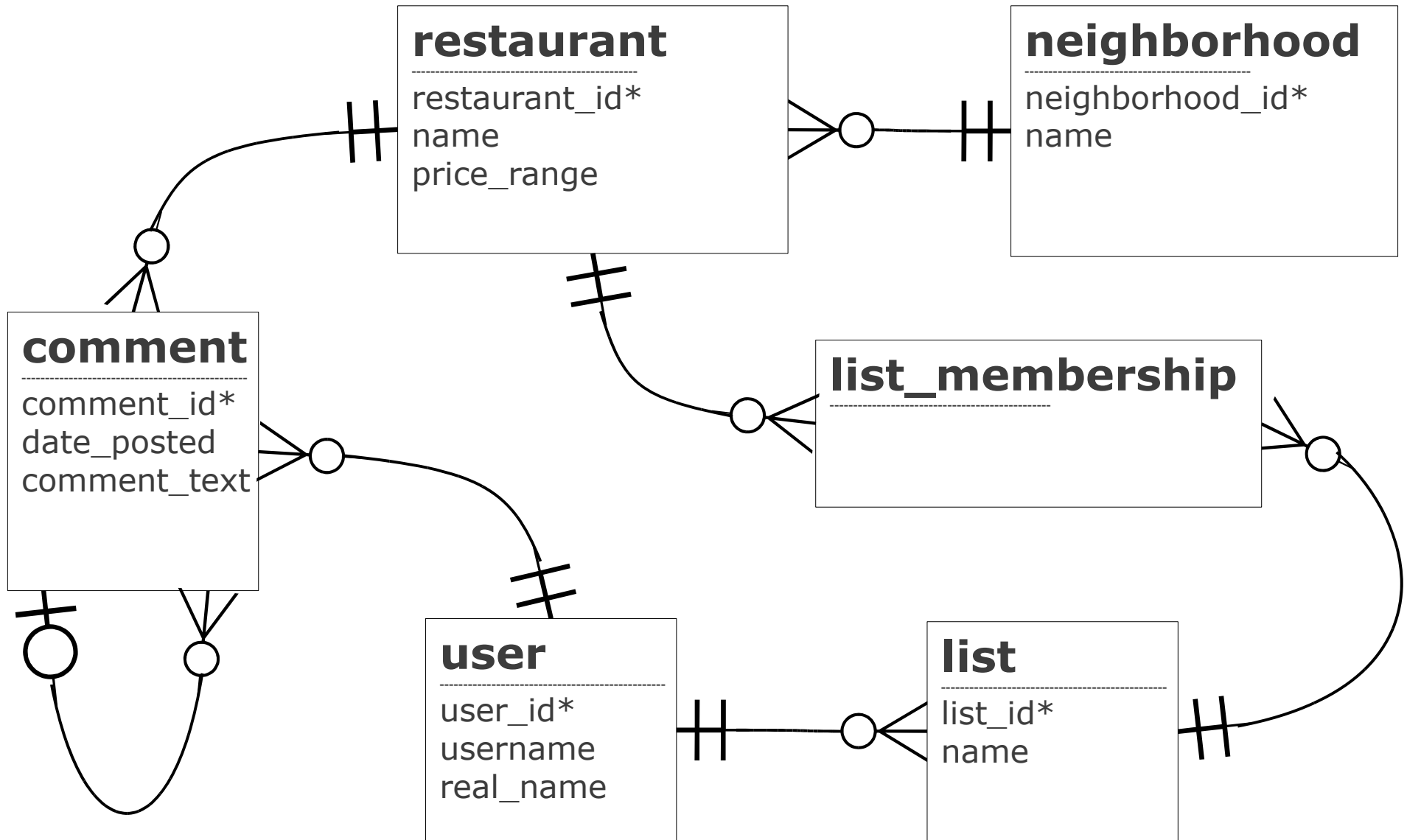
A PK of another table

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 normally **not** itself a “key”.)

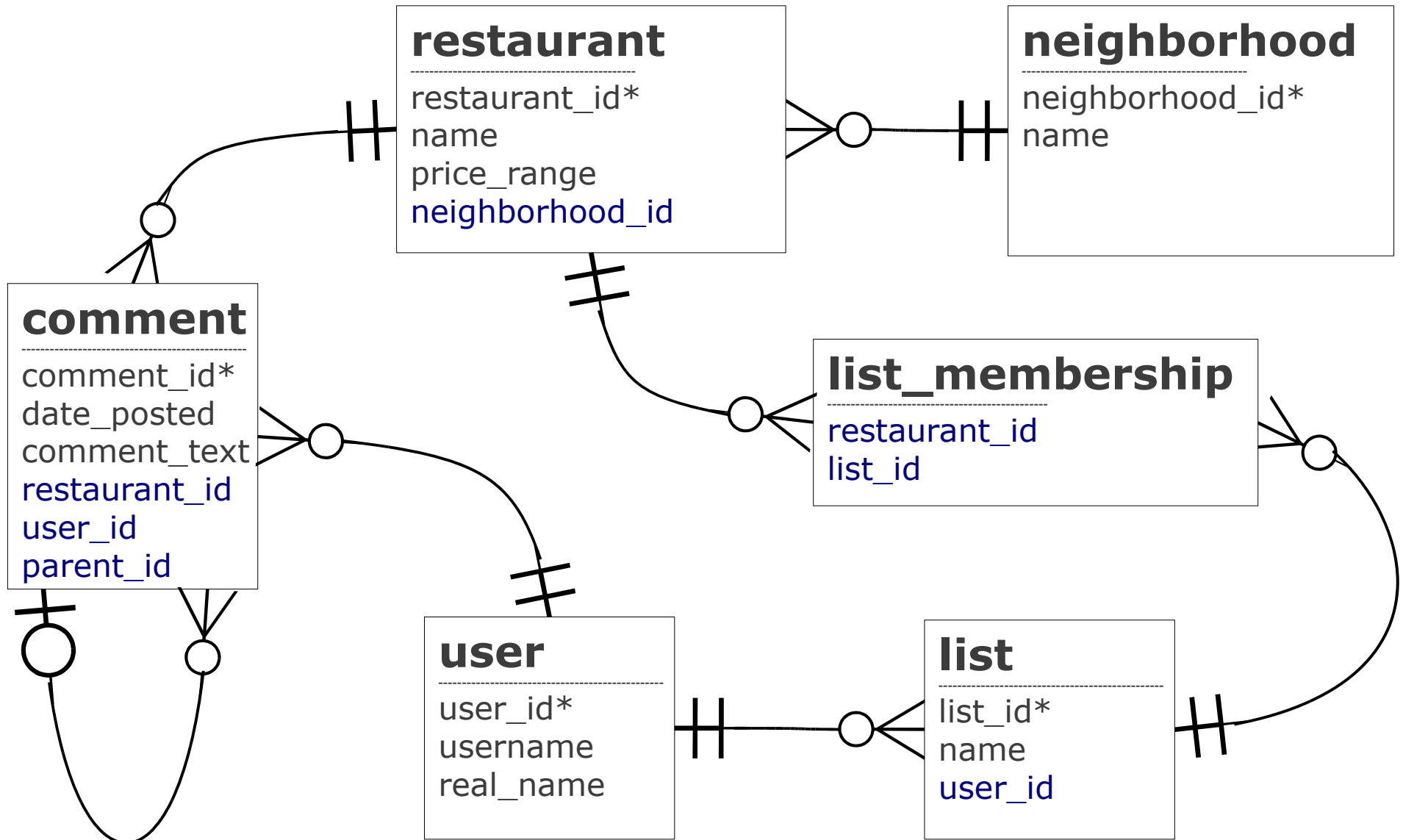
Implementing 1:M

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 address(es)

restaurant:

name

address

tag(s)

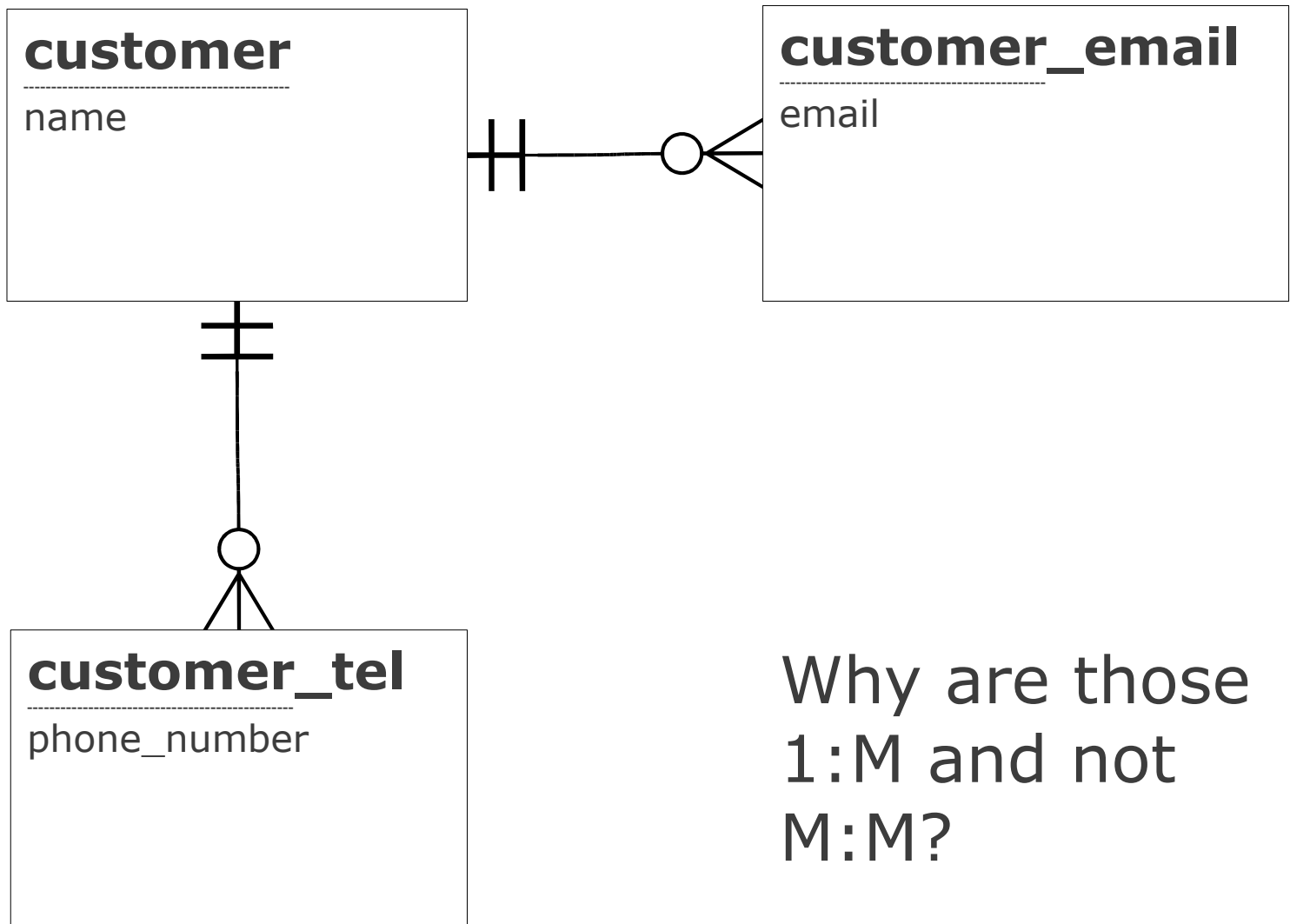
Multivalued Attributes

Problem:

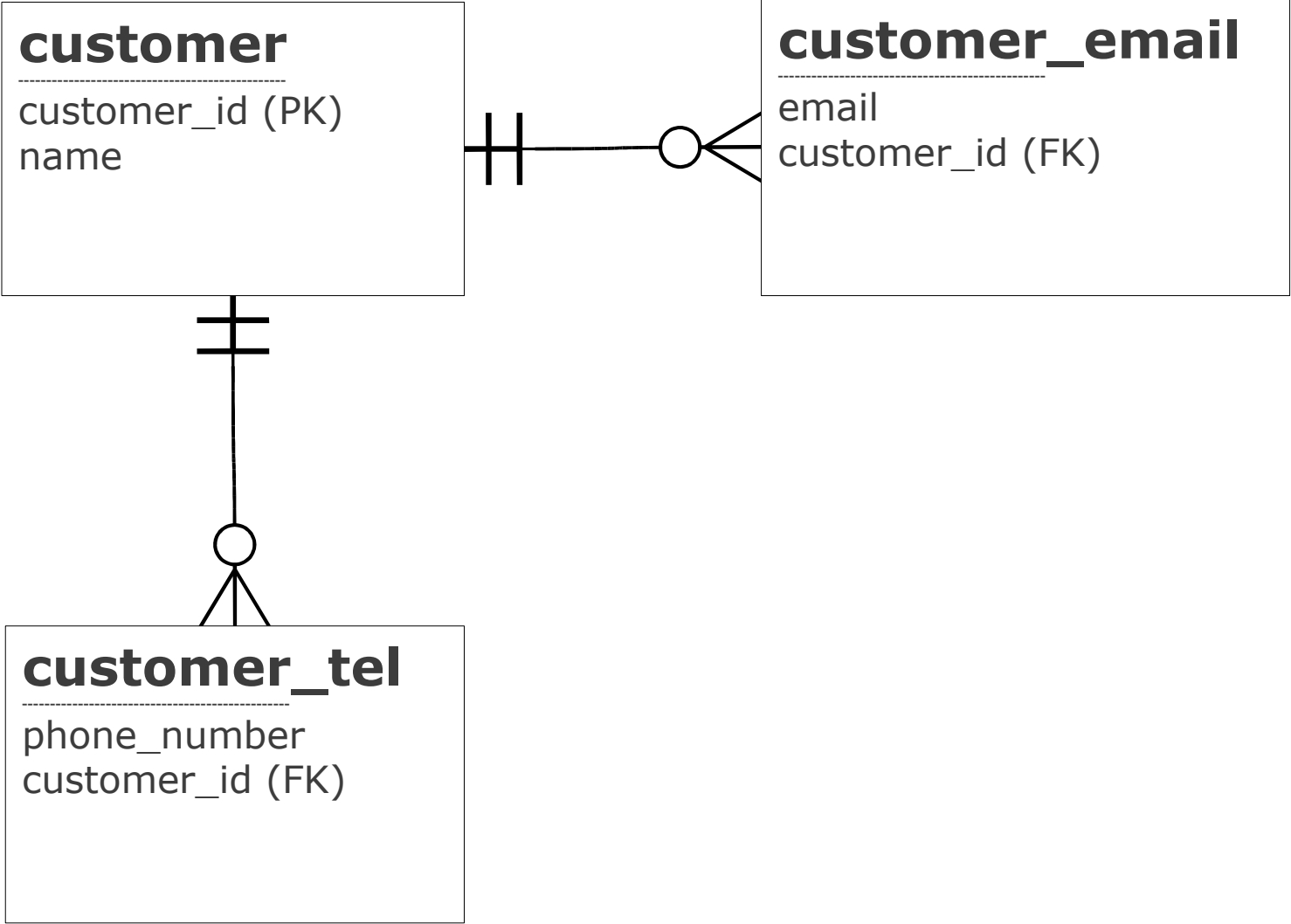
Multivalued attributes may be ok in ER, but definitely not in a relational database.

Solution:

Treat multivalued attributes as simple entities.



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



```
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)  
);
```