**软件设计**

1. **设计概述**

该系统可以实现对gaze/fixation/event/position等数据进行分析。其分析角度可以针对不同的dynamic obj和scene。并可以设置objective任务，来判断session的完成程度。同时，后台还会对每个session在执行过程中的场景进行复原。

本系统设计为使用三个逻辑模块实现基本功能。其中包括 1）server模块，用于实现与VR设备和客户端的数据同步与数据分析 2）客户端模块，用于实验人员操作及查看分析数据 3）VR-unity程序模块，用于收集实验数据。同时，为了适应不同的场景需求，提供了3中不同的部署方式。

下面将通过关键词解释、系统架构、物理部署、模块功能详情、VR设备与服务器端交互详解、数据分析内容详解、数据结构设计几部分对系统进行详细阐述。另外，对于系统的使用效果、使用说明文档、接口文档、测试文档等内容将放在其他文件中详细介绍。

1. **关键词解释**

**organization:**

组织代表整套系统的根节点，一个组织包括多个user/project/participant。在使用系统时，需要先创建组织，之后在组织中创建用户才可以使用本系统。

**user:**

用户代表一个组织下的可操作账户。是登陆客户端用户操作的基本单位。每个组织下的不同用户可访问所有项目，例如userA创建的项目userB也可以访问。（[TODO]用户权限控制）创建用户需要知道组织的账号和密码。

**project:**

每个组织下有多个项目，每个项目可以是一个单独的实验。

**PRO\_KEY:**

用于标识每个项目的唯一KEY，主要作用是将PRO\_KEY粘贴到VR开发时的资源配置文件中，这样就可以通过该key区分不同的项目。将新创建的VR项目的数据上传到相应的project中去。

**participant:**

参与者代表参与实验的人。每个参与者有1个或n个session。开发者可以在程序中设置c3d.participant.id来区分不同的参与者，开发者也可以设置c3d.participant.name，但是作为唯一标识。同时该信息也可以由参与者在实验之前手动在VR设备中输入。

**scene/scene\_version:**

场景与unity中的场景概念相同，每个场景都有1-n个版本。可以在unity中上传更新场景的版本。也可以重新上传新场景。

**dynamic\_obj:**

动态物体代表程序中需要监控gaze/fixation/event的物体。我们需要将这些物体设置dynamic\_obj脚本。

**category:**

代表多种group的配置方案，每个group代表多个动态对象的集和。

**objectives/objectives\_version:**

实验人员设定的完成任务的步骤。每一个objectives都有1-n个objectives\_version。每个objectives必须针对一个scene的scene\_version。通过设置objectives的步骤，实验人员可以把控每个session完成的进度。

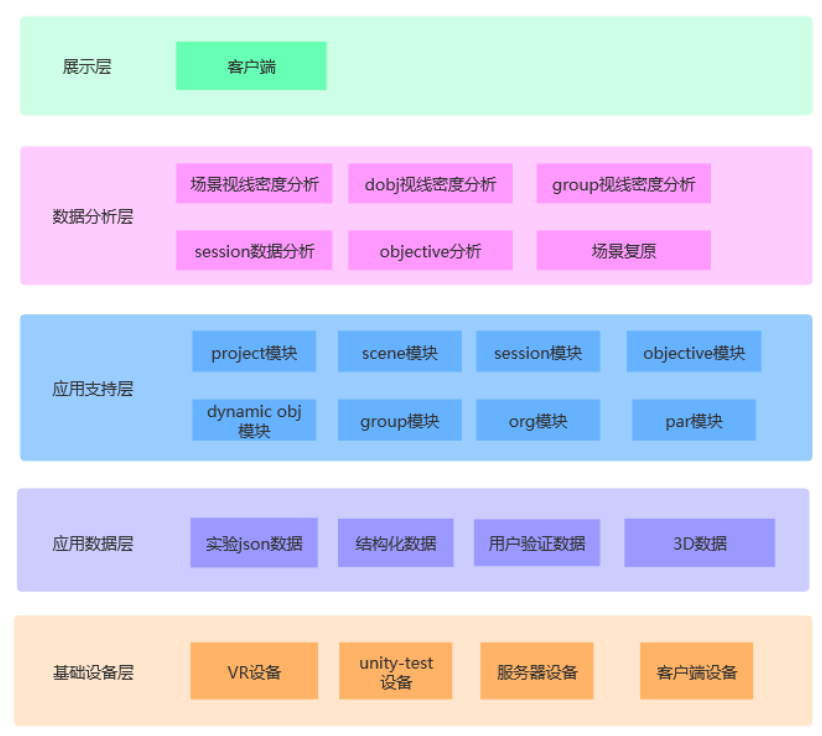
**session:**

代表参与者进行的一次实验。会从VR设备上手机Gaze/Fixation/Event/Dynamic obj/Sensor数据。([TODO]Sensor)

下面举一个例子说明上述关键词在该系统中是如何使用的。

我们对VR键盘输入进行分析。首先需要创建一个组织(organization)或使用之前进行实验的组织。这个组织可以是一个大学，公司...随后，我们为该组织创建一个用户(user)，或使用之前注册的用户。为了进行VR KEYBOAR的实验，我们需要为这个实验建立一个项目(project)。随后，当我们开发相应的unity程序时，为了标识该程序属于这个项目，我们就需要将PRO\_KEY放入资源文件中[具体使用方法在使用文档中详细介绍]。我们为了分析QWERTY键盘的视线数据，就需要我们将这些键位设置为动态物体(Dynamic obj)。然后我们会将场景(scene)和dynamic obj上传到平台上。这时系统就会自动创建场景和一个场景版本。当编写完unity程序并上传到系统后。我们就可以开始进行用户实验了。其中每次不同参与者(participant)进行实验时，我们都为其设定不同的c3d.participant.id当实验完成后，每次进行的实验都会生成一条完整的(session)信息。最后，当我们进行完实验后，需要对数据进行分析。这时我们可以在各分析区域查看可视化的效果。也可以设定用户的完成步骤(objectives)，来查看完成整体任务的情况。最后，我们为了分析出于一个区域内的多个键位的共同数据分析结果。我们可以设置(group)进行聚合。

1. **系统架构**



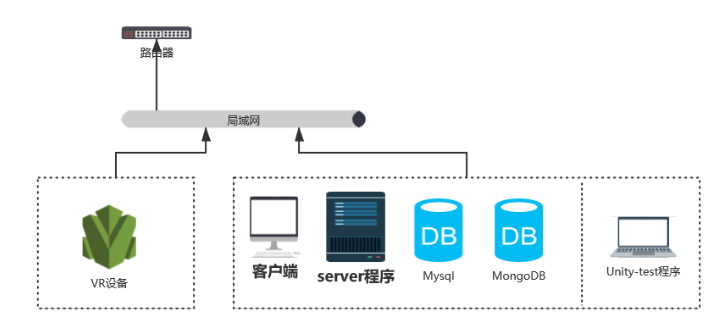
1. **物理部署**

本系统由3部分逻辑模块组成，分别是（1）服务器端程序，包括以统部署的mysql和mongodb，用于接收VR设备的数据，以及向客户端提供分析内容（2）客户端，用户使用，来查看分析内容（3）unity-VR插件，用于VR设备与服务器端同步场景和dynamic obj数据，以及上传session数据。

这些逻辑模块可以一不同的物理形式部署。下面将介绍三种适用于不同场景的部署形式，同时在不同逻辑模块设备所处的物理设备上需要符合其硬件与软件环境。具体的部署和安装方式将在使用文档中详细介绍。

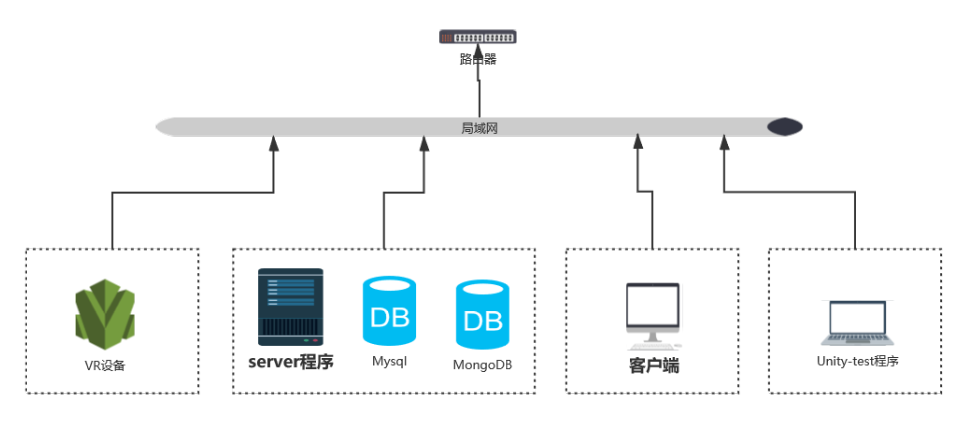
1. 完全单机部署模式

该部署形式是将服务器端程序、客户端程序全部部署到一台设备上。参与者在与该设备出于同一局域网内使用VR设备。使用者可以在该设备上打开客户端查看后台分析内容。同时，也可以在UNITY程序开发时，将unity安装在这台设备上，这样就可以实现本机的开发、测试、运行与分析。该部署形式适用于，测试与开发阶段。



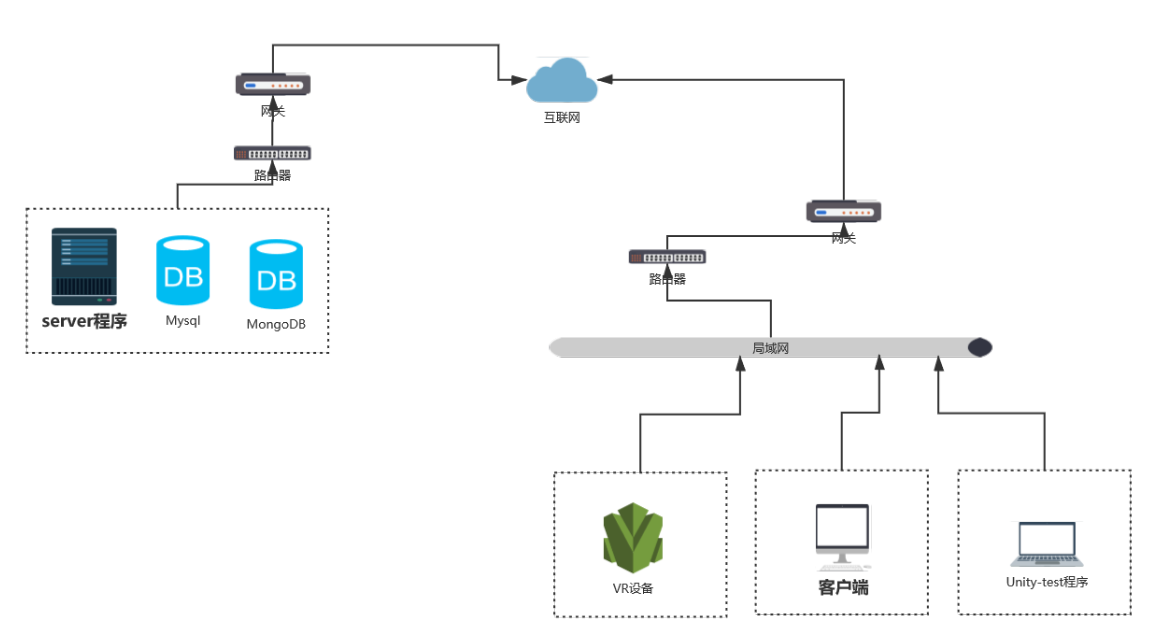
1. 局域网部署模式 **[\*]**

该部署形式是将服务器端程序和客户端程序分别部署到局域网中的不同设备上。可以作为正式的实验平台使用。可以由1台服务器和多台客户端及VR设备组成。在兼顾了灵活性的同时保证了安全性。



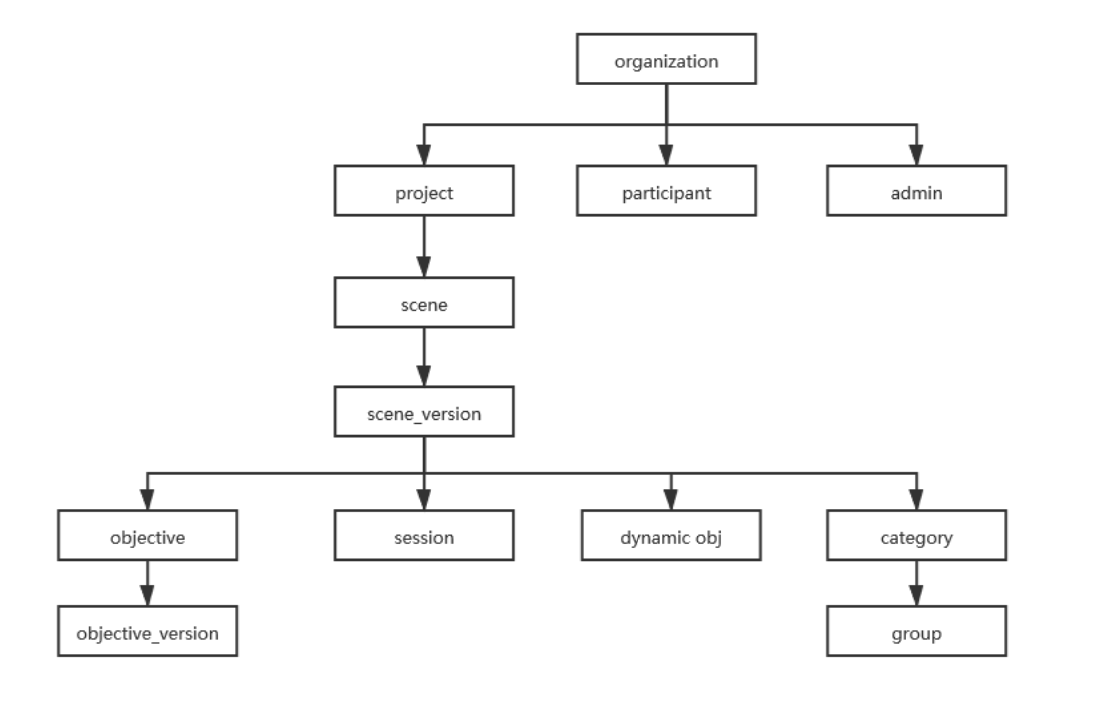
1. 公网部署模式

该部署方式是将服务器端程序部署于公网可访问的地址上，客户端和VR设备只要可以接入互联网就可以使用本系统。(同原版)



1. **模块功能详情**

从系统整体的角度看，我们将系统按组织的层级和包含关系表示为以下模块结构图。



从开发和部署的角度，又将本系统分为（1）服务器端（2）客户端（3）unity插件三部分，下面分别针对服务器端和客户端进行模块定义。

1. 服务器端

在服务端我们将分为与VR设备数据对接的模块类和与客户端对接的模块类。

VR数据模块类：

**scene\_device模块：**用于处理VR设备端与场景上传、场景信息获取、场景版本更新相关的数据

**Function:**

1. get\_scene\_setting 获取某个场景和该场景下各版本的基本信息
2. upload\_screen\_shot 上传场景封面（如果之前有封面会更新）
3. upload\_new\_scene 上传新场景（首次上传）会创建scene/scene\_version资源文件夹、scene列、scene\_version
4. upload\_new\_version\_scene上传新版本的场景（非首次上传）会创建scene\_version资源文件夹、scene\_version

**session\_device模块：**用于接收用户实验中的session数据

**Function:**

1. get\_gaze\_data 服务器端接收VR中session的Gaze数据（将json数据插入mongodb。如果是part1就插入或更新现有的session列，把基本信息+gaze数据id\_json插入。如果不是part1就更新gaze数据id\_json）
2. get\_fixation\_data 服务器端接收VR中session的Fixation数据（将json数据插入mongodb。如果session在接收本数据之前不存在就先暂时创建一条）
3. get\_event\_data 服务器端接收VR中session的Event数据（将json数据插入mongodb。如果session在接收本数据之前不存在就先暂时创建一条）
4. get\_dynamicobj\_data 服务器端接收VR中session的Dynamic数据（将json数据插入mongodb。如果session在接收本数据之前不存在就先暂时创建一条）

**dynamic\_obj\_device模块：**用于处理VR设备端与dynamic obj上传、更新、信息获取相关的数据

**Function:**

1. get\_dynamic\_object\_manifest 获取dynamic obj的基本信息清单 [TODO]
2. upload\_dynamic\_object 上传dynamic obj（判断是否在上传该dobj之前已经在数据库中插入一列，如果已经有了就更新）
3. upload\_dynamic\_object\_manifest 上传dynamic obj的基本信息清单（判断数据库中是否已经有一列，如果有就更新，没有就创建）

注:

[upload\_dynamic\_object\_manifest/upload\_dynamic\_object都会插入数据，看先调用哪个]

客户端对接模块类：

**admin模块：**处理与系统组织和用户相关的内容

**Function:**

1. organization\_regist 组织注册
2. organization\_list 查询组织列表
3. organization\_detail 查询组织基本信息
4. organization\_delete 删除组织(不具体实现)
5. user\_regist 注册用户
6. user\_org\_list 查询某个组织下的用户
7. user\_detail 查看用户详情
8. user\_delete 删除某个组织下的用户(不具体实现)
9. user\_edit 用户内容更改(不具体实现)
10. user\_login 用户登陆

**category模块：**处理与dynamic obj组成的category相关的内容

**Function:**

1. set\_obj\_category 创建category和group
2. set\_dobj\_to\_group 向group中设置dynamic obj
3. get\_obj\_category\_list 获取category和group的详情信息

**dynamic\_obj模块：**处理与dynamic obj查看、分析相关的内容

**Function:**

1. dynamic\_obj\_list dynamic obj列表
2. dynamic\_obj\_detail 某个dynamic obj详情

**objective模块：**用于设置objective执行步骤，并查看分析数据

**Function:**

1. objective\_create 创建objective（创建objective同时创建新版本）
2. objective\_create\_version 创建新版本objective
3. objective\_detail objective的各版本的基本步骤
4. objective\_list objective列表
5. objective\_delete 删除某个objective(不具体实现)
6. objective\_version\_stepresult 获取某个版本objective的各步结果
7. objective\_session\_result 获取某个session在其scene\_version下的objective的各步结果

**participant模块：**与本组织进行session的参与者相关到的内容（其中的参与者标识需要在VR设备或其程序中设置）

**Function:**

1. participant\_list 获取参与者列表
2. participant\_detail 获取某个参与者基本信息及他的session基本信息

**project模块：**与本组织中项目相关的操作

**Function:**

1. project\_scene\_detail 获取某个项目的基本详情及他的scene和scene\_version
2. project\_list 获取项目列表
3. project\_key 获取PRO\_KEY
4. project\_create 创建项目
5. project\_delete 删除某个组织下的用户(不具体实现)
6. project\_edit 用户内容更改(不具体实现)

**scene模块：**与本项目中场景和场景版本相关的操作

**Function:**

1. scene\_detail 获取场景的基本信息及其各版本的基本信息
2. scene\_delete 删除某个scene(不具体实现)
3. scene\_edit scene更改(不具体实现)

**session模块：**与参与者进行的session相关的操作，包括列表、数据分析、现场复原

**Function:**

1. session\_list 获取某个场景版本下的session列表
2. session\_metadata 获取某个session的基本信息
3. session\_single\_queries 查询单个session
4. session\_queries 查询session
5. session\_gaze\_data 获取某个session的gaze数据
6. session\_fixation\_data 获取某个session的fixation数据
7. session\_event\_data 获取某个session的event数据
8. session\_dynamicobj\_data 获取某个session的dynamicobj数据
9. session\_slicer\_field\_queries 获取事件/属性，用于筛选[待完善]
10. session\_slicer\_metric\_object\_queries 获取某个dynamic obj的计算指标
11. session\_slicer\_metric\_group\_queries 获取某个group的计算指标
12. session\_slicer\_queries 查询session，用于客户端analyze模块
13. dynamic\_obj\_cube\_agg\_queries 获取cube的分析数据
14. 客户端

**admin模块：**处理客户端登陆权限等功能

1. 登陆页面 admin.login
2. 用户详情页面 admin.main

**participant模块：**展示参与者极其session信息

1. 参与者列表页面 participant.participant\_list
2. 参与者详情页面 participant.participant\_detail

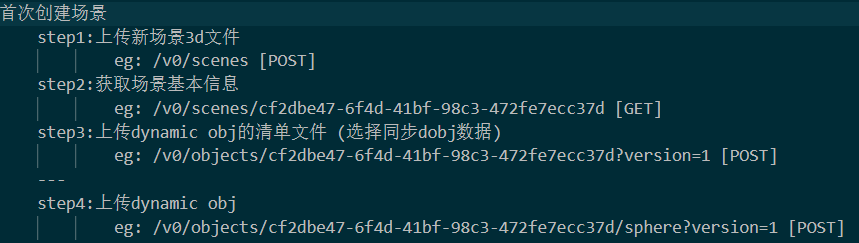
**project模块：**与该组织中的项目极其下场景相关的操作

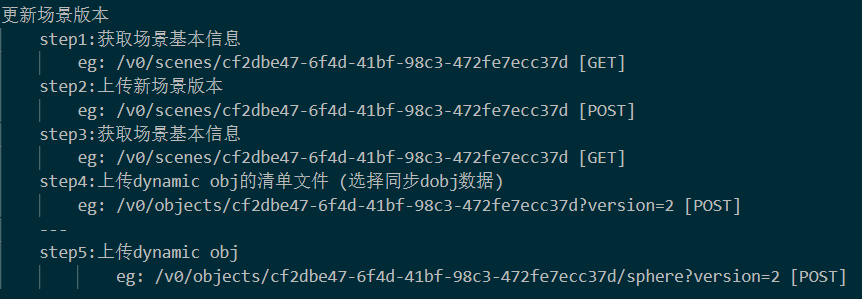
1. 项目列表页面 project.project\_list
2. 场景列表页面 project.scene\_list
3. objective列表页面 project.objectives
4. 创建objective页面 project.objectives\_create  
    (5) objective分析页面 project.objectives\_analyze
5. 分析页面 project.analyze

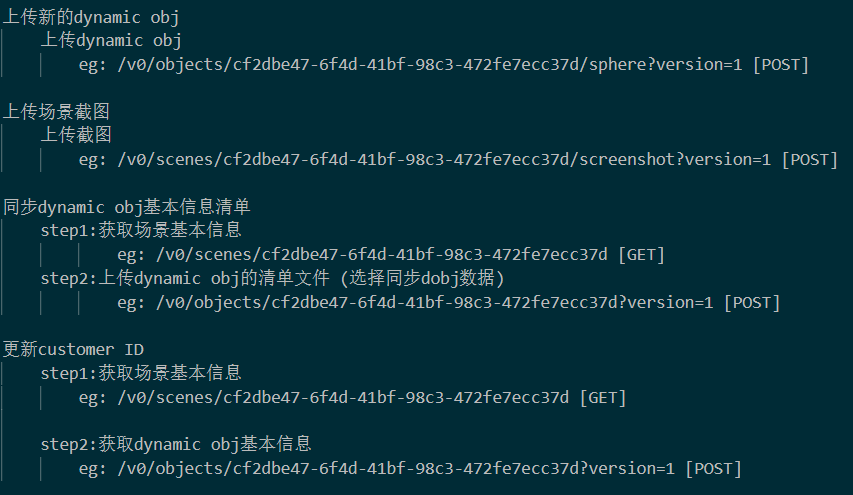
**scene模块：**与场景及场景版本相关的内容

1. 场景基本详情页面 scene.insights
2. 场景浏览页面 scene.scene\_viewer
3. session列表页面 scene.allsessions
4. session详情页面 scene.session\_detail
5. session场景复原页面 scene.session\_show
6. dynamic obj列表页面 scene.objectlist
7. dynamic obj分析页面 scene.dobj\_analyze
8. group页面 scene.group
9. group分析页面 scene.group\_analyze
10. **VR设备与服务器端交互详解**

本节介绍，VR设备常用操作与server端的具体数据交互。







1. **数据分析内容详解**

本节介绍系统内的主要数据分析内容。

**scene viewer cube分析：**

用于分析不同数据出现在场景内的密度。其中包括 worldFixation/worldGaze/worldPosition/worldEvent。其中worldFixation/worldGaze是筛选那些将视线定位到场景中的数据（剔除投射在dynamic obj上的数据和sky数据）。在程序中通过输入的筛选条件检索出全部数据，并分析数据在三维中xyz的最大和最小值，并按interval切分，计算不同的区域的热度。最后在客户端中以cube的形式进行显示，对每个cube内有不同数量的点按不同颜色来表示。这里由两种可展示的思路（1）手动分段设定不同点数的颜色深度（2）将数据中最多的点数设为1，最少为0，自动计算中间区域颜色。本系统默认使用第一种方式。

**dynamic obj cube分析：**

用于分析不同与该dynamic obj关联数据的密度。其中包括 objectFixation/objectGaze。同上

**dynamic obj metric分析：**

分析dynamic obj的众多与gaze和fixation有关的指标。

**group metric分析：**

分析group的众多与gaze和fixation有关的指标。

**objective分析：**

设定分步任务，分析每个session能成功完成的step。同时分析，全部session完成所有step的比例等。本系统将任务分为event/gaze/fixation三种。在程序中分别对每一个session的每一步分别进行判断，再将信息进行聚合。

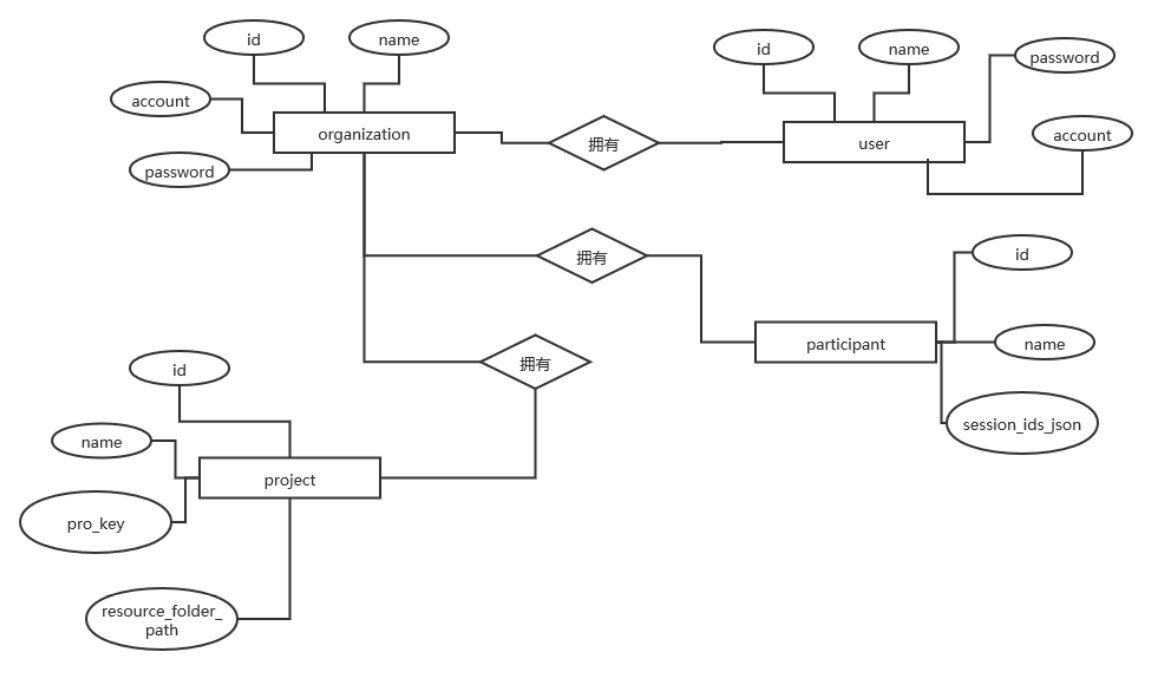
**session detail分析：**

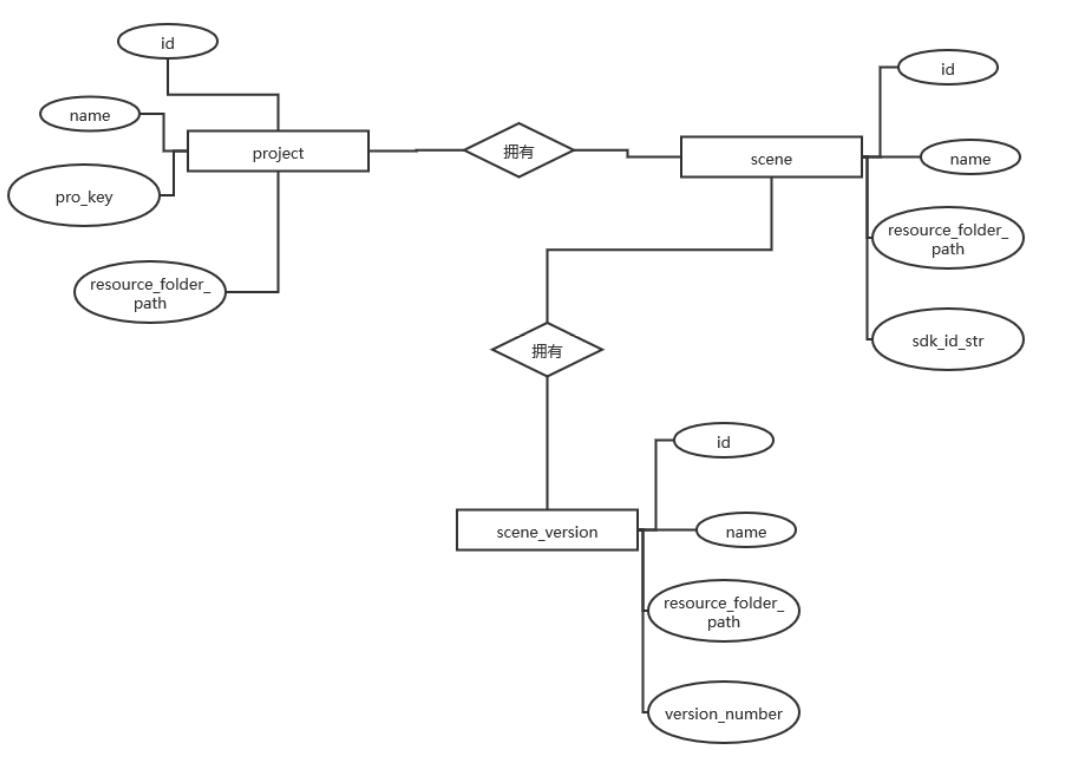
分析session的详细内容，包括展示基本信息，objective信息等。

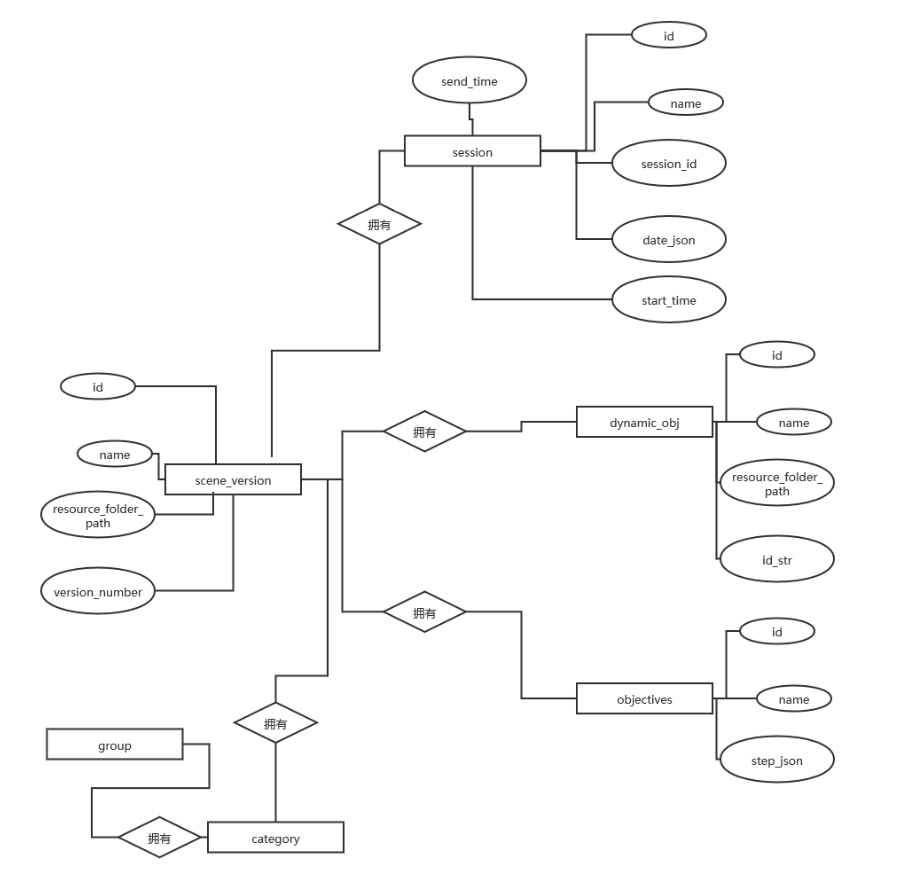
**analyze模块分析：**

以各种筛选条件计算以天为单位的数据，其中数据类型包括，session数量，event出现的次数，session length的最大长度，最小长度，平均长度。

1. **ER图**

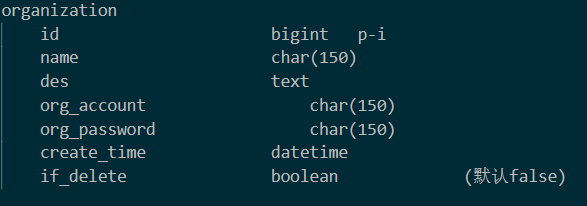


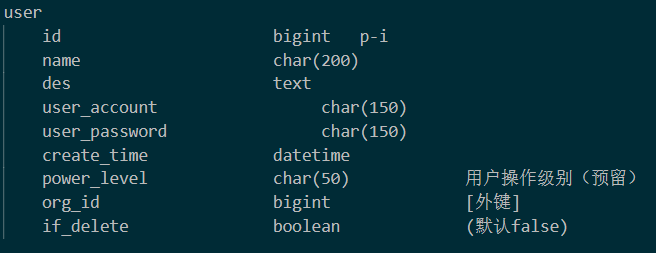


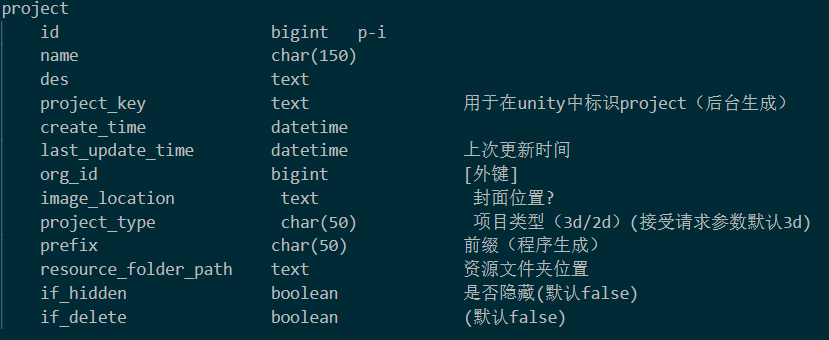


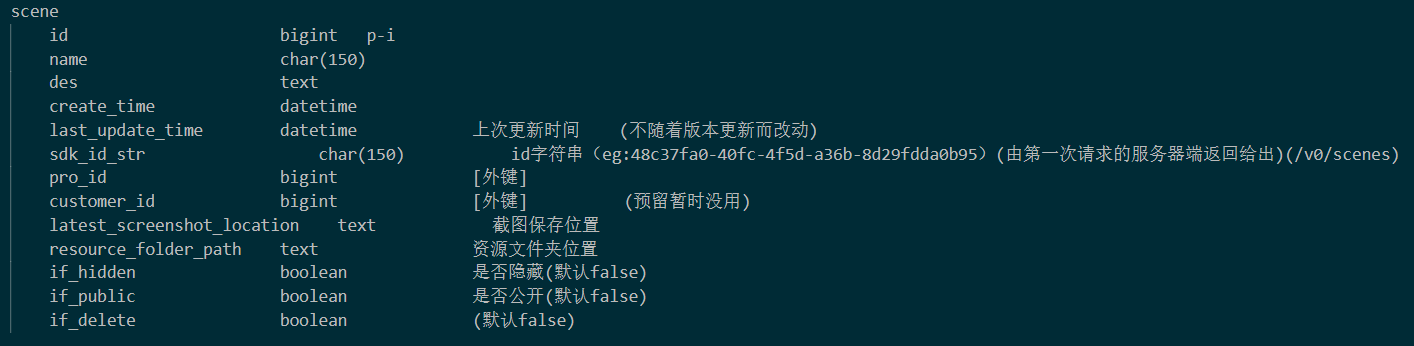
1. **数据结构设计**

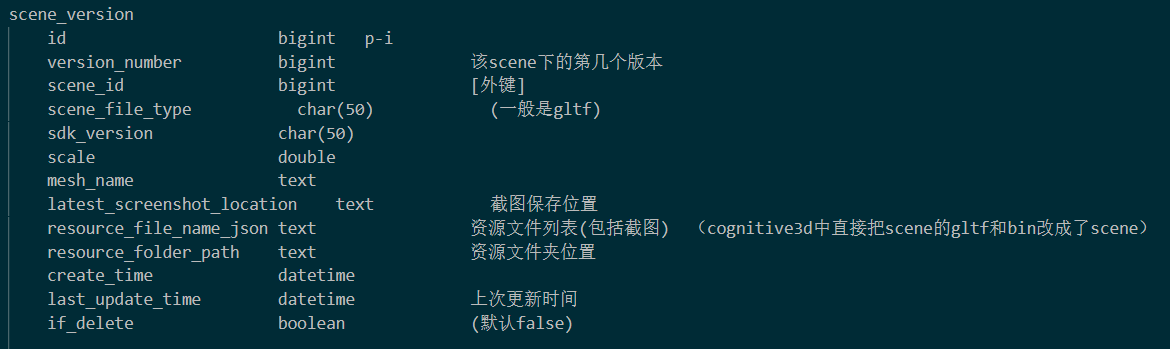
**Mysql：**

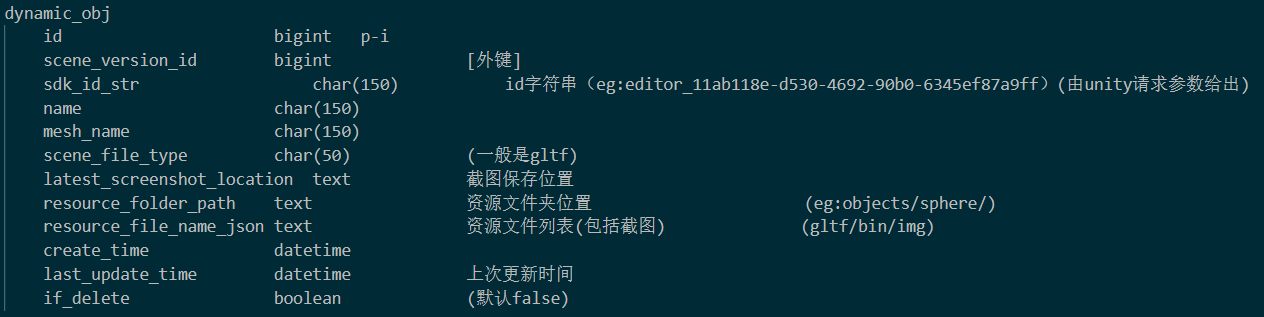


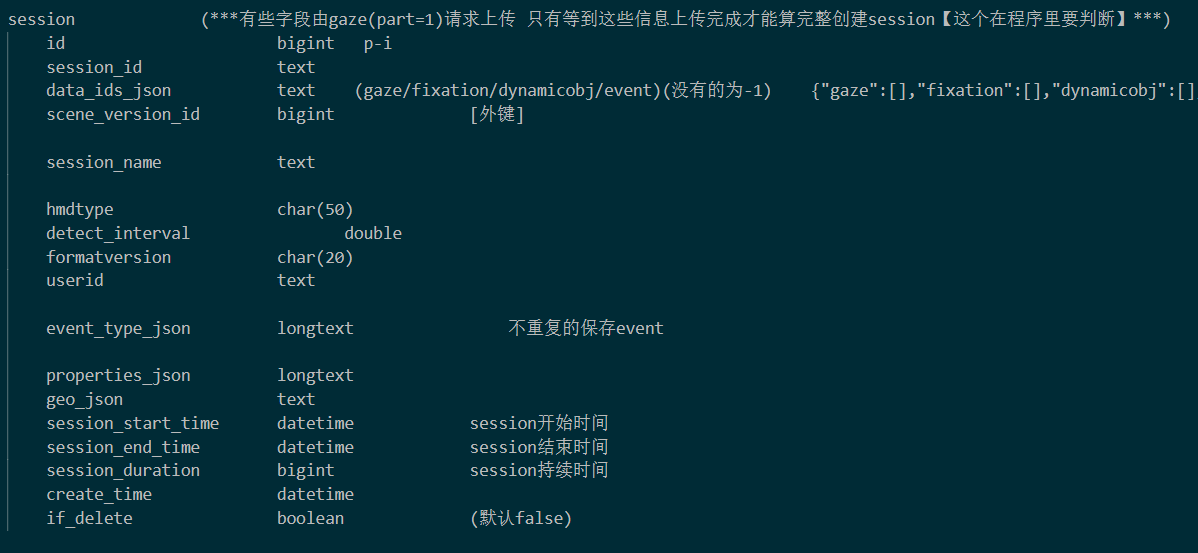


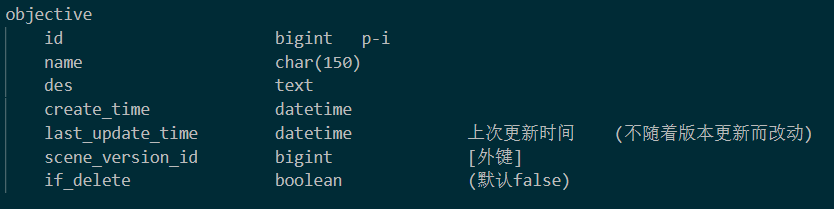


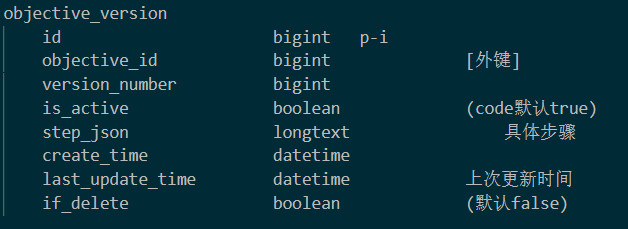


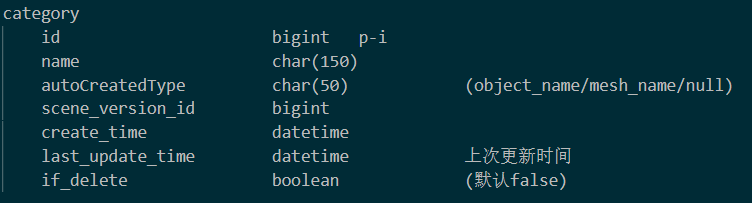


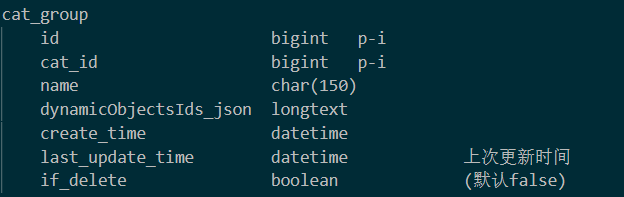


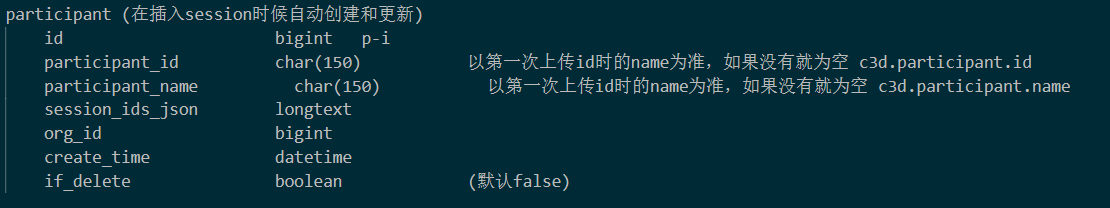












**Mongodb：** dynamicobj event fixation gaze