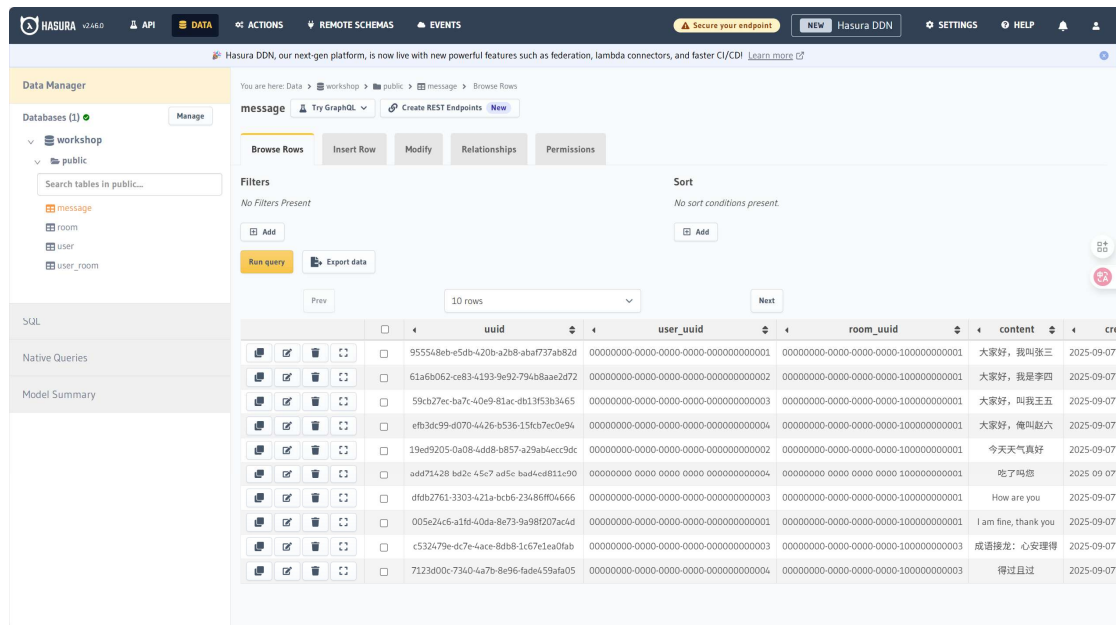


## Hw\_database

2.1:

2.1.1,2.1.2:已完成环境配置， 在 setting 中导入 metadata 并打开数据库



已在 graphql 中实现了如下功能：

在房间内发信息， 以及实时获取房间内的信息

新建房间， 按邀请码查找房间， 查询加入了哪些房间， 加入房间

2.1.3:

前端通过如下语句生成可供调用的函数（虽然目前好像还没有被用）：

```
export type AddMessageMutationFn =
```

```
Apollo.MutationFunction<AddMessageMutation, AddMessageMutationVariables>;
```

被调用之后则发送 post 请求到/register， 被后端处理， 发给 hasura

再被 hasura 经过权限认证后变成 sql 语言， 返回数据。

```
const mutationResult = await graphql.addUser({ username: username, password:
password });
```

```
const payload: userJWTPayload = {
  uuid: mutationResult.insert_user_one?.uuid,
  "https://hasura.io/jwt/claims": {
    "x-hasura-allowed-roles": ["admin", "user"],
```

```

"x-hasura-default-role": "user",

},

};

const token = jwt.sign(payload, process.env.JWT_SECRET!, {

  expiresIn: "24h",

});

```

2.2:

分别按要求向三个 graphql 中插入如下语句，已在 hasura api 中验证之：

The screenshot shows the GraphQL IDE interface. The query editor on the left contains the following mutation:

```

1 mutation deleteUser($uuid: uuid!) {
2   delete_user(where: {uuid: {_eq: $uuid}}) {
3     affected_rows
4     returning {
5       username
6       uuid
7     }
8   }
9 }
10

```

The query variables section at the bottom left shows:

```

1 {
2   "uuid": "00000000-0000-0000-0000-000000000004"
3 }

```

The response area on the right displays the JSON result:

```

{
  "data": {
    "delete_user": {
      "affected_rows": 1,
      "returning": [
        {
          "username": "赵六",
          "uuid": "00000000-0000-0000-0000-000000000004"
        }
      ]
    }
  }
}

```

At the bottom right, the status bar indicates: RESPONSE TIME 339 ms, RESPONSE SIZE 127 bytes.

The screenshot shows the GraphQL IDE interface. The query editor on the left contains the following mutation:

```

1 mutation quitRoom($room_uuid: uuid = "", $user_uuid: uuid = "") {
2   delete_user_room(where: {room_uuid: {_eq: $room_uuid}, user_uuid: {_eq: $user_uuid}}) {
3     returning {
4       room_uuid
5       user_uuid
6     }
7     affected_rows
8   }
9 }
10

```

The query variables section at the bottom left shows:

```

1 {
2   "user_uuid": "00000000-0000-0000-0000-000000000003",
3   "room_uuid": "00000000-0000-0000-0000-100000000001"
4 }

```

The response area on the right displays the JSON result:

```

{
  "data": {
    "delete_user_room": {
      "returning": [
        {
          "room_uuid": "00000000-0000-0000-0000-100000000001",
          "user_uuid": "00000000-0000-0000-0000-000000000003"
        }
      ],
      "affected_rows": 1
    }
  }
}

```

At the bottom right, the status bar indicates: RESPONSE TIME 112 ms, RESPONSE SIZE 172 bytes.

GraphQL

Prettify

History

Explorer

Cache

Code Exporter

REST

Derive action

Analyze

< Docs

1\*  
2\*  
3  
4  
5  
6  
7  
8  
9  
10  
11\*  
12\*  
13  
14  
15  
16

```
mutation quitRoom($room_uid: uuid = "", $user_uid: uuid = "") {  
  delete_user_room(where: {room_uid: {_eq: $room_uid}, user_uid: {_eq: $user_  
    returning {  
      room_uid  
      user_uid  
    }  
  }  
  affected_rows  
}  
  
query getMessageByUser($user_uid: uuid = "") {  
  message(where: {user_uid: {_eq: $user_uid}}) {  
    user_uid  
    room_uid  
    content  
  }  
}
```

QUERY VARIABLES

1 {  
2 "user\_uid": "00000000-0000-0000-0000-000000000003"  
3 }

"data": {  
 "message": [  
 {  
 "user\_uid": "00000000-0000-0000-0000-000000000003",  
 "room\_uid": "00000000-0000-0000-0000-100000000001",  
 "content": "大家好，叫我王五"  
 },  
 {  
 "user\_uid": "00000000-0000-0000-0000-000000000003",  
 "room\_uid": "00000000-0000-0000-0000-0000-0000-100000000001",  
 "content": "How are you"  
 },  
 {  
 "user\_uid": "00000000-0000-0000-0000-000000000003",  
 "room\_uid": "00000000-0000-0000-0000-0000-0000-100000000003",  
 "content": "成语接龙：心安理得"  
 },  
 {  
 "user\_uid": "00000000-0000-0000-0000-000000000003",  
 "room\_uid": "00000000-0000-0000-0000-0000-0000-100000000003",  
 "content": "忘恩负义"  
 }  
 ]  
}

RESPONSE TIME 315 ms

RESPONSE SIZE 891 bytes