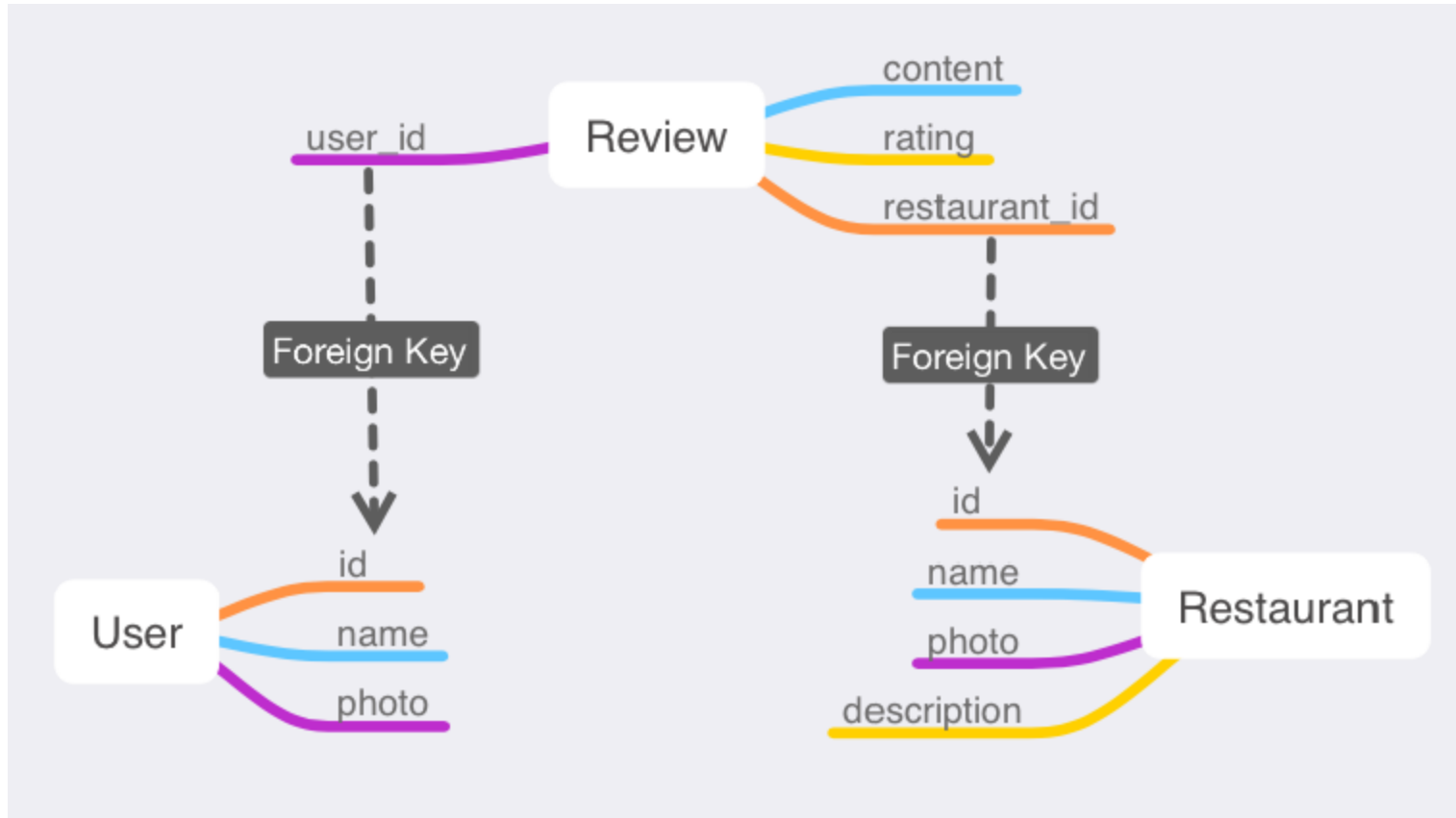


## **Advanced Schema (continued)**

**Let's Recap**

## Many-to-many relationship



On Tuesday, we learnt the most common operation in **CRUD** - **READ**

- As a public user, I can view all restaurants
- As a public user, I can view one restaurant
- As a public user, I can view all reviews for one restaurant

On Thursday, we created a fully-functioning WeChat **login** system

- As a user, I can log in

This afternoon, we will add **more features** to our mini program

As a logged-in user, I can now post reviews

写点评

Homeslice Pizza(158坊店)

发表

总体

Best pizza in Shanghai!

再加80个字，3张图有机会赢取100积分!

完成

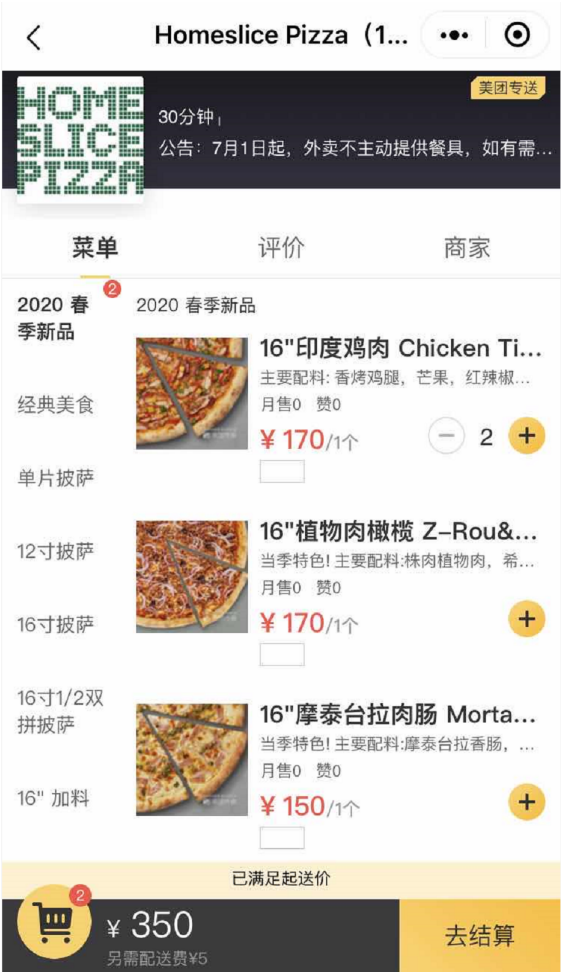
q w e r t y u i o p

a s d f g h j k l

z x c v b n m

123 地球 麦克风 space return

As a logged-in user, I can now place an order





**CRUD - CREATE**

## A 3-step process with the SDK

### 1. Create an empty record

```
// define table object
let Review = new wx.BaaS.TableObject('reviews')
// create a new record in that table
let review = Review.create()
```

### 2. Give the record some data

```
review.set(data)
// for example review.set({'rating': 5})
```

### 3. Store data to the backend

```
review.save() //.then(dosomething) if needed
```

We can implement the CREATE function for different use cases

## Live Code 1: Posting Reviews

## Reviews form

```
<!-- at bottom of show.wxml -->
<form wx:if="{{currentUser}}" bindsubmit="createReview">
  <view class="section">
    <input name="content" placeholder="put down your thoughts"/>
  </view>
  <button formType="submit" class="footer-btn">Submit</button>
</form>

<button wx:else bindtap="showUserPage" class="section">
  Please Log In to Review
</button>
```

**Bonus: Using WeChat Components**

## Scroll picker

```
<!-- in review form of show.wxml -->  
<picker mode="selector" range="{{ [1, 2, 3, 4, 5] }}" bindchange="onRate">Rating</picker>
```

## bindchange event handler

```
// show.js  
onRate: function(event) {  
  console.log(event) // find where is the value we need  
}
```

## Create review

```
// show.js
createReview: function(event) {
  // ...
  let Review = new wx.BaaS.TableObject('reviews')

  let review = Review.create()

  let data = {
    // review's content and rating
  }

  review.set(data).save().then(res => {
    // do something with the response
  })
},
```



Your turn!

**EXERCISE 1: POST REVIEWS** 🦾

## Live Code 2: Placing Orders

Two additional tables: `meals` and `orders` .

Which of the two is the **joint** table? 🤔

A user can view all `meal`s for a restaurant

Sounds familiar? Similar to listing a restaurant's reviews!

Except now, a user can place an order for a meal

```
<!-- in wx:for meals loop of show.wxml -->  
<button data-id="{{meal.id}}" bindtap="submitOrder">Order</button>
```

## Create order

```
// show.js
submitOrder: function(event) {
  // ...
  let Order = new wx.BaaS.TableObject('orders')

  let data = {
    // meal_id and current user_id
  }

  Order.create().set(data).save().then(dosomething)
  // relaunch and switch to user profile to display orders
},
```

Display meals ordered

For that, we need to **expand** the meals table so we can include the meal's `name`, `price` and `photo` in the order!



```
// user.js
onLoad: function (options) {
  let page = this
  wx.BaaS.auth.getCurrentUser().then(function(res) {
    page.setData({
      currentUser: res
    })
    // ⚠️
    let Order = new wx.BaaS.TableObject('orders')
    let query = new wx.BaaS.Query()
    let currentUser = page.data.currentUser.id.toString()
    query.compare('user_id', '=', currentUser)
    Order.setQuery(query).expand(['meal_id']).find().then(function(res) {
      console.log('response ->', res) // if we want to see what's in the response
      page.setData({
        orders: res.data.objects
      })
    })
  })
},
```

⚠️ Subtle but important: we save the `order` after the `currentUser` responds **success**, in its handler function

Your turn!

**EXERCISE 2: PLACE ORDERS** 

**Happy Weekend!**