Data Engine 关联图数据结构设计

计

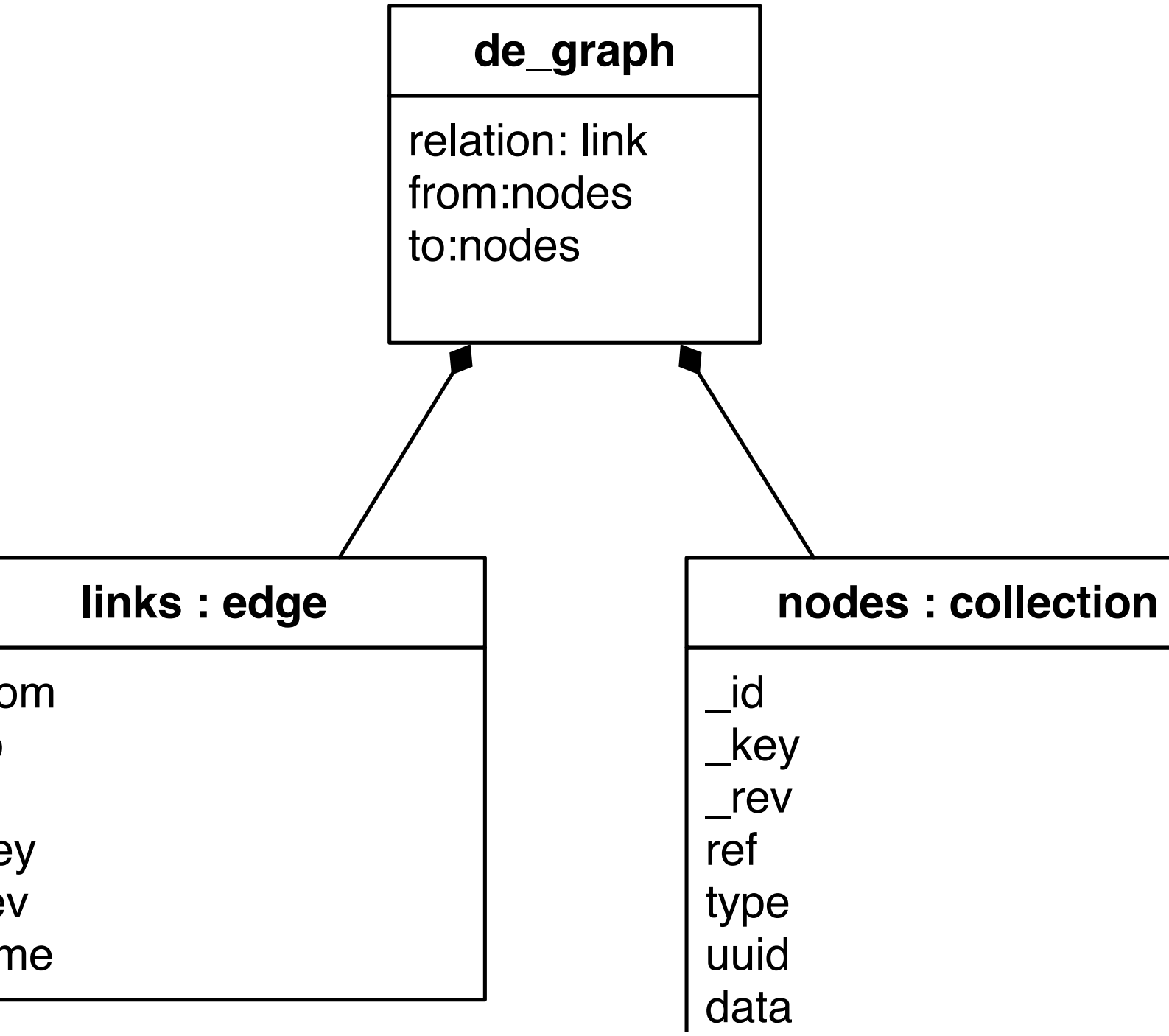
平台的构建哲学与理念是建立在Graph之上。作为数据引擎的de子系统的底层数据结构当然是优选建立在graph db之上。

下，de数据结构要全面采用graph结构，但是由于受限于当前的技术水平，考虑到执行效率因素，需要权衡底层数据结构的Graph化程度。

首先是Graph结构的表的存取效率肯定比简单Key-Value结构表的低很多。以arangodb为例，除了由于数据结构复杂和跨表操作造成的效率降低问题外，graph操作还大量引入transaction操的重要因素。其次，考虑到数据的快速查询，必须为数据结构建立索引，而高效率索引建立的关键就是要维持同一张表中各条记录保持基本相同的shape（即字段结构定义），为此，也需要把结到同一个数据集合中，在arangodb中以collection形式组织。

，作为数据引擎，de设计时并不局限于arangodb这一种数据库和数据源。de的核心graph会以outline和branch形式聚合各种类型、不同数据源的数据。这些成为graph叶子节点的数据，同collection中的document；也可以是其他arangodb database中的document；甚至可以是其他关系型数据库表中的数据；抑或是部署在互联网上的Web数据。

考虑，我们首先需要一张以数据关系为纲，然后其他各种数据源的寻址定位信息为目的graph。这张graph负责操作数据节点之间的关联，采用与数据节点无关的最简化的通用结构。我们称这张如下图所示。



edge collection和一个vertex collection组成，分别对应下图中的links表和nodes表。其中links表中带“-”前缀的字段是arangodb的系统自定义字段，唯一自定义的字段是name，用于关联图
不支持内建字段与自定义字段的复合索引，需要在取值上使name唯一。其命名改变为\key-of-from/name。

中自定义字段中的ref字段，用于存放vertex的数据源寻址信息。默认情况下，type类型是`undefined`及本地**arangodb**数据源，ref中存放的是document handler，即“collectionName/_key”
db数据库中；当`type`类型是`arangodb`时，`ref`中存放三段结构“databaseName/collectionName/_key”，是在本地arangodb的不同数据库中寻址。上述两种情况下，`ref`字段保持全集唯
通过*Document Handle*查找*node*节点。

的可选值为`"_self", "arangodb", "rest", "file", "uri", "function", "odbc"`等等时，*数据源*数据存放在`data`字段中，而不是`ref`字段。常用的是前三种，其中`rest`类型时存放*REST*接口
类型时存放本地文件路径。`uri`类型涵盖`rest`和`file`两种类型，存放数据采用*URI*格式。`function`类型是万用类型，`data`中存放函数名称和参数。`odbc`类型则存放*ODBC connection*信息和

中的uuid是一个可选字段，用于提供一个与DBMS无关的Key-Value快速存取方式，其中uuid采用UUID格式和定义。uuid字段可以用于nodes所指向内容的反向寻址。

个提供兼容性的字段，数据实体可以直接存放在nodes表中。当前设计不推荐直接在nodes表中存放最终数据。当使用本地数据而非外部数据时，type取值为“_self”，ref的取值没有意义。

式

于持久化数据对象之间的关系。最终数据的寻址信息采用两段、三部分内容进行组织。第一段以“/”分隔，用于在关联图中以graph方式定位数据节点，第二部分以“.”分隔在数据节点内部进行
寻址和本地寻址信息外，还有一个stub，用于在关联图中锚定一个root节点，以此根节点为相对寻址的起点。参考如下示例：

```
/3100950/model/spec.screen.resolution
```

位于寻址的第一段，采用“/”分隔。stub通常分两部分。第一部分为stub寻址方式定义，暂时定义为如下几种：

，采用nodes表中的uuid字段定位stub。

如：`"/uuid:/08486BD2-7339-4406-937E-90A75B47B3F3/"`

，采用arangodb寻址方式，以nodes表中document handler定位stub。当前关联图中仅支持一个edge collection，因此Collection名称省略。

如：`"/key:/782440272"`

内容id定位stub。

如：`"/foo/531437392"`

的是，由于类似arangodb这种数据库会使用纯数字的id，其命名规则与javascript的变量命名规则不同。因此，“.”在寻址方式应用中，很容易与数字标点混淆。因此在实际的脚本系统中，通常
如：

```
/foo/531437392/bar"}} {{name}}: {{a.b.c}} {{/path}}
```

时，数据分别来自于`/foo/531437392/bar.name`，和`/foo/531437392/bar.a.b.c`。