1.介绍

云开发提供了一个 JSON 数据库，顾名思义，数据库中的每条记录都是一个 JSON 格式的对象。

一个数据库可以有多个集合（相当于关系型数据中的表），集合可看做一个 JSON 数组，数组中的每个对象就是一条记录，记录的格式是 JSON 对象。

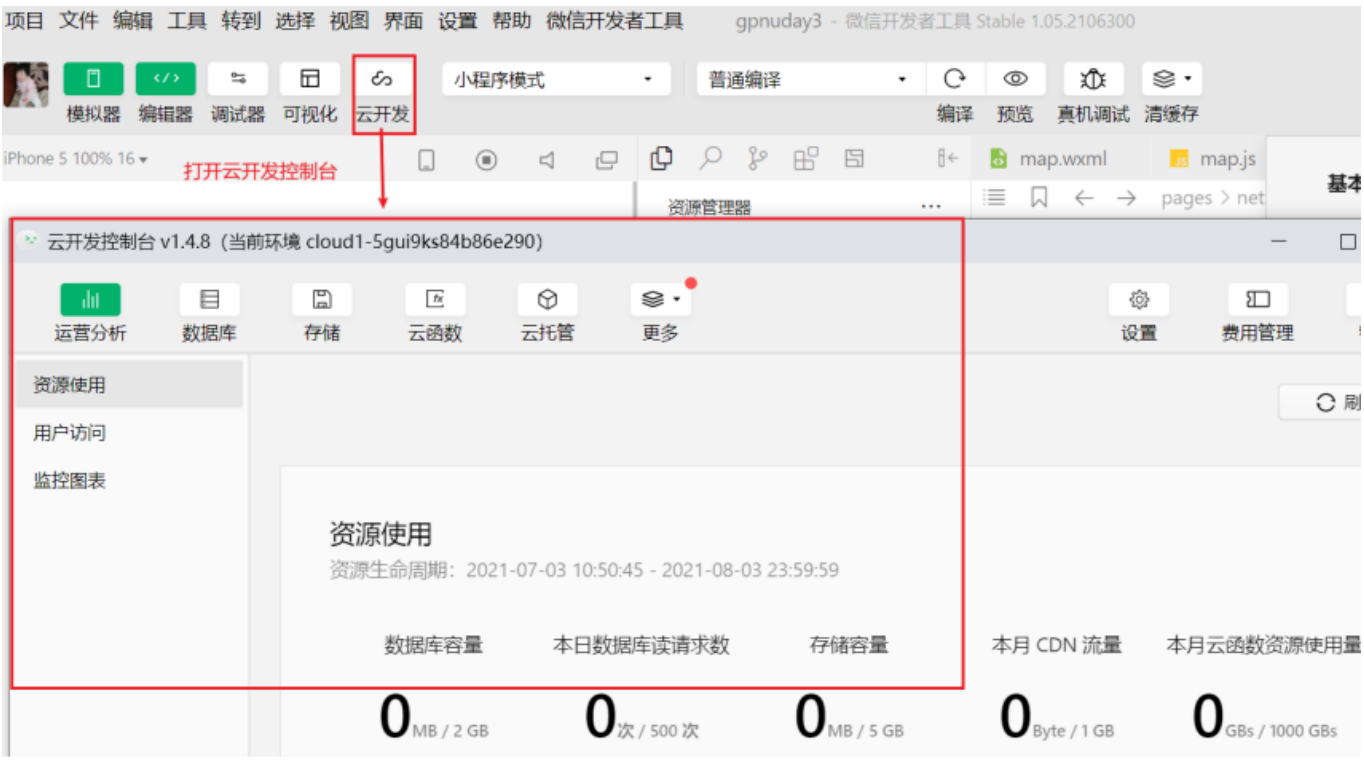
这样的话，数据库的存储也不用考虑了，直接提供了一个类似于 MongoDB一样的数据库，而且免费的存储空间达到了 2G，一般的项目足够使用了。

关系型数据库和 JSON 数据库的概念对应关系如下表：

关系型	文档型
数据库 database	数据库 database
表 table	集合 collection
行 row	记录 record / doc
列 column	字段 field

2. 云数据库操作

注意：操作数据库需要先创建集合名称

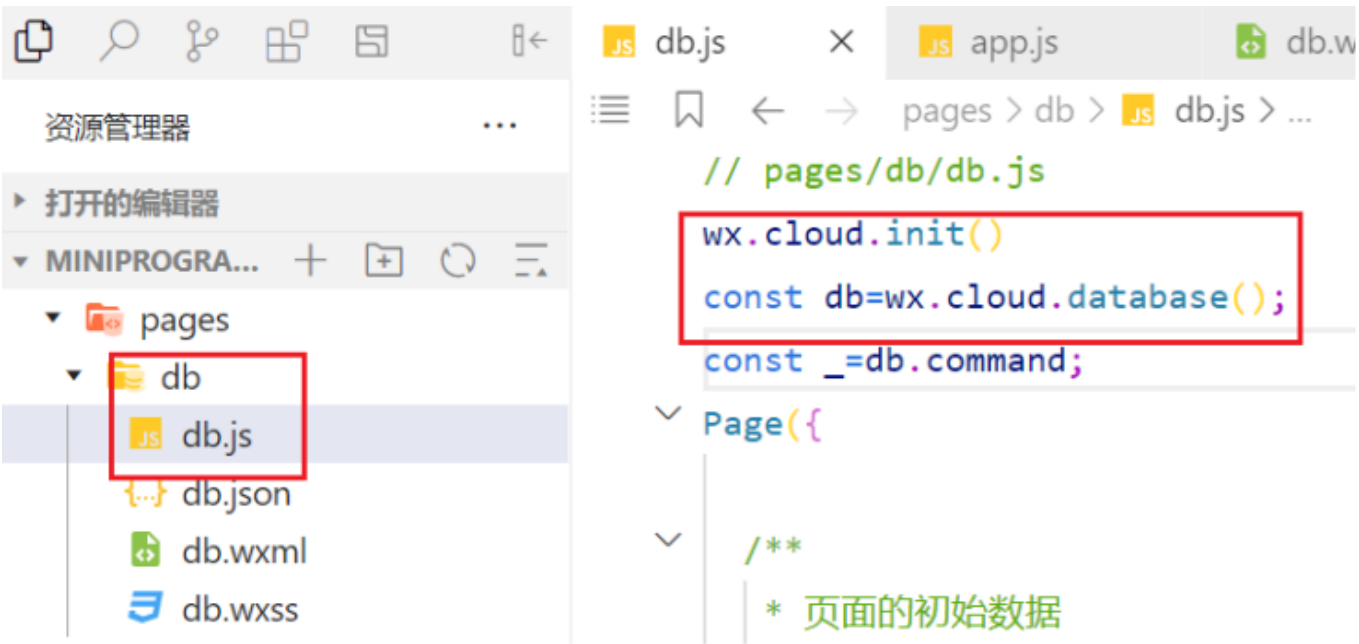


效果

1.新建一个db页面

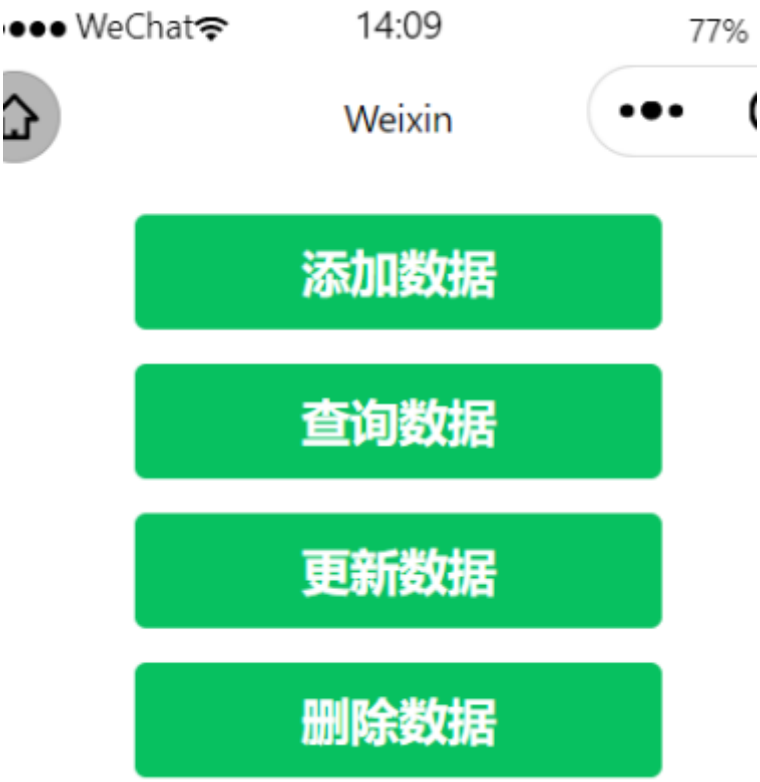
2.初始化数据库

```
wx.cloud.init();
const db = wx.cloud.database();
```



3.设置界面

db.wxml




```
<button type="primary" bind:tap="add">添加数据</button>
<button type="primary" bind:tap="get">查询数据</button>
<button type="primary" bind:tap="update">更新数据</button>
<button type="primary" bind:tap="delete">删除数据</button>
```

db.wxss

```
button{
    margin-top: 30rpx;
}
```

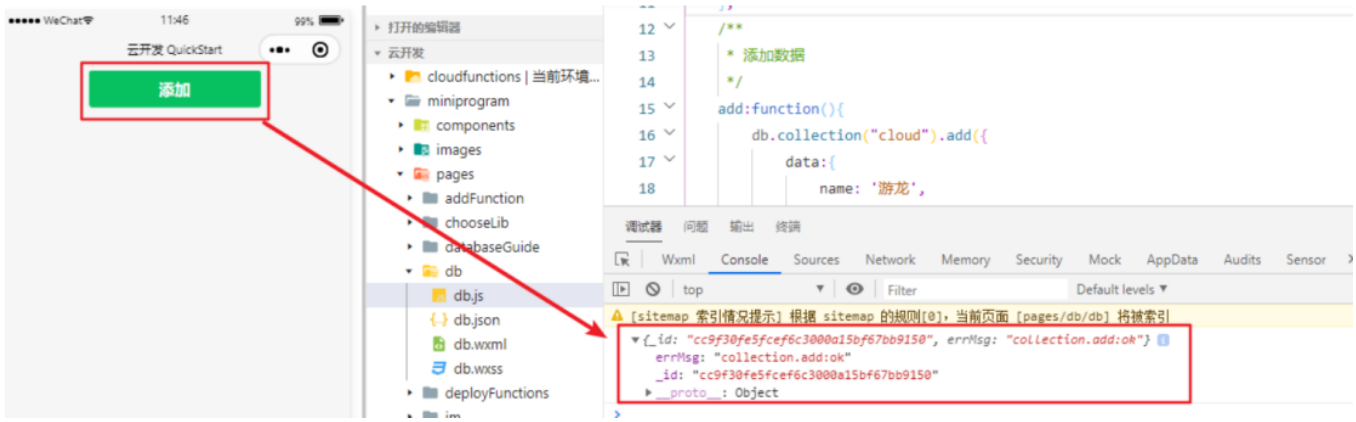
4.添加数据

语法：db.collection(‘集合名称‘).add(“json格式数据”)

```
// 添加数据
add:function(){
    db.collection("gpnudb").add({
        data:{
            name:"游龙",
            age:18,
            sex:'男'
```

```
        }
    })
        .then(res=>{
            console.log(res,'添加数据')
        })
        .catch(err=>{
            console.log(err,'添加数据')
        })
},
```

效果

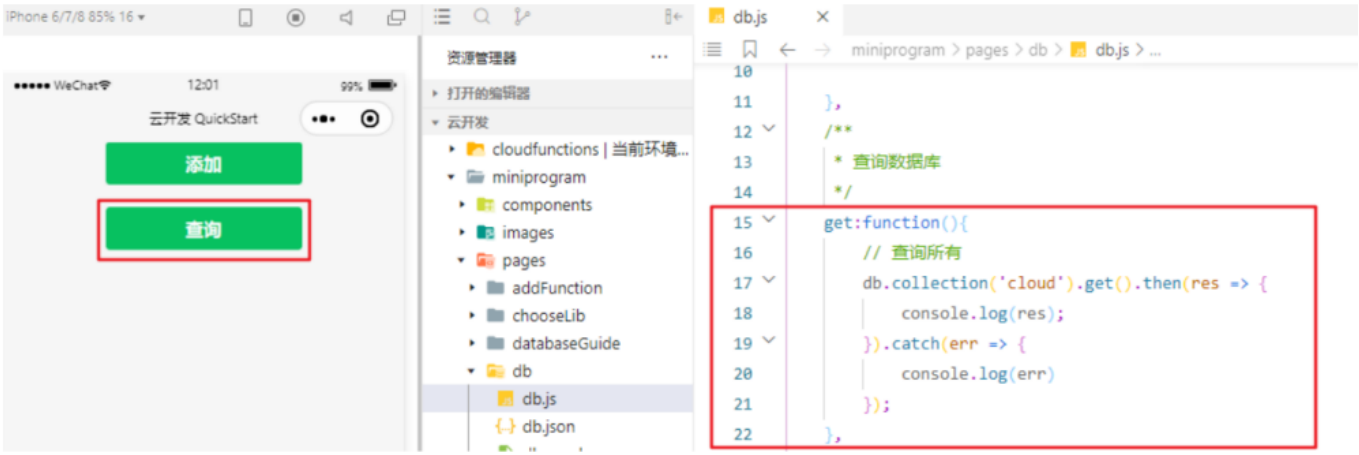


5.查找数据

语法：db.collection(‘集合名称‘).where("查询条件也使用json格式").get()

```
db.collection('cloud').where({name: 'cloud'}).get().then(res => {
    console.log(res, '数据查询')
}).catch(err => {
    console.log(err, '数据查询')
})
```

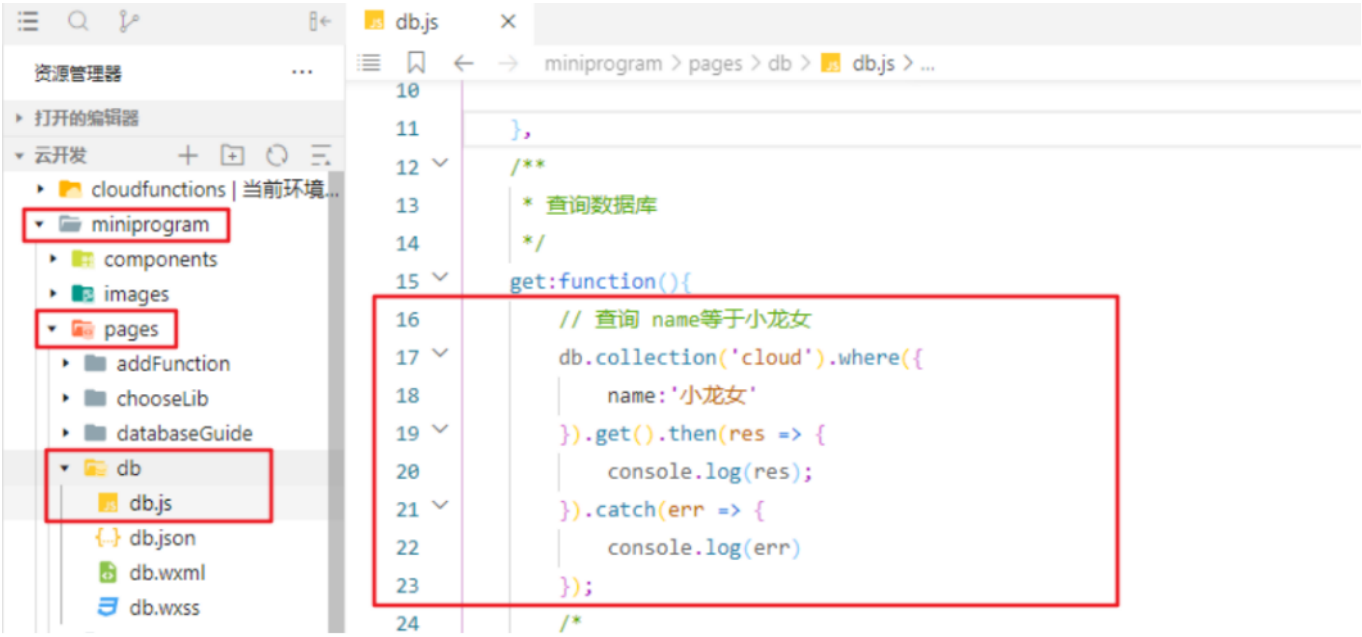
查所有数据



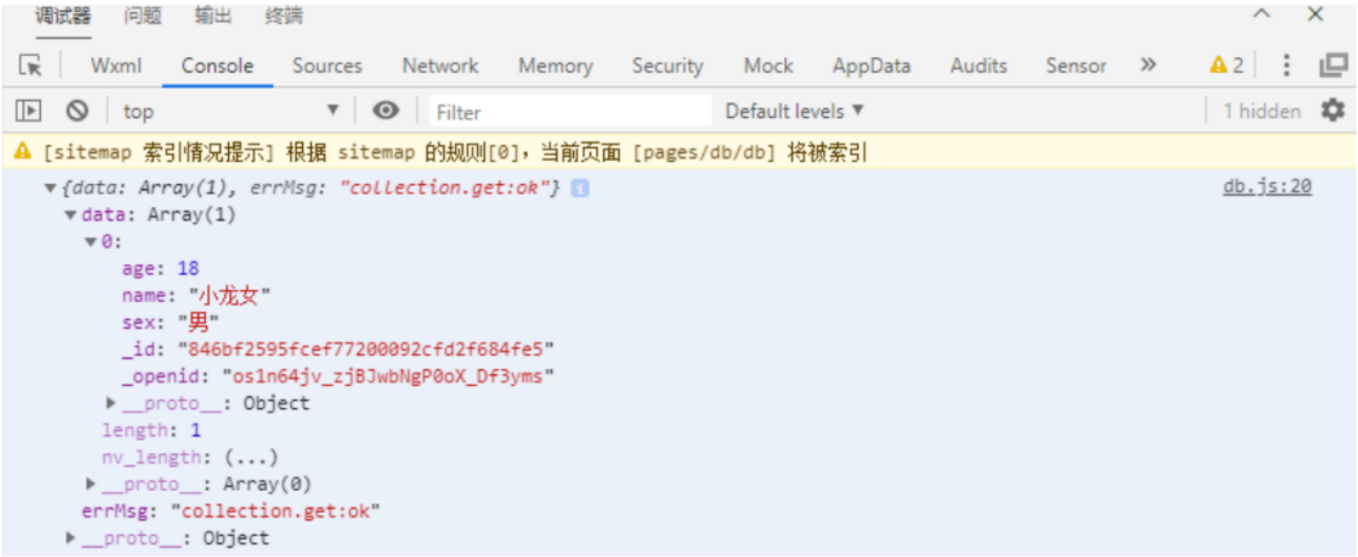
效果

根据指定条件查询

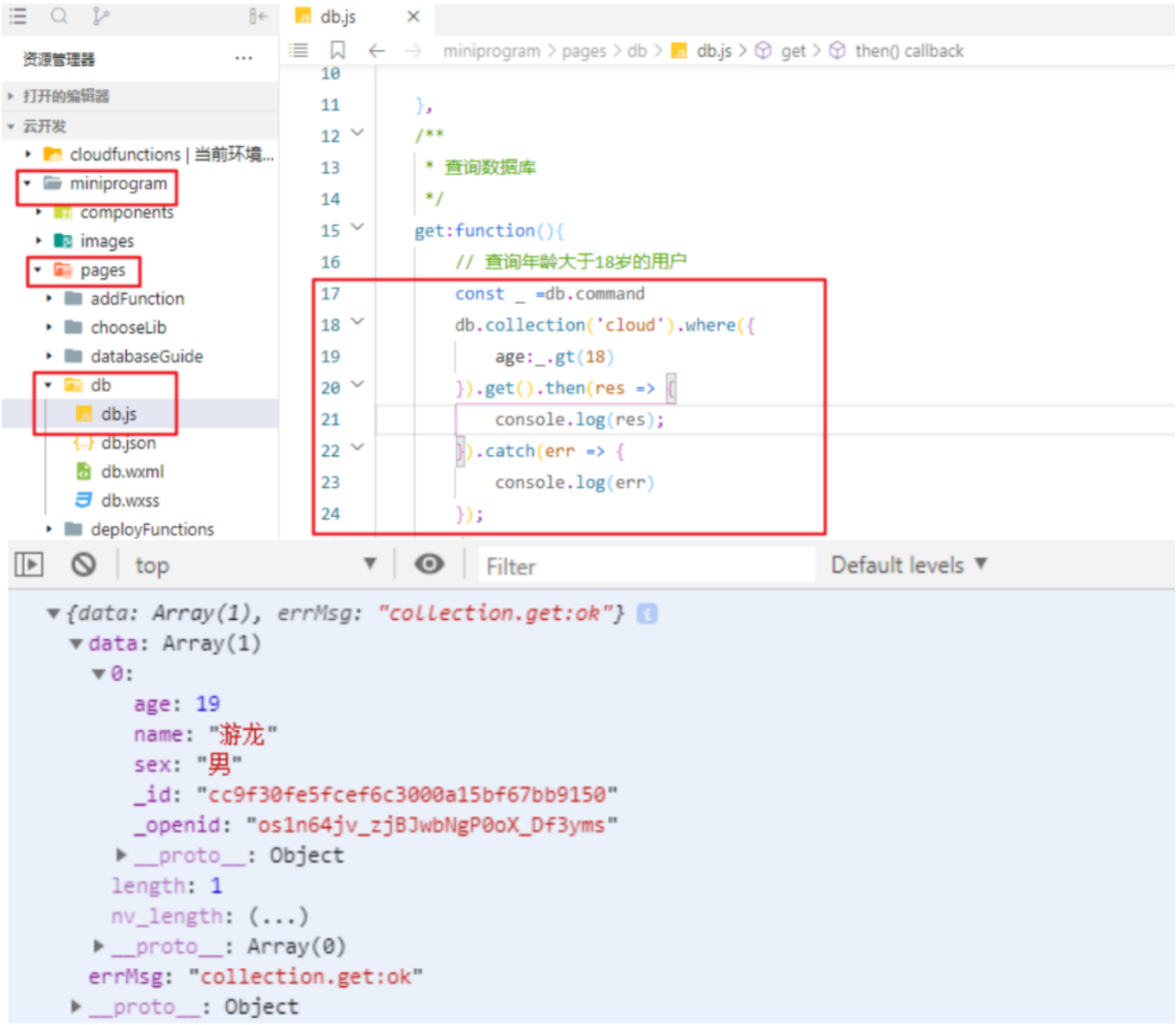
`name` 等于小龙女



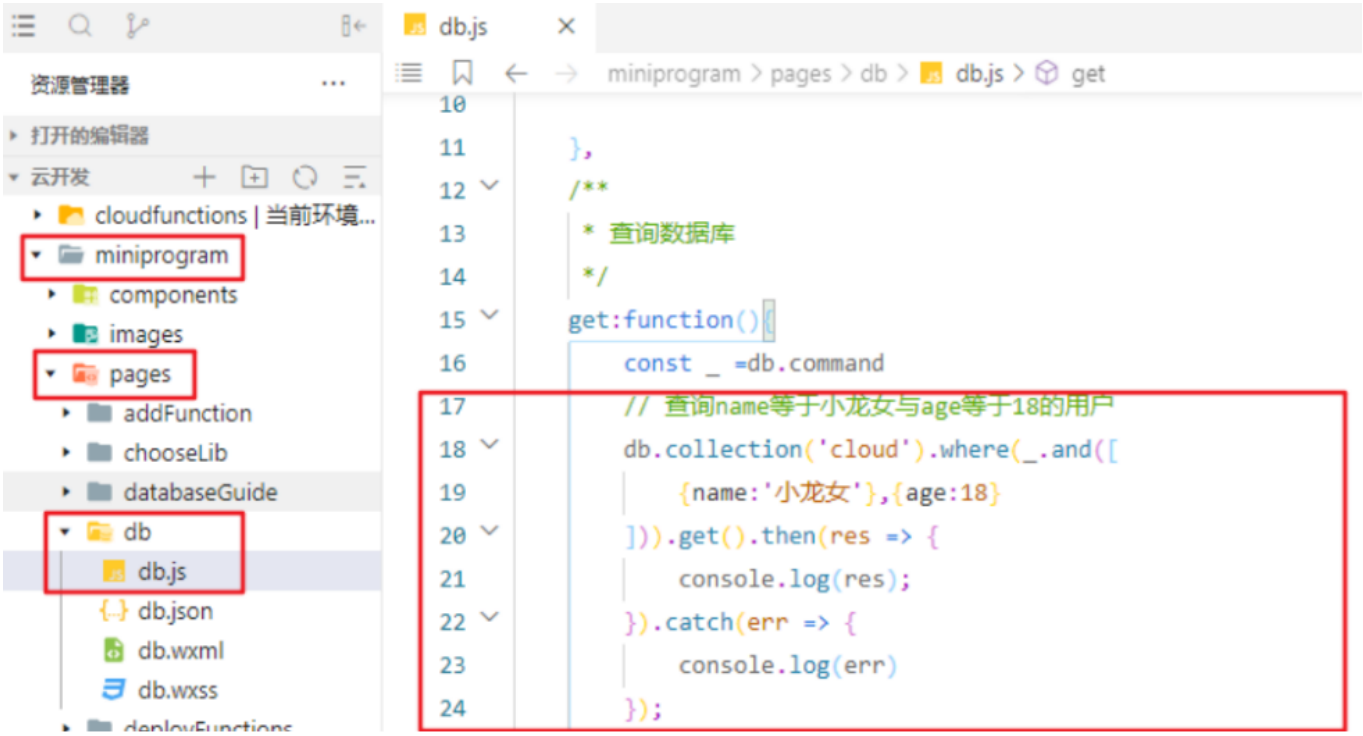
效果



查询年龄大于18岁的用户



查询name等于小龙女与age等于18的用户



6.更新数据

语法：

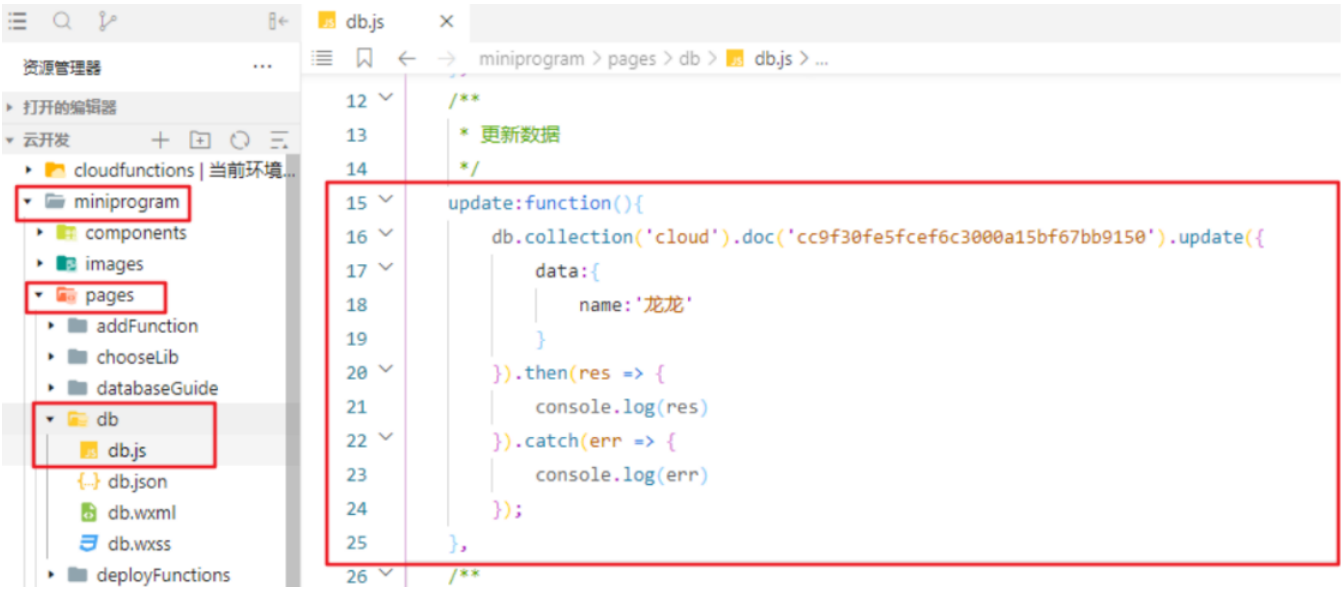
局部更新

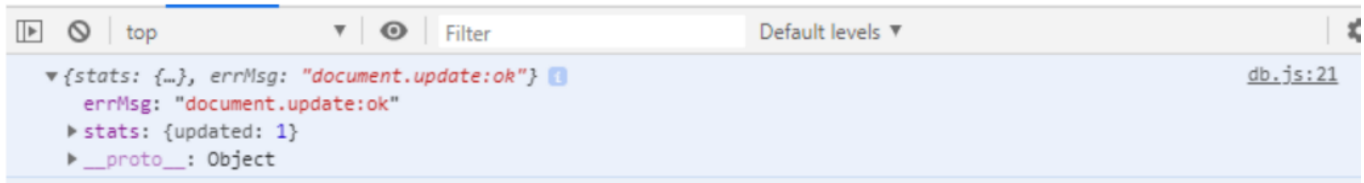
```
db.collection('集合名称').doc(记录id).update(json格式数据)
```

替换更新

```
db.collection('集合名称').doc(记录id).set(json格式数据)
```

```
db.collection('cloud').doc(id).update({
    data: {
        name: 'cloud',
        age:18
    }
}).then(res => {
    console.log(res, '数据更新')
}).catch(err => {
    console.log(err, '数据更新')
})
```





7.删除

语法：db.collection(‘集合名称‘).doc(记录d).remove()

```
db.collection('cloud').doc(id).remove().then(res => {
    console.log(res, '数据删除')
}).catch(err => {
    console.log(err, '数据删除')
})
```