KG 存储 & Neo4j

基于RDF存储

- 关注点 —— 数据易于发布、共享
- 以三元组（SPO）形式存储数据
- 实体和关系不包含属性信息
- 标准的推理引擎(RDFS扩展的OWL，自动推理)
- W3C标准
- 多数应用在学术界场景

基于图数据库存储

- 关注点 —— 高效地图查询与搜索
- 以属性图作为基本的表示形式
- 节点和关系包含属性
- 没有标准的推理引擎
- 完全兼容ACID
- 基本应用于工业界场景

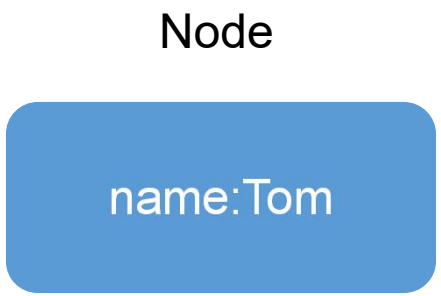
根据统计

- Neo4j
  - 使用率最高图数据库
  - 系统本身查询效率最高
  - 拥有活跃的社区
- OrientDB、JanusGraph
  - 系统相对较新
  - 社区较为沉寂
- Jena
  - 针对RDF的存储系统

数据模型

- 节点（Node）
  - 用以表示一个实体记录
  - 能包含多个属性（Property）、多个标签（Label）

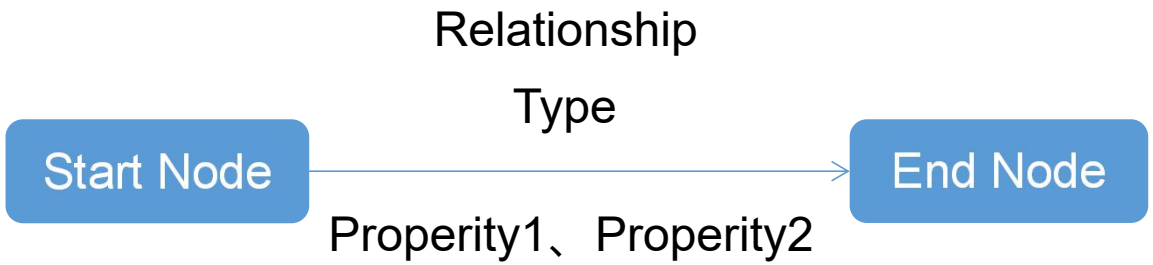
带属性和标签的节点



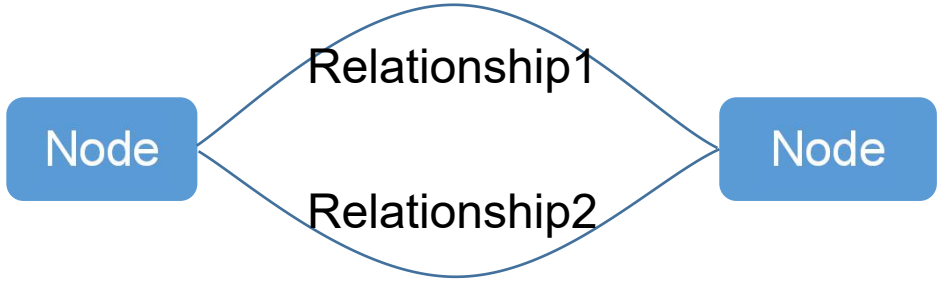
只有属性的节点

数据模型

- 关系（Relationship）
  - 对应图论的边（Edge），连接两个节点
  - 要求：起始端、末端都必须是节点，不能为空
  - 可包含多个属性，但是只能有一个类型（Type）



带有类型和属性的关系



多个关系指向同一节点

数据模型

- 属性（Properity）
  - 由键值对组成
  - 属性值可以是基本数据类型、数组
  - 属性值没有null的概念

- 遍历
  - Neo4j 提供一套遍历API
  - 根据给定的遍历规则，自动遍历并返回结果
  - 深度优先、广度优先

Cyhper：图数据库查询语言

一种声明式的模式匹配语言

地位和作用如同关系数据库中的SQL

专注于清晰地表达从图中检索什么（what），而非怎么去检索（how）

Cyhper：创建节点和关系


```
CREATE ( Keanu: Person  { name: "Keanu Reeves",  born: 1964 } )
```

节点名　标签名称　属性1　属性2

```
CREATE (TheMatrix: Movie { title: "The Matrix",
        released: 1999, tagline: "Welcome to the Real World"} )
```

```
CREATE (Keanu)-[: ACTED_IN {roles: ["Neo"]}]->(The Matrix)
```

节点1　关系　关系的属性名与属性值　节点2

Cyhper：匹配图模式

查找人员、电影节点等

- MATCH （cloudAtlas {
  title: “Cloud Atlas”})
- RETURN cloudAtlas

查找多个人物（随机查找10个人物）

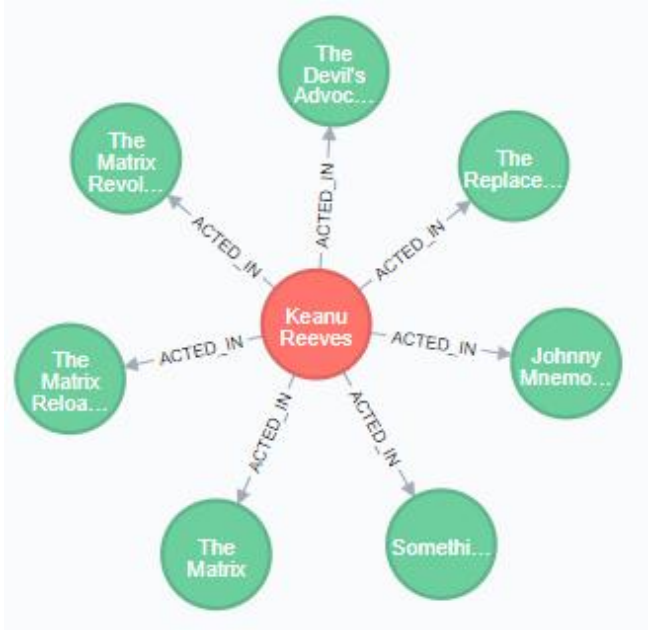
- MATCH （people: Person)
- RETURN people.name LIMIT 10

Cyhper：匹配图模式

查找关系

以“查找 Keanu Reeves 参演过的电影” 为例

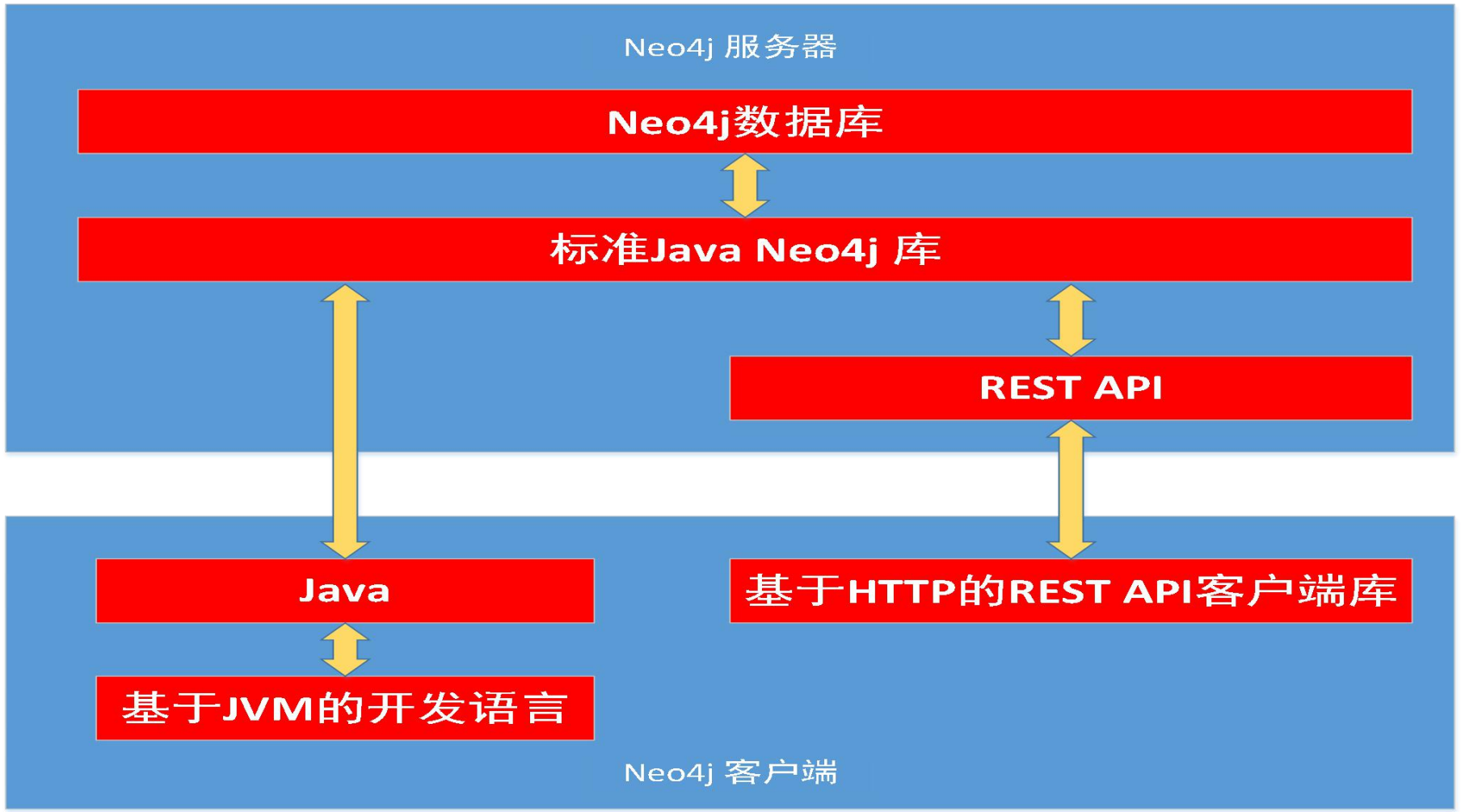
- MATCH (Kea: Perosn {name : “Keanu Reeves”}) - [: ACTED_IN] -> (KeaReeMovies)
- RETURN Kea, KeaReeMovies



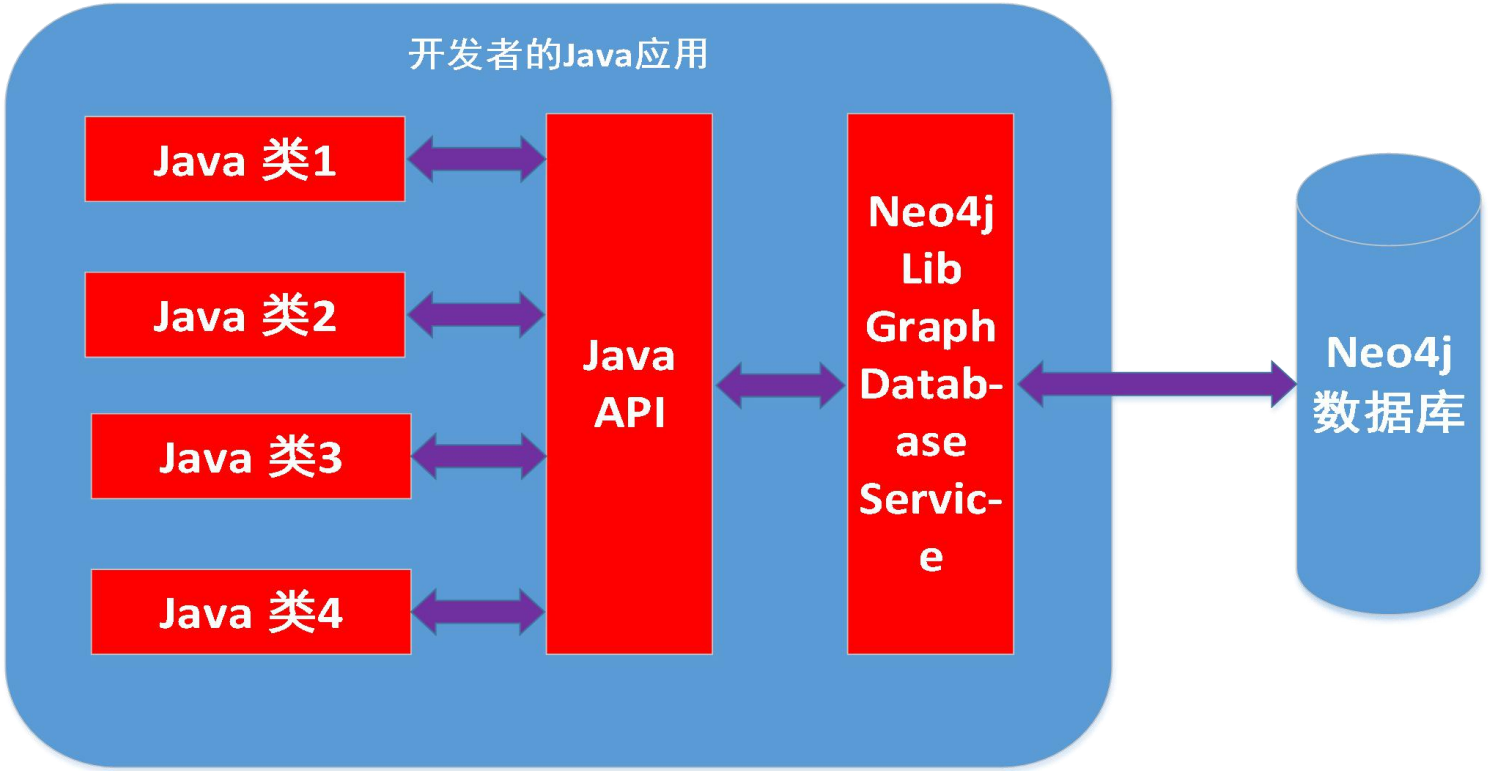
Neo4j 程序开发

- 支持 .Net、Java、JavaScript、PHP、Python的二进制Bolt协议驱动程序
- 引入相应驱动程序包，就能与Neo4j相互集成
- 实例代码：https://github.com/neo4j-examples
- 开发模式
  - Java嵌入式开发模式
  - 各语言驱动包开发模式

Neo4j开发模式结构图



Java嵌入式应用结构图



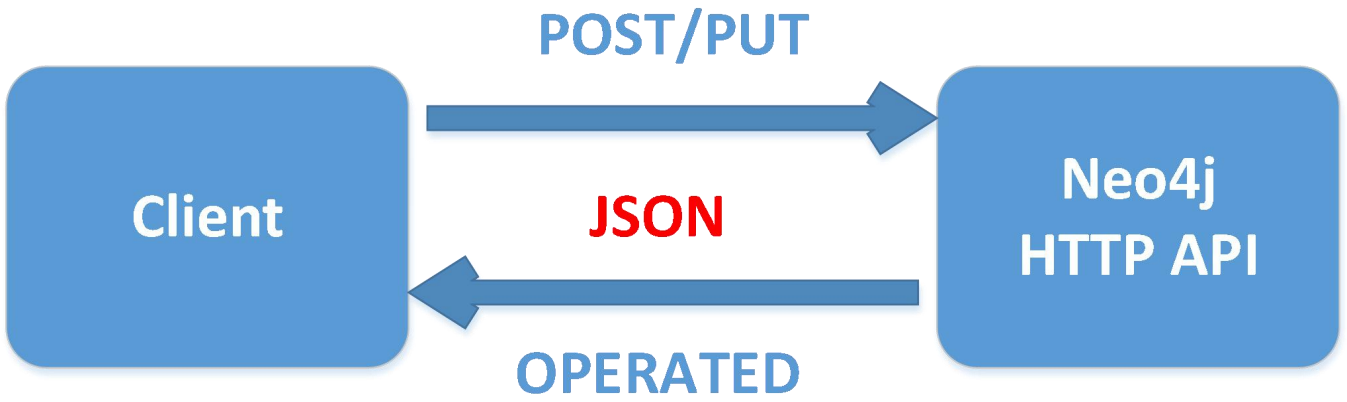
- Neo4j 所有操作逻辑、遍历、查询均可由Java程序实现
- 除了操作节点、关系和路径对象，还能定制高速遍历和图算法
- 优势体现在插入大量数据，能轻松地摄取数十亿个节点和关系

各语言驱动包开发模式

- Neo4j驱动程序使用Bolt协议进行通信
- 借助驱动，可与数据库基于事务的回话
- 会话中，事务可以：
  - 创建和查询数据库
  - 定义数据库模式
  - 监视和管理数据库
- 在因果集群中，驱动程序可以：
  - 处理读\写操作的路由
  - 处理负载均衡

Neo4j HTTP API

- 一套与开发平台、开发语言无关的API
- 默认情况是使用JSON传输



其他开发技术简介

- Spring-Data-Neo4j 库
  - 使用该库访问Neo4j，包括对象映射、Spring数据存储库、转换、事务处理等
- Neo4j-OGM 库
  - Spring-Data-Neo4j 中已集成
  - 提供快速全面的对象图映射

NBA季后赛预测

问题

确定两支球队间的胜率，可查看彼此的历史比赛：胜负场次数；
如果对阵双方无交手历史，可以分别计算胜率：总胜场次/总比赛场次；
季后赛中，每支球队的对手只有几支，所以，考虑胜率似乎作用不大

NBA季后赛预测

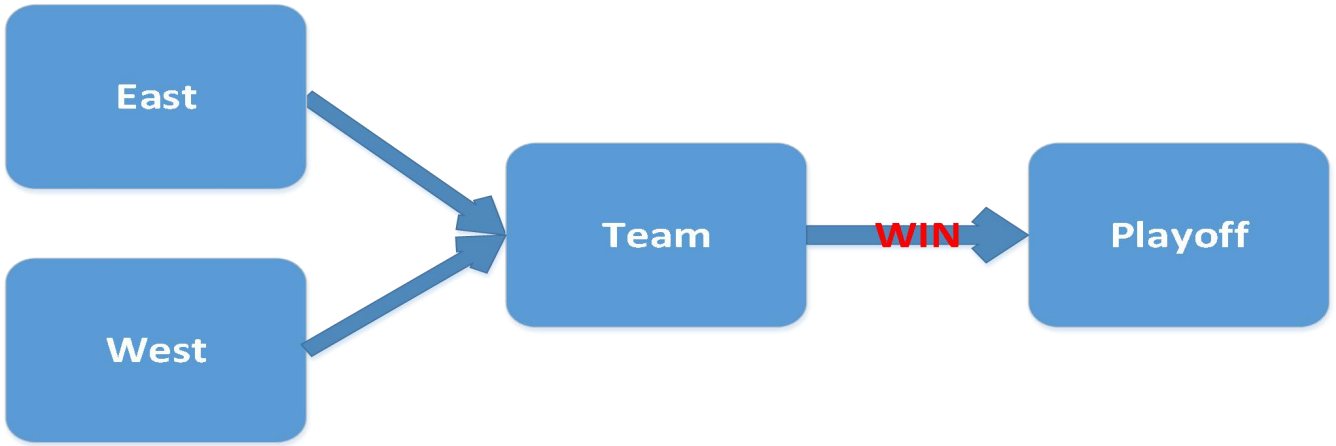
求解

统计季后赛中各支球队交手历史，利用图形数据库的 Path 概念，将任意两支球队连接。根据链接，找到任意两支球队间最密切的关系

为这些链接分配数值，便可衡量两支球队实力，给出预测

NBA季后赛预测

数据模型



- 球队节点属性：name、code 分别代表 球队名称、代码
- 比赛节点属性：year、round 分别代表 比赛年份、回合
- 关系类型WIN：属性win 表示在一轮比赛中赢得的场次

NBA季后赛预测

建立

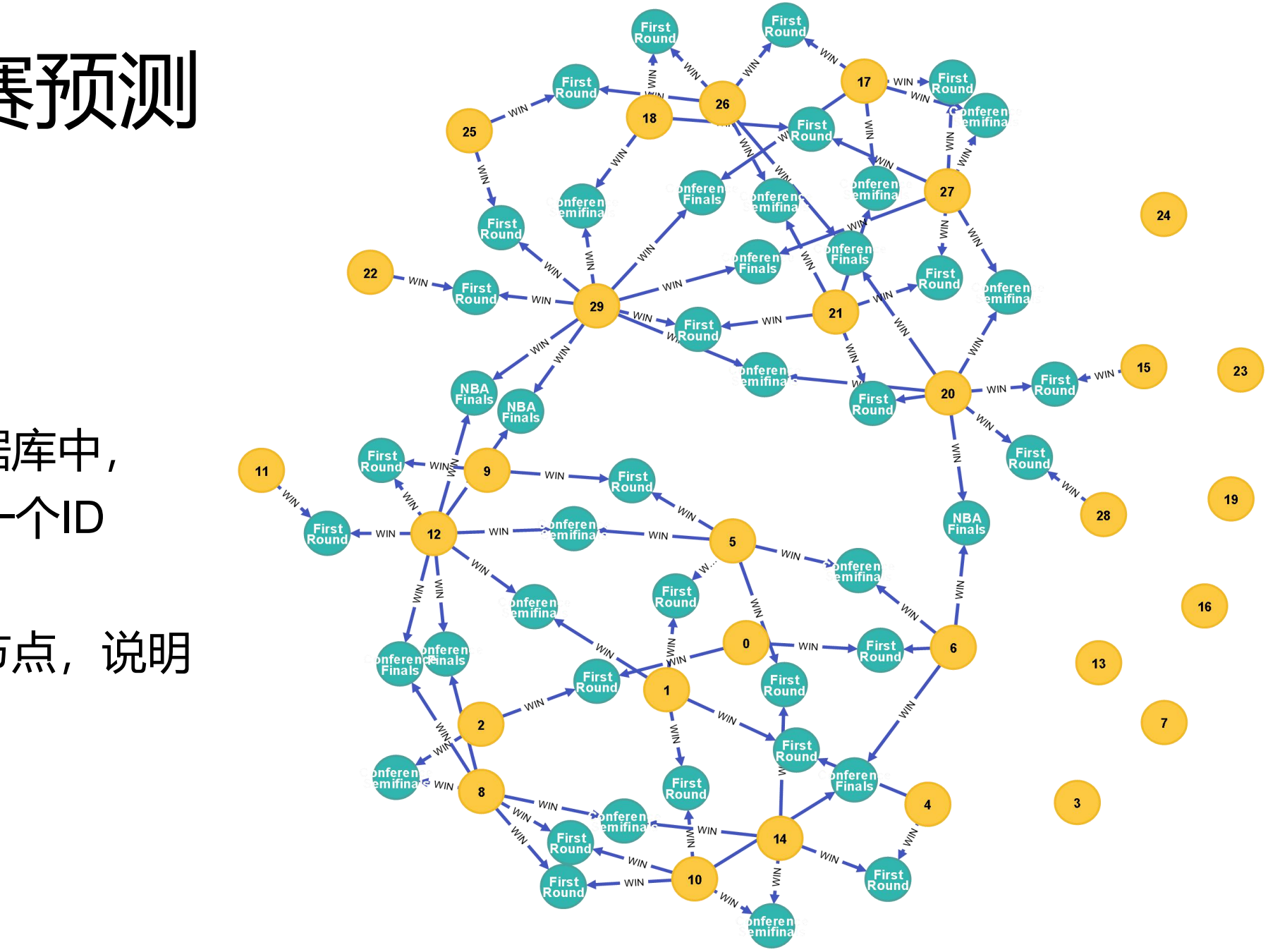
设置代码将近三年NBA季后赛结果加载到图表数据库作为样本数据

```
//create NBA TEAM nodes
CREATE (BOS:Team:E{Name: "Boston", Code: "BOS"})
CREATE (BKN:Team:E{Name: "Brooklyn", Code: "BKN"})
CREATE (NYK:Team:E{Name: "New York", Code: "NYK"})
.....
//create PLAYOFF nodes
CREATE (P201501:Playoff{Year: "2015", Round: "First Round"})
CREATE (P201502:Playoff{Year: "2015", Round: "First Round"})
CREATE (P201503:Playoff{Year: "2015", Round: "First Round"})
.....
//create PLAYOFF relationships between teams
CREATE (ATL)-[:WIN{Win:4}]->(P201501)
CREATE (BKN)-[:WIN{Win:2}]->(P201501)
CREATE (TOR)-[:WIN{Win:0}]->(P201502)
```

NBA季后赛预测

数据导入效果图

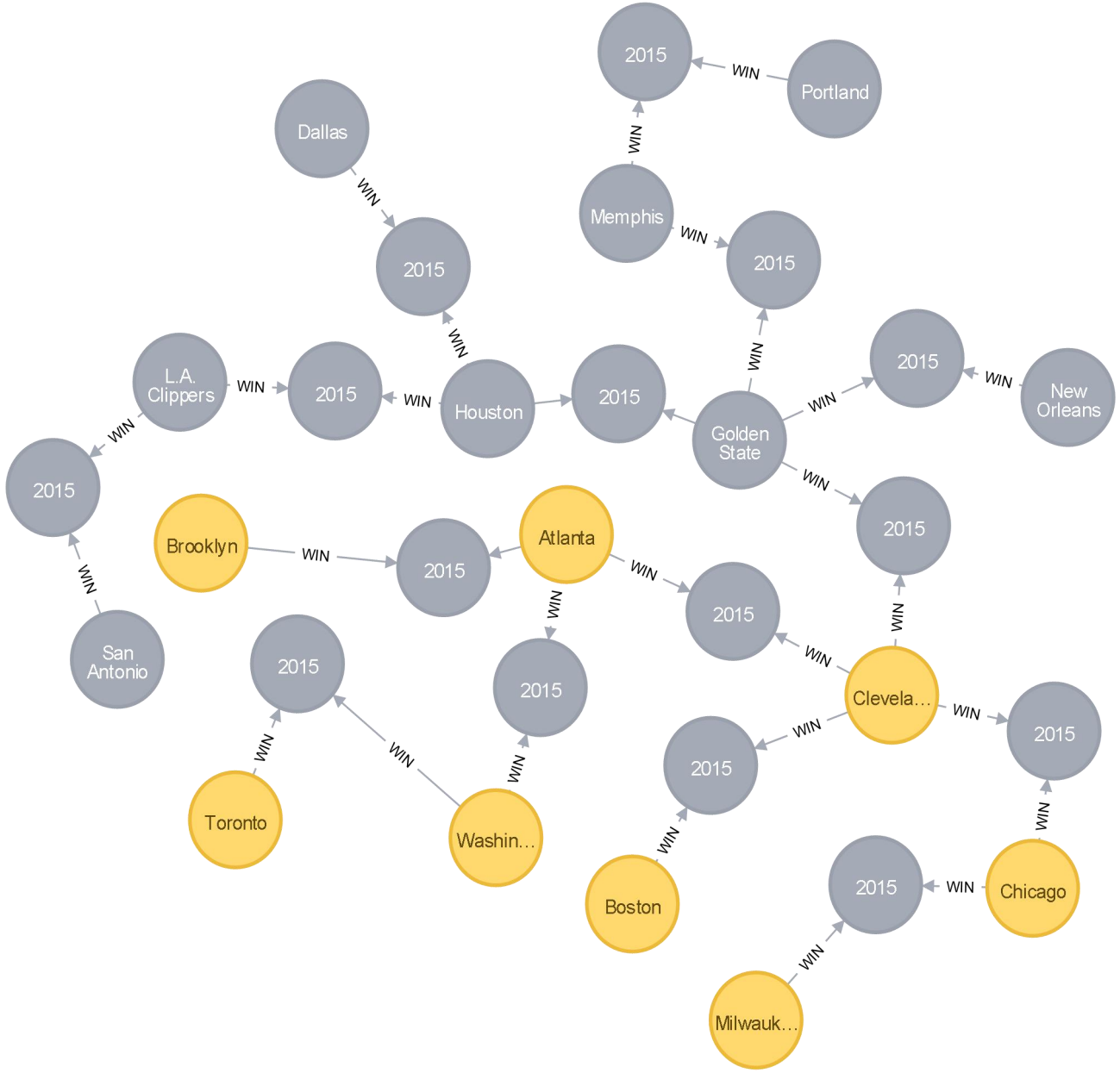
- 每支球队在图数据库中，都会被默认设置一个ID
- 没有建立关系的节点，说明未能进入季后赛



NBA季后赛预测

若想查询某年的季后赛成绩

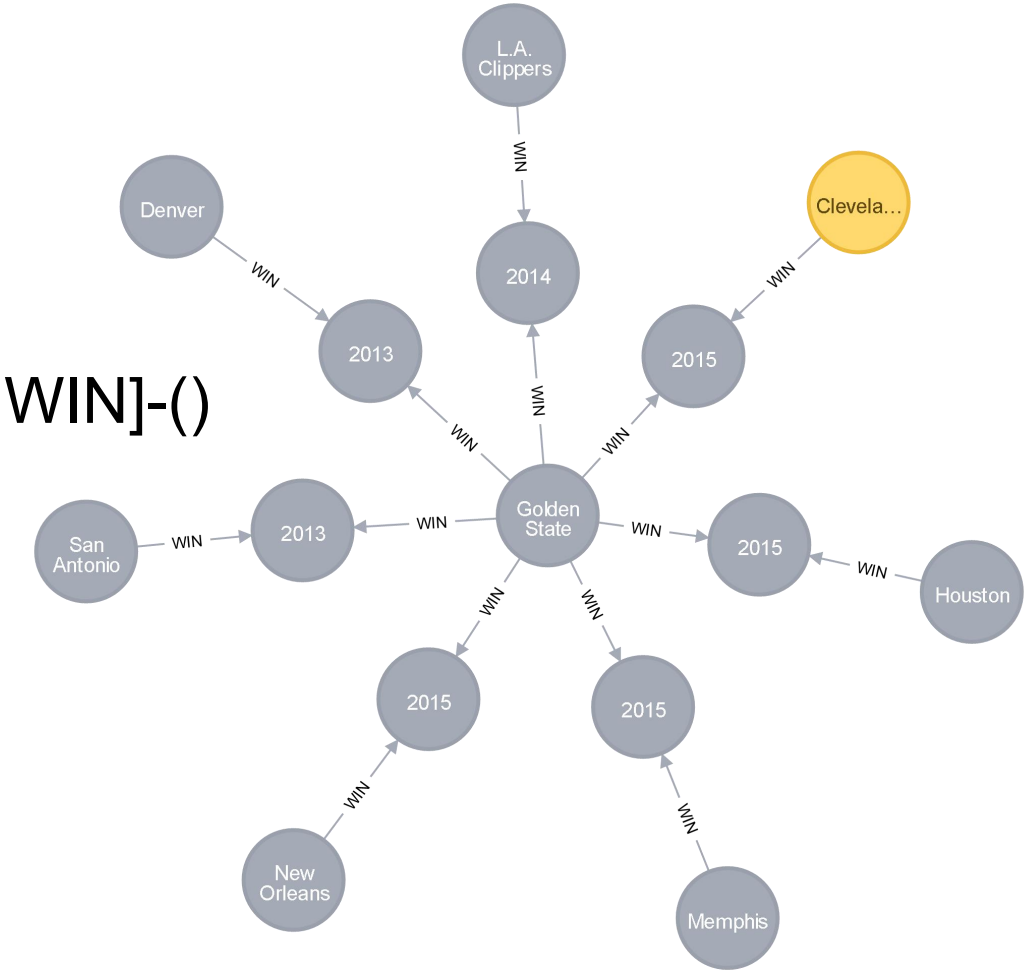
- MATCH (t)-[]->(p:Playoff)
- WHERE p.Year = "2015"
- RETURN t,p



NBA季后赛预测

若想查询某支球队的所有历史季后赛

- MATCH (t:Team {Name: "Golden State"})-[w:WIN]->(:Playoff)<-[l:WIN]-()
- RETURN t,w,l



NBA季后赛预测

列出最近3年中每年球队的所有历史输赢

- MATCH (t:Team)-[w:WIN]->(:Playoff)<-[l:WIN]-()
- RETURN t.Name AS TEAM, SUM(w.Win) AS TOTAL_WIN, SUM(l.Win) AS TOTAL_LOSS,
- (toFloat(SUM(w.Win)) / (toFloat(SUM(w.Win))+ toFloat(SUM(l.Win)))) AS WIN_PERCENTAGE
- ORDER BY SUM(w.Win) DESC

```
$ MATCH (t:Team)-[w:WIN]->(:Playoff)<-[l:WIN]-() RETURN t.Name AS TEAM, SUM(w.Win) AS TOTAL_WIN, SUM(l.Win) AS TOTAL_LOSS, (toFloat(…
```

Table | Text | Code

TEAM	TOTAL_WIN	TOTAL_LOSS	WIN_PERCENTAGE
"San Antonio"	34	17	0.6666666666666666
"Miami"	29	14	0.6744186046511628
"Golden State"	25	15	0.625
"Indiana"	21	17	0.5526315789473685
"Memphis"	17	16	0.5151515151515151
"L.A. Clippers"	15	18	0.45454545454545453
"Oklahoma City"	15	15	0.5
"Cleveland"	14	6	0.7
"Houston"	13	16	0.4482758620689655
"Atlanta"	13	16	0.4482758620689655

NBA季后赛预测

两支球队有交手历史

- MATCH (t1:Team {Name: "Miami"})-[r1:WIN]->(p:Playoff)<-[r2:WIN]- (t2:Team {Name:"San Antonio"})
- RETURN t1,r1,p,r2,t2

计算胜负

- MATCH (t1:Team {Name: "Miami"})-[r1:WIN]->(p:Playoff)<-[r2:WIN]-(t2:Team {Name:"San Antonio"})
- RETURN p.Year AS Year,r1.Win AS Miami,r2.Win AS San_Antonio
- ORDER BY p.Year DESC

```
$ MATCH (t1:Team {Name: "Miami"})-[r1:WIN]->(p:Playoff)<-[r2:WIN]-(t2:Team {Name:"San Antonio"}) RETURN p.Yea
```

Table

Text

Year	Miami	San_Antonio
"2014"	1	4
"2013"	4	3

NBA季后赛预测

两支球队没有交手历史

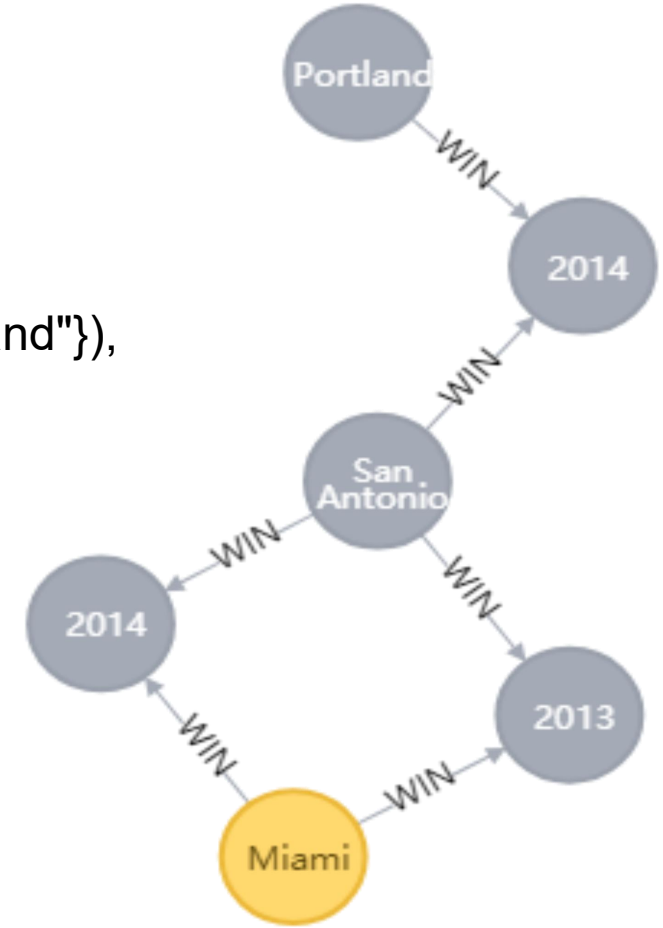
- 查找两支球队间的所有最短路径
- MATCH (t1:Team {Name: "Miami"}),(t2:Team {Name:"Portland"}),
  p = AllshortestPaths((t1)-[*..14]-(t2))
- RETURN p

```
$ MATCH p= AllShortestPaths
```

Table

Text

RArray
[4, 3, 4, 1]
[1, 4, 4, 1]



说明：一条路[1,4,4,1]表明两支球队可能有相同的获胜机会，
但迈阿密在另一支球队中显示出一点优势[4,3,4,1]

相关资源

- Neo4j官网: http://neo4j.com
- 在线数据库沙箱：https://neo4j.com/sandbox-v2/
- Github开源项目和代码库: https://github.com/neo4j-contrib/
- Neo4j推荐引擎构建:https://github.com/graphaware/neo4j-reco
- Neo4j程序开发实例代码:https://github.com/neo4j-examples
- 中文社区：http://neo4j.com.cn
- Neo4j 中文社区 QQ:547190638
- 《Neo4j权威指南》
- 《图数据库》(电子书)：https://neo4j.com/graph-databases-book/

Thank You For Watching