数据库设计说明文档

设计场景

基于 DMC（Data Management Center）进行数据库的设计与建立，这是一个数据管理中心，其中包括的组成部件包括：LoadBalancer、Rack、Server、VirtualMachine、Application、Category、DatabaseServer、User。

实体介绍

1. LoadBalancer（负载均衡器）：这是一种网络设备或服务，用于在多个计算资源（如服务器、虚拟机或容器）之间分配和管理网络流量，以实现高可用性和性能优化。负载均衡器在计算资源之间平衡负载，确保每个资源都能够有效地处理请求，并将流量分配到最佳的资源上，从而提供更好的用户体验。它会搭载在机架上面。

需要进行说明的属性：LBid、名称、IP 地址、端口、采用的负载分发算法、是否启用SSL加速、是否启用会话保持设置，装载在哪个机架上 Rid。

2. Rack（机架）：通常用于描述数据中心或服务器机房中的一个物理结构单元。它是一种用于组织和安装计算设备、网络设备和存储设备的框架或架构。
   需要进行说明的属性：Rid、名称、所在的物理位置、它的容量。
3. Server（服务器）：是一种计算机硬件或软件系统，旨在提供网络服务、数据存储、资源共享和应用程序运行等功能。它通常是在客户端-服务器模型中作为服务提供方而存在，以响应客户端的请求并提供所需的服务。
   需要进行说明的属性：Sid、名称、IP 地址、采用的操作系统、采用的处理器类型、内存容量、装载在哪个机架上 Rid。
4. Application（应用）：是指为实现特定功能或任务而开发的计算机程序。应用程序是在操作系统上运行的软件，可以提供各种功能和服务，包括处理数据、执行任务、展示内容、进行通信等。
   需要进行说明的属性：Aid、名称、版本、开发者信息、对该 application 的简述、占用的空间大小、声明其采用了什么数据库服务器DBSid。
5. Category（类别）：这是对 application 的类别属性专门建立的实体，一个 application 可以属于多个类别，同时多个 application 也可以属于同一个类别。
   需要进行说明的属性：Cid、名称、对类别的说明。
6. DatabaseServer（数据库服务器）：一种专门用于存储和管理数据的服务器。它提供了数据存储、数据访问和数据管理的功能，用于支持应用程序、网站和其他系统对数据的处理和操作。
   需要进行说明的属性：DBSid、名称、IP 地址、DBMS 类型、版本、处理器类型、内存容量。
7. User（用户）：这是使用 application 的用户。
   需要进行说明的属性：Uid、性别、年龄、邮箱。

关系介绍：

首先对必要的一对多关系进行说明：

LoadBalancer 架构在 Rack 上，在常见的情况下，这是一个一对多关系（LoadBalancer 是多端）。

Server 架构在 Rack 上，在常见的情况下，这是一个一对多关系（Server 是多端）。

VirtualMachine 架构在 Server 上，这是一个一对多关系（VirtualMachine 是多端）。

Application 使用 DBS，这是一个一对多关系（Application 是多端）。

然后对必要的多对多关系进行说明：

1. runon 关系：表示 application 在 virtualmachine 上运行的关系，这是一个多对多关系，一个 applicaton 可以在多个虚拟机上运行，同时一个虚拟机上也可以运行多个 application。

2. belong 关系：表示一个 application 属于 category 的关系，一个 application 可以属于多个类别，同时多个 application 也可以属于同一个类别。也是一个多对多关系。
3. Replicia 关系：表示一个 DBS 是另一个 DBS 的复制品的关系，这也是一个多对多关系。
4. useof 关系：表示用户与 application 之间的使用关系，很显然这是一个多对多关系。

下面针对关系型数据库和图数据库具体设计：

关系型数据库（MYSQL）

关系型数据库简介

关系型数据库（Relational Database）是一种基于关系模型的数据库管理系统（DBMS）。在关系模型中，数据以表（Table）的形式组织，每个表包含若干行（Records）和列（Columns），其中每一行代表一个实体记录，每一列代表一个数据字段。

MYSQL 简介

MySQL是一种开源的关系型数据库管理系统（RDBMS），广泛应用于各种应用程序和网站中。它以其可靠性、高性能和灵活性而闻名，并被广泛用于各种规模的项目和企业。

设计

根据上文的实体与关系，进行关系型数据库的建立

1. 我对实体以及他们所要表达的属性进行实体表的建立。
2. 然后对一对多关系进行建表，这里的一对多关系在建表时候需要与多端的实体集合进行合并，所以这里我用外键约束进行表示。
3. 对多对多关系进行建表，就是取两个实体集合的主键共同作为多对多关系表的主键，同时这两个也是参照别的表的外键。

具体的表模式

实体表：

1. LoadBalancer 表：
   - LBid (PK) - varchar(10)
   - 名称 - varchar(20)
   - IP地址 - varchar(25)
   - 端口 - varchar(15)
   - 负载分发算法 - varchar(20)
   - SSL加速 - varchar(5)
   - 会话保持 - varchar(5)
   - Rid (FK-Rack(Rid)) - varchar(10)

2. Server 表：
    - Sid (PK) - varchar(10)
    - 名称 - varchar(20)
    - IP地址 - varchar(25)
    - 操作系统 - varchar(15)
    - 处理器类型 - varchar(10)
    - 内存容量 - varchar(10)
    - Rid (FK-Rack(Rid)) - varchar(10)
3. Rack 表：
    - Rid (PK) - varchar(10)
    - 名称 - varchar(20)
    - 位置 - varchar(25)
    - 容量 - varchar(10)
4. VirtualMachine 表：
    - VMid (PK) - varchar(10)
    - 名称 - varchar(20)
    - 版本 - varchar(15)
    - 访问控制 - varchar(5)
    - 认证配置 - varchar(5)
    - 容错设置 - varchar(5)
    - Sid (FK-Server(Sid)) - varchar(10)
5. Application 表：
    - Aid (PK) - varchar(10)
    - 名称 - varchar(20)
    - 版本 - varchar(15)
    - 开发者 - varchar(20)
    - 简述 - varchar(50)
    - 占用空间大小 - varchar(10)
    - DBSid (FK) - varchar(10)
6. Category 表：
    - Cid (PK) - varchar(10)
    - 名称 - varchar(20)
    - 说明 - varchar(50)
7. DatabaseServer 表：
    - DBSid (PK) - varchar(10)
    - 名称 - varchar(20)
    - IP地址 - varchar(25)
    - DBMS类型 - varchar(20)
    - 版本 - varchar(15)

- 处理器类型 - varchar(10)
- 内存容量 - varchar(10)

8. User 表:
    - Uid (PK) - varchar(10)
    - 性别 - varchar(5)
    - 年龄 - varchar(5)
    - 邮箱 - varchar(25)

关系表:

1. runon 表:
    - Aid (PK) (FK) - varchar(10)
    - VMid (PK) (FK) - varchar(10)
2. Belong 表:
    - Aid (PK) (FK) - varchar(10)
    - Cid (PK) (FK) - varchar(10)
3. Replicia 表:
    - 复制品DBSid (PK) (FK) - varchar(10)
    - 被复制DBSid (PK) (FK) - varchar(10)
4. useof 表:
    - Uid (PK) (FK) - varchar(10)
    - Aid (PK) (FK) - varchar(10)****

数据

这里的数据集是根据实际情况进行了简化的设计，然后将数据都编写成 insert 语句的 SQL 语句，直接运行既可以实现数据的插入。

图数据库（NEO4J）

图数据库简介

图数据库是一种特殊类型的数据库，旨在存储和处理图结构数据。图数据库以图（Graph）的形式组织数据，其中图由节点（Nodes）和边（Edges）组成，节点表示实体，边表示节点之间的关系。

NEO4J 简介

Neo4j 是一种流行的图数据库管理系统，旨在存储、管理和处理图结构数据。它是一款高性能、可扩展和灵活的图数据库，广泛应用于各种领域，如社交网络分析、推荐系统、**知识图谱**、网络和IT运维管理等。

设计

根据上文的实体与关系，进行图数据库的建立

1. 我对实体以及他们所带的标签进行实体节点的建立，将上面8个实体表中的所有记录都作为一个节点建立，他们所带的标签是将对应属性列去掉外键属性列的属性列表。
2. 然后对一对多关系进行边的建立，这里的一对多关系在上面的外键约束中隐含着，所以在建边时需要去对每一个有外键约束的实体表的每一行记录，抽取其外键属性列，根据匹配到的两个节点进行建立边。
3. 对多对多关系表进行边的建立，读取多对多关系表，然后根据匹配到的节点对进行边的建立，注意这里要针对每行记录建立两条边，这两条边的节点相同但是方向相反。

具体的图数据库模式

节点:

- (机架x : rack {Rid: value1, name: value2, address: value3, capacity: value4})
- (负载均衡器x : loadbalancer {LBid: value1, name: value2, IP: value3, port: value4, load: value5, SSL:value6,session: value7})
- (虚拟机x : virtualmachine {VMid: value1, name: value2, version: value3, visitcontrol: value4, auth: value5, allowerror:value6})
- (应用程序x : application {Aid: value1, name: value2, version: value3, dev: value4, profile: value5, size:value6})
- (X : category {Cid: value1, name: value2, profile: value3})
- (数据库服务器x : databaseserver {DBSid: value1, name: value2, IP: value3, DB_class: value4, version: value5, pro_class:value6, size:value7})
- (userxxx : user {Uid: value1, gender: value2, age: value3, email: value4})
- (服务器x : server {Sid: value1, name: value2, IP: value3, operationsystem: value4, processor: value5, memoryspace:value6})

外键（一对多关系）代表的边:

- (负载均衡器x : loadbalancer)-[r : LdB_Rack]→(机架x : rack)
- (服务器x : server)-[r : server_Rack]→(机架x : rack)
- (虚拟机x : virtualmachine)-[r : VM_Srv]→(服务器x : server)
- (应用程序x : application)-[r : App_DBS]→(数据库服务器x : databaseserver)

多对多关系代表的边:

- (应用程序x : application)-[r : RunOn]→(虚拟机x : virtualmachine)
- (虚拟机x : virtualmachine)-[r : RanOn]→(应用程序x : application)
- (应用程序x : application)-[r : Belong]→(X : category)
- (X : category)-[r : BeBelonged]→(应用程序x : application)

- (userxxx : user)-[r : UseOf]→(应用程序x : application)
- (应用程序x : application)-[r : BeUsedOf]→(userxxx : user)
- (数据库服务器x : databaseserver)-[r : Replicia]→(数据库服务器x' : databaseserver)
- (数据库服务器x' : databaseserver)-[r : BeReplicia]→(数据库服务器x : databaseserver)

数据

这里的数据集是根据 mysql 的数据集进行变化而来的，我们利用 python 程序实现了对节点、一对多关系（外键）、多对多关系（关系表）的抽取。

扩展功能：DJANGO

DJANGO 简介

Django 是一个高级的 Python 网络框架，可以快速开发安全和可维护的网站。由经验丰富的开发者构建，Django 负责处理网站开发中麻烦的部分，因此你可以专注于编写应用程序，而无需重新开发。 它是免费和开源的，有活跃繁荣的社区，丰富的文档，以及很多免费和付费的解决方案。

设计

这里为了更加便捷我们选取了链接 mysql 的方式来实现，在 views 文件中定义出一些函数，连接数据库后直接对数据库进行增删改查，类似于课堂作业进行路由相关的配置和设置，注意为了避免采用映射类的数据库，我们直接使用前文已经建好的数据库进行连接。同时我们对 html 页面进行了适当的美化（基于 Material Design 样式与布局），增强可视性。而且我们还利用到了重定位和跳转技术。

```python
def result1(request):
    results = simpleQuery(
        "databaseserver", ["DBSid", "IP地址", "DBMS类型", "版本", "处理器类型", "内存容量"], ""
    )
    values = []
    for result in results:
        values.append(list(result.values()))
    return render(
        request,
        "DataCenterManager\\result.html",
        {"keys": results[0].keys(), "values": values},
    )
```

例如在 `views.py` 中的上述代码，我们使用 mysqlclient 方法查询所有数据库服务器的相关信息，然后使用Django的模板语言向 `result.html` 传入参数，进一步将 SQL 结果显示在网页中。

```HTML
<body>
  <table>
    <tr>
      {% for col_name in keys %}
        <th>{{ col_name }}</th>
      {% endfor %}
    </tr>
    {% for value in values %}
    <tr>
      {% for v in value %}
        <th>{{ v }}</th>
      {% endfor %}
    </tr>
    {% endfor %}
  </table>
</body>
</html>
```

如上述 html 代码，我们首先通过查询结果获取得到 `keys` 和 `values`，分别对应查询结果的列名和字段值。然后，使用 for 循环进行遍历，在网页中显示相应的结果。在实际的代码里，我使用了谷歌 Material Design 的 CSS 样式进行显示。

其他扩展内容：模拟场景应用、复杂查询等

模拟场景应用

我们基于用户 User 类以及其使用的应用所属的种类 Category，实现了一种应用**推荐**机制。我们使用 SQL 查询进行实现：

```SQL
SELECT Aid, 版本, 名称, 简述
    FROM application
    WHERE Aid IN (
        SELECT DISTINCT Aid
        FROM belong
        WHERE Cid IN (
            SELECT Cid
            FROM user
            NATURAL JOIN useof
            NATURAL JOIN belong
            NATURAL JOIN category
            WHERE Uid='{user}'
        )
        AND Aid NOT IN (
            SELECT DISTINCT Aid
            FROM useof
            WHERE Uid='{user}'
        )
    )
    ORDER BY Aid ASC
```

其中 `'{user}'` 是格式化字符串，网页中可以选择，表示为特定的用户进行推荐。

复杂查询

在 MySQL 部分，我们也实现了**复杂查询**的功能，可以进行两个表连接（笛卡尔积）形式的查询。此外，我们也有 `CustomSQL` 函数，用于执行任意的合法 SQL 字符串，并返回得到结果，正如上面的推荐方法也是调用的此函数。

图数据库与关系型数据库的分析

关系型数据库（例如 MySQL）的一张表对应于图数据库的一个 label，图数据库一般有比较多的节点，以 label 进行分类。

关系型数据库有以下缺点：

- **建模难**: 不复杂***就不能建模和存储数据和关系***
- **性能低**: 随着关系数量和层次的增加，数据库尺寸的增加，***性能降低***
- **查询难**: 由于需要 JOIN 操作，***查询复杂性增加***
- **扩展难**: ***增加新类型的数据和关系***，需要重新设计模式，增加了上市时间

而图数据库 Neo4j 则具有**开发优势：模型维护容易、查询简单**，以及部署优势：**超高性能、使用最少的资源。**

同时，我们也应清晰地认识到，真正高性能的数据模型，取决于业务数据使用形态，在图数据模型以及范式化数据模型之间取得一个适合于业务的平衡。