Advanced Backend as a Service (BaaS)

Agenda

You will continue building your **Dianping** mini-program. Enter the user 🙋‍♀️

Consider `reviews`

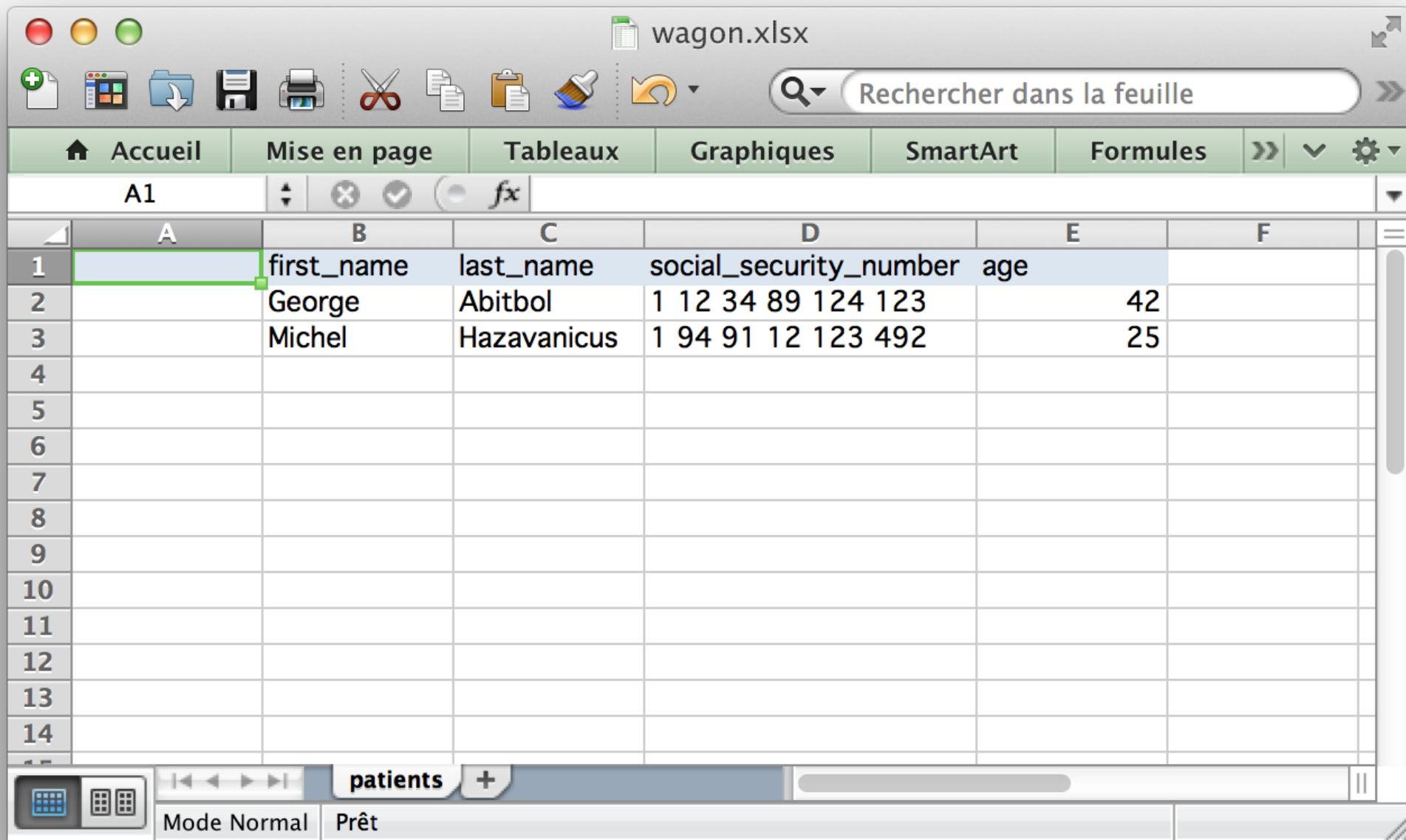Users can post reviews on restaurants. So relationship-wise:

- Each restaurant can be reviewed by many users, and
- Each user can review many restaurants

This is called a **many-to-many relationship**.

Many to Many Relationships

Excel++

Let's go further

wagon.xlsx

Rechercher dans la feuille

Accueil | Mise en page | Tableaux | Graphiques | SmartArt | Formules

A1

	A	B	C	D	E	F
1		first_name	last_name	social_security_number	age	
2		George	Abitbol	1 12 34 89 124 123	42	
3		Michel	Hazavanicus	1 94 91 12 123 492	25	
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

patients +

Mode Normal | Prêt

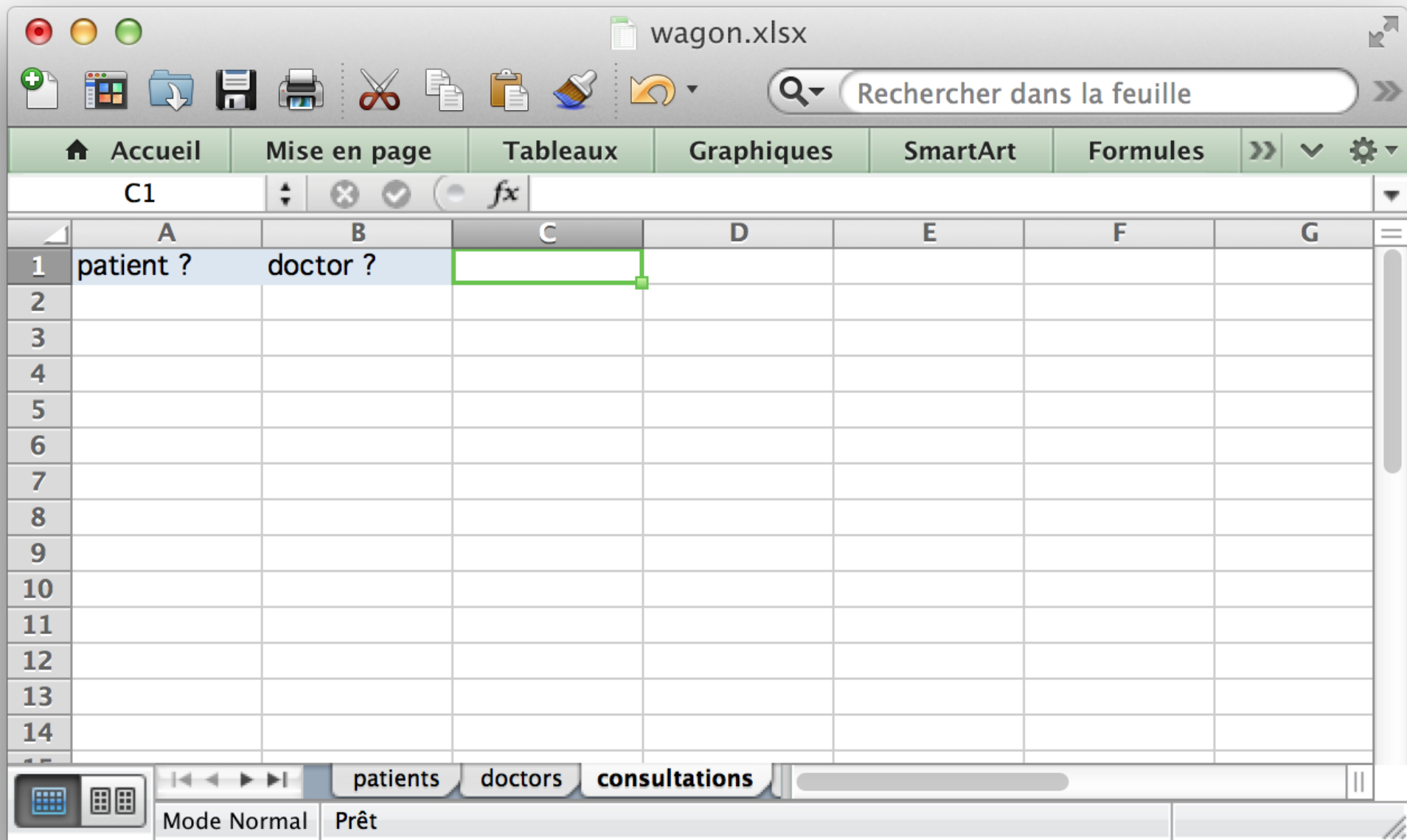|	A	B	C	D	E	F
1		first_name	last_name	specialty		
2		Sigmund	Freud	Psychology		
3		Henri	Castafolte	Robotics		
4		John	Doe	Cardiag Surgery		
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

patients | **doctors** | +

Mode Normal | Prêt

- **One** doctor can have **many** patients
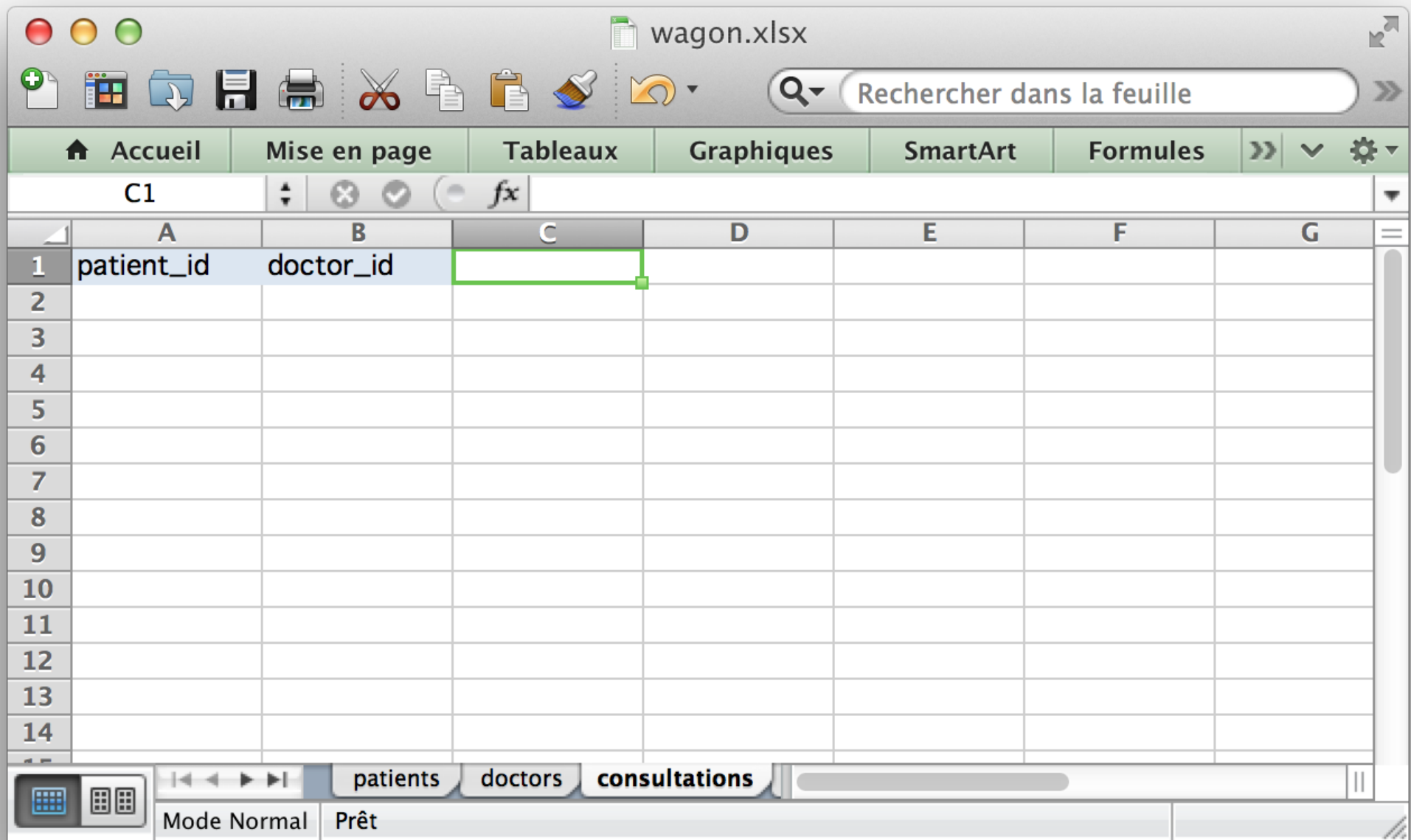
- **One** patient can see **many** doctors

How do we manage this relationship? 🤔

Consultations!

	A	B	C	D	E	F	G
1	patient ?	doctor ?					
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

patients | doctors | **consultations**

Another problem - what if we have two doctors named Mary?...

...and two patients named George?

	A	B	C	D	E	F	G
1	patient_id	doctor_id					
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

patients doctors **consultations**

Mode Normal   Prêt

	A	B	C	D	E	F
1	id	first_name	last_name	social_security_number	age	
2		George	Abitbol	1 12 34 89 124 123	42	
3		Michel	Hazavanicus	1 94 91 12 123 492	25	
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

patients | doctors | consultations

Mode Normal   Prêt

wagon.xlsx

Accueil | Mise en page | Tableaux | Graphiques | SmartArt | Formules

A4

	A	B	C	D	E	F
1	id	first_name	last_name	social_security_number	age	
2	1	George	Abitbol	1 12 34 89 124 123	42	
3	2	Michel	Hazavanicus	1 94 91 12 123 492	25	
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

patients | doctors | consultations

Mode Normal | Prêt

	A	B	C	D	E	F
1	id	first_name	last_name	specialty		
2	1	Sigmund	Freud	Psychology		
3	2	Henri	Castafolte	Robotics		
4	3	John	Doe	Cardiag Surgery		
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

patients **doctors** consultations

wagon.xlsx

	A	B	C	D	E	F	G
1	patient_id	doctor_id					
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

patients | doctors | **consultations**

Mode Normal   Prêt

George Abitbol (**id = 1**) has seen Doctor John Doe (**id = 3**)

n:n relation (many to many)

A patient **has many** doctors and a doctor **has many** patients.

You can download this example: consultations.xslx

Relational Database

1:n
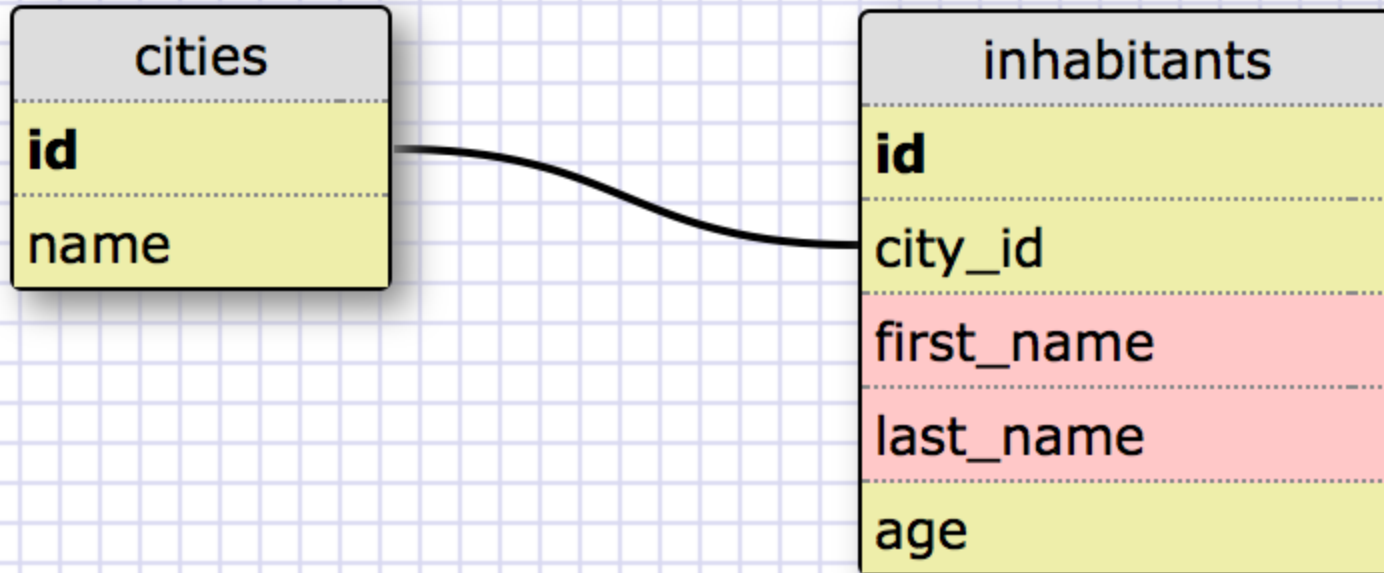
n:n ?

patients

id	
first_name	
last_name	
social_security_number	
age	

doctors

id	
first_name	
last_name	
specialty	

n:n

consultations
- doctor_id
- patient_id
- created_at

patients
- **id**
- first_name
- last_name
- social_security_number
- age

doctors
- **id**
- first_name
- last_name
- specialty
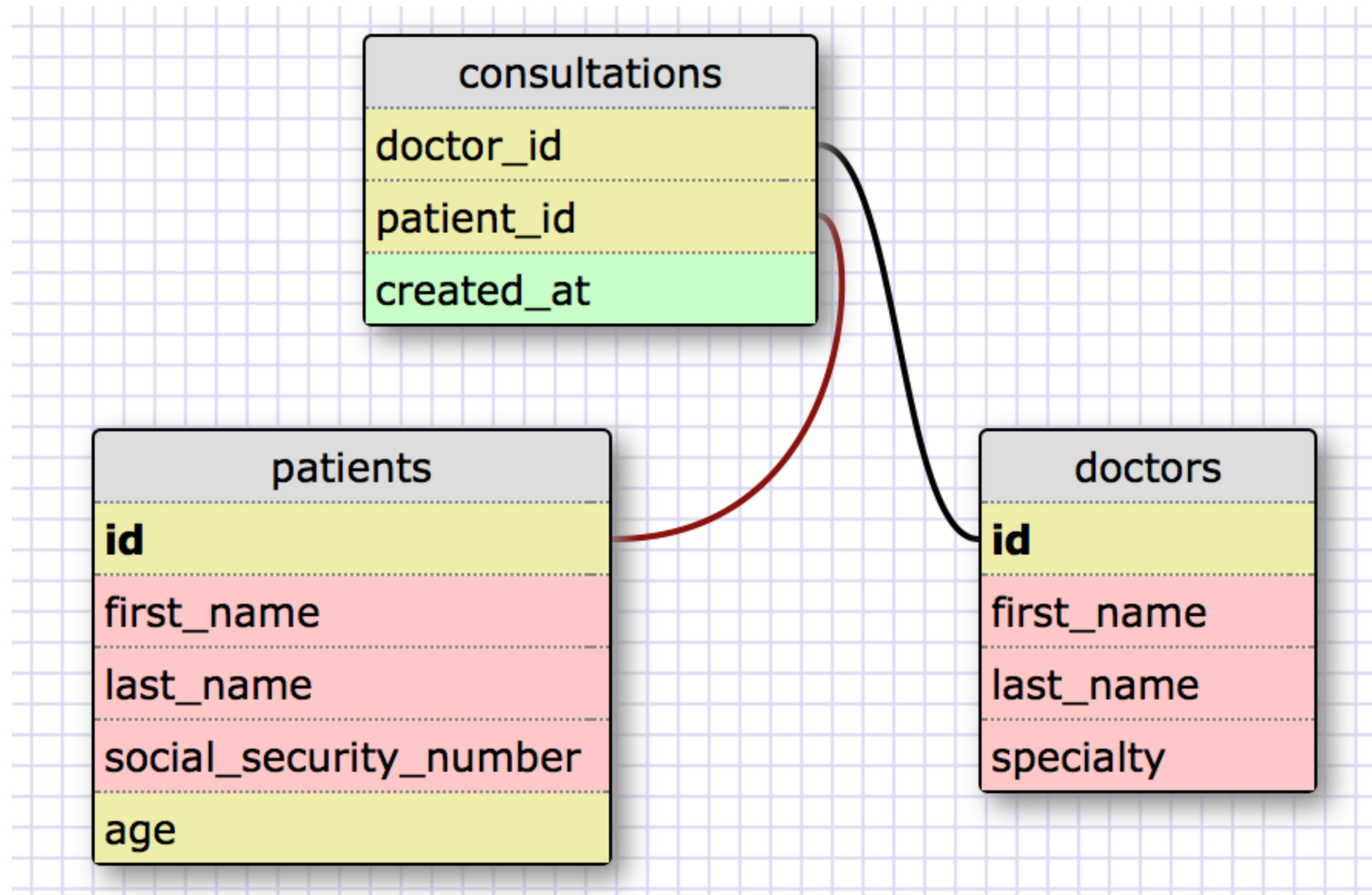
Vocabulary

- A schema is composed of **tables**.

- Each table has a set of **columns**.

- When inserting data in a table, you create a **record** in a new **row**.

- A table between two n:n tables (in this case `consultations`) is a **joint table**.

DB Schema Composer

db.lewagon.com

Core User Journey

Notice how service products - like Eleme, Tmall, Airbnb - have the same basic types of functionality

Take **Dianping** for example!

User Login

As a user, I can log in

Restaurant Index

As a user, I can view all restaurants

Restaurant Page

As a user, I can view one restaurant

Review Index

As a user, I can read and write reviews for a restaurant
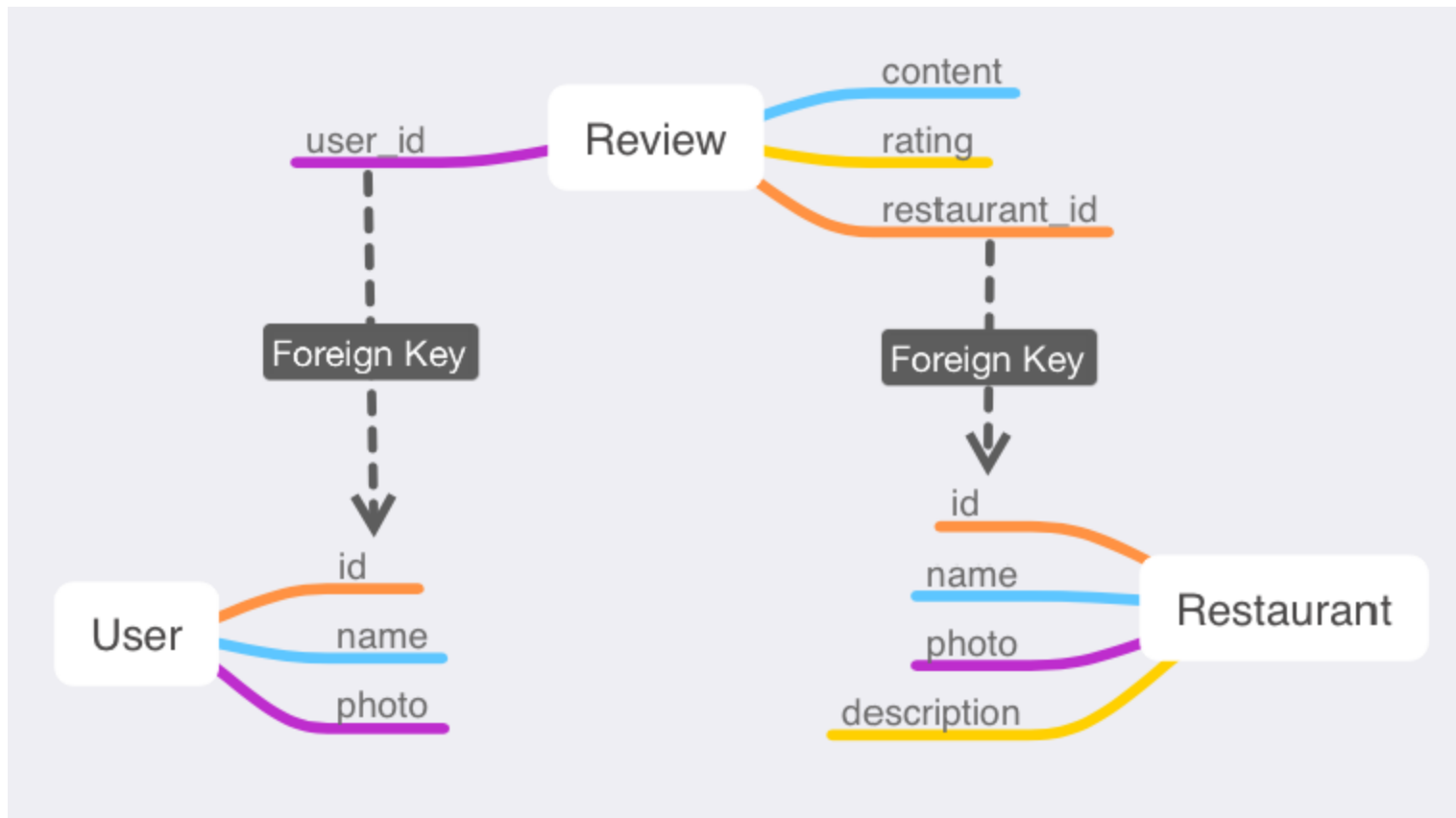
We will cover more advanced user actions later on

Dianping Data Schema

Review relationship is **many-to-many**

Therefore, review's relationship can be its own **(joint)** table
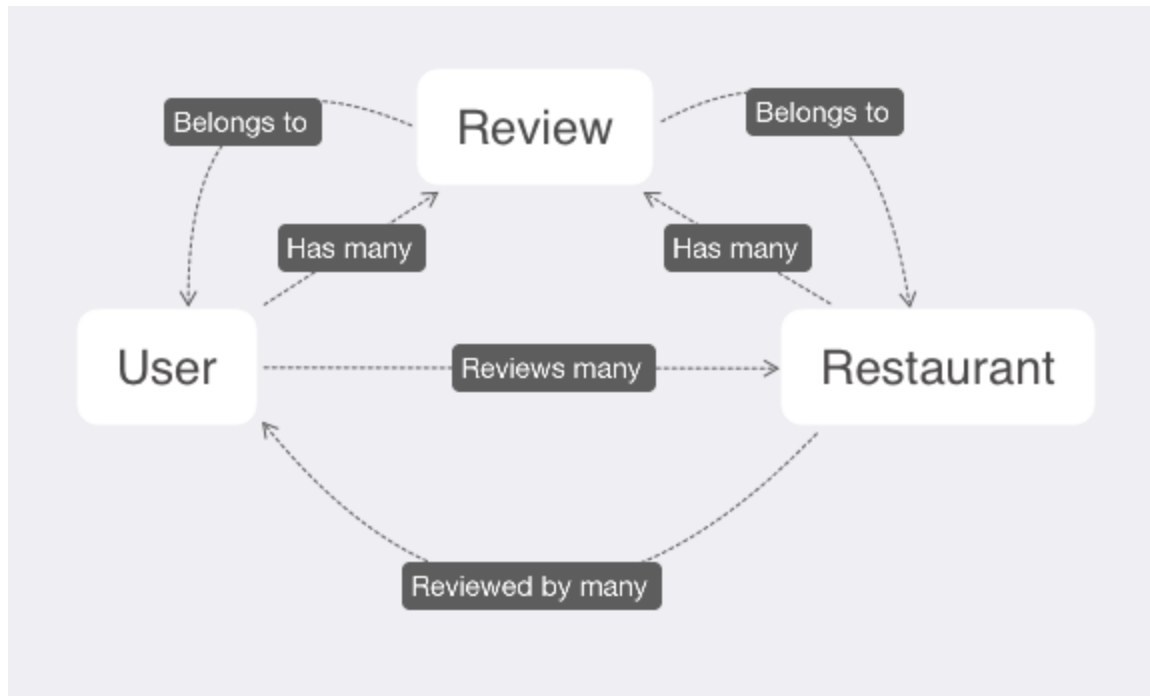
Schema

Livecode! Let's update our database 👩‍💻

We only want logged in users to execute certain functions (like adding a new restaurant)

Create table ✕

Table name
restaurants

Data sheet notes
There will be less bugs if your comments are well written.

──────────────── ACL ⑦ ────────────────

Set data table entry permission (new)

○ Public   ● Logged-in user   ○ User group   ○ Private
需要先创建临时用户

Set the default read and write permissions for the row (query, modify, delete)

Public   ☑ Read   ☐ Write
                              需要先创建临时用户

Logged-in user   ☑ Read   ☑ Write

Creator   ☑ Read   ☑ Write

User group   ☑ Read   ☑ Write

Cancel   Create

Your turn!

EXERCISE 1: UPDATE YOUR DATABASE 💪

User login

The beauty of being in WeChat, is that we don't have to ask the user to register. We let them login using their WeChat profile!

How does **WeChat login** work?

1. In the mini program, we ask user to authorize us to use their **public** WeChat user info

2. We use the Minapp SDK to create a user in our BaaS with a **permanent unique** ID

To make the first step, we need to use a button which triggers the `getUserInfo` API. It looks something like this:

```
<button open-type="getUserInfo" bindgetuserinfo="userInfoHandler">微信 Login</button>
```

> If `bindtap` triggers a JS function on a tap, what does `bindgetuserinfo` do? 🤔

> "Let's put this button right at the start! 😃" – new developer

⚠️ Not a good idea! Tencent is protective of it's users and might block your mini program!

Instead, let's create a `user` page

Time to implement our `userInfoHandler` !

The awesome folks at Minapp give us a `wx.BaaS.auth.loginWithWechat()` function

```
userInfoHandler: function(data) {
    wx.BaaS.auth.loginWithWechat(data).then(user => {
        // do something with the 'user' object
      }, err => {
        // might need to log the error message
    })
  }
```

User in the BaaS

Live Code 3: WeChat Login 👩‍💻

We can then start adding some view logic to the user page…

Bind data to display user's profile

```
<!-- user.wxml -->
<view wx:if="{{currentUser}}">
  <text>Welcome back {{currentUser.nickName}}!</text>
</view>

<view wx:else>
  <!-- login button -->
</view>
```

Note: Both blocks of code are written in `user.wxml`. Therefore, we need to set specific **conditions** for different views

After the user logs in once, we want to "remember" them

1. We can use the `getCurrentUser()` function provided by the BaaS

```js
// user.js in onLoad function
wx.BaaS.auth.getCurrentUser().then(dosomething).catch(function(err) {
  wx.showModal({
    title: 'Error',
    content: err.message
  })
})
```

BUT this is still an API call, so it relies on network

2. We can cache the `user` object on the users phone

```
// inside the getCurrentUser or loginWithWechat callback
wx.setStorageSync('user', user);
```

How (and when) do we retrieve the user object? 🤔 Have a look at getStorageSync()

Your turn!

EXERCISE 3: USER LOGIN 💪

Happy 2nd Day of BaaS 🤩