数据库系统概论新技术篇

图数据库管理系统

卢卫

中国人民大学信息学院

2017年6月

提纲

- 为什么要有图数据库管理系统
- 如何存储和查询图数据

提纲

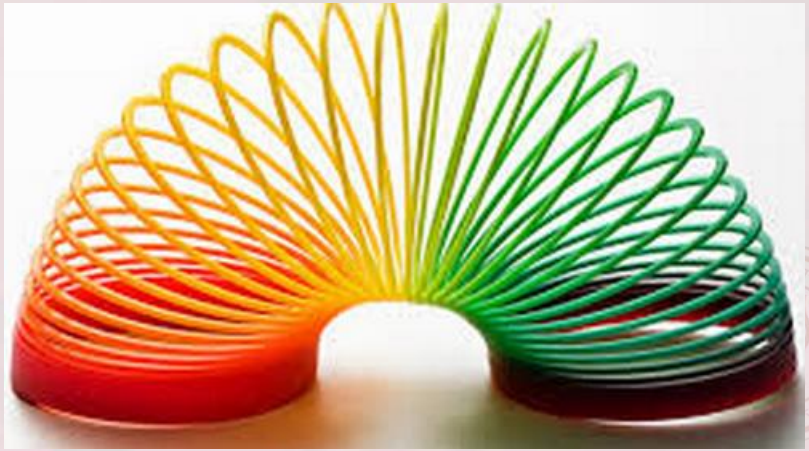
- 为什么要有图数据库管理系统
- 如何存储和查询图数据

为什么要有图数据库管理系统

- 相对于传统的关系数据库管理系统，图数据库管理系统在处理图数据时具有以下两个优点：

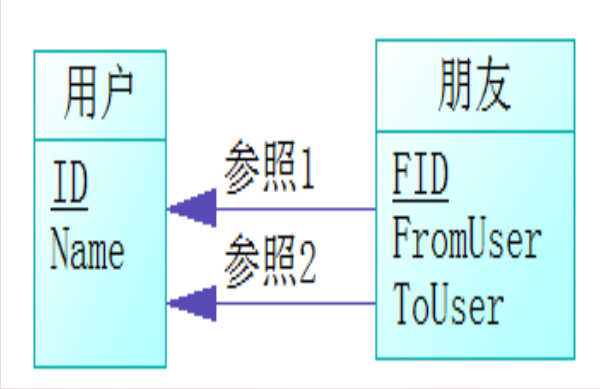
性能

灵活性



性能

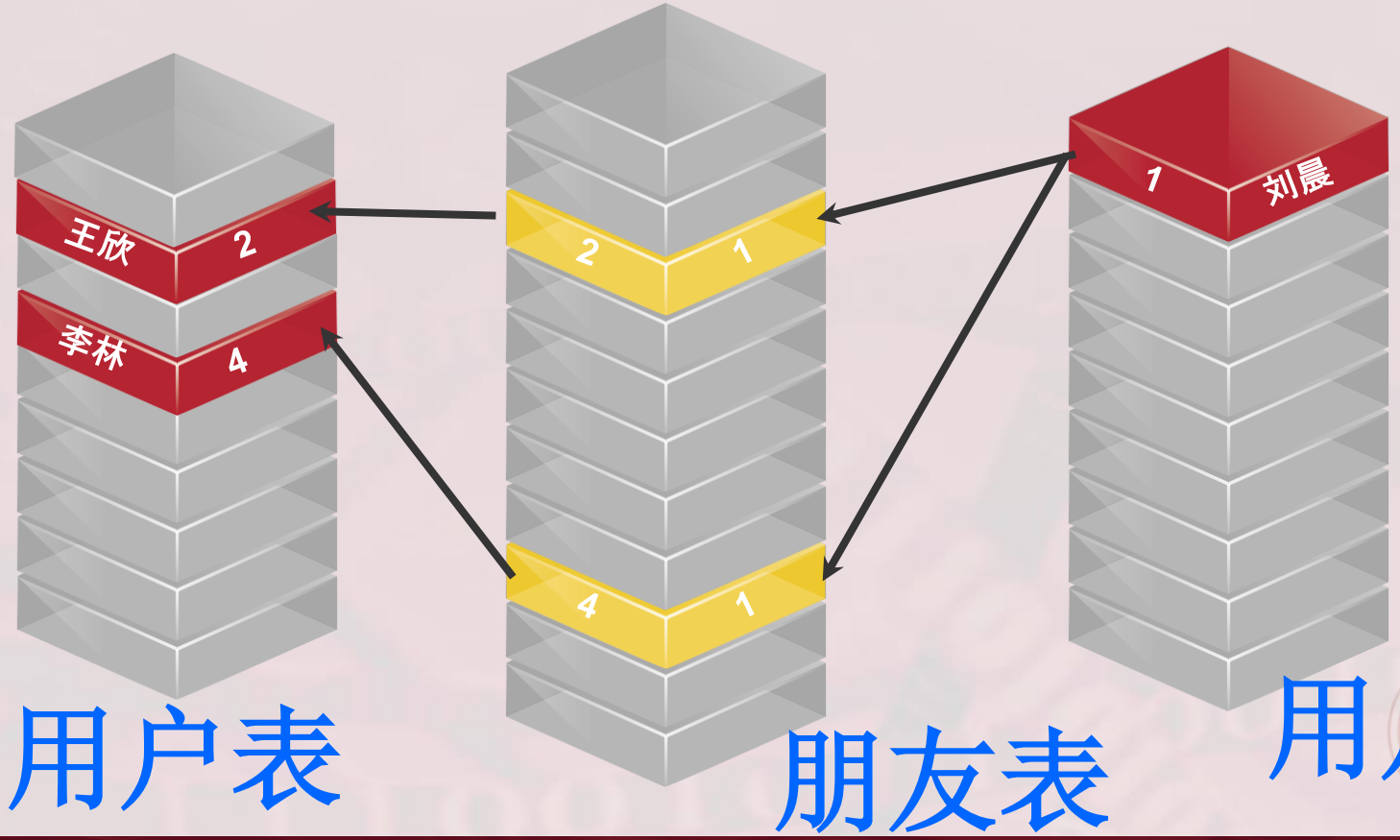
- 人，以及人与人之间的关系，构成了社交网络应用（例如微信、微博）的基本架构。社交网络的基本结构可以用如下的关系模式来描述



ID	Name
1	刘晨
2	王欣
3	张强
4	李林
5	马兴
6	吴曦

ID	FromUser	ToUser
1000	1	2
1001	3	5
1002	4	1
1003	6	2
1004	4	5
1005	1	4

性能（续）



性能（续）-社交网络中的查询

- 查询‘刘晨’的朋友数量----跳数为1
  - Select count(*) from 用户，朋友 where 用户.name = ‘刘晨’and 用户.ID = 朋友. FromUser
- 查询‘刘晨’的朋友的朋友数量----跳数为2
  - Select count(*) from 用户，朋友 as F1,朋友 as F2, where 用户.name = ‘刘晨’and 用户.ID = F1. FromUser and F1.ToUser = F2.FromUser
- 查询‘刘晨’的朋友的朋友的朋友数量----跳数为3
- …….

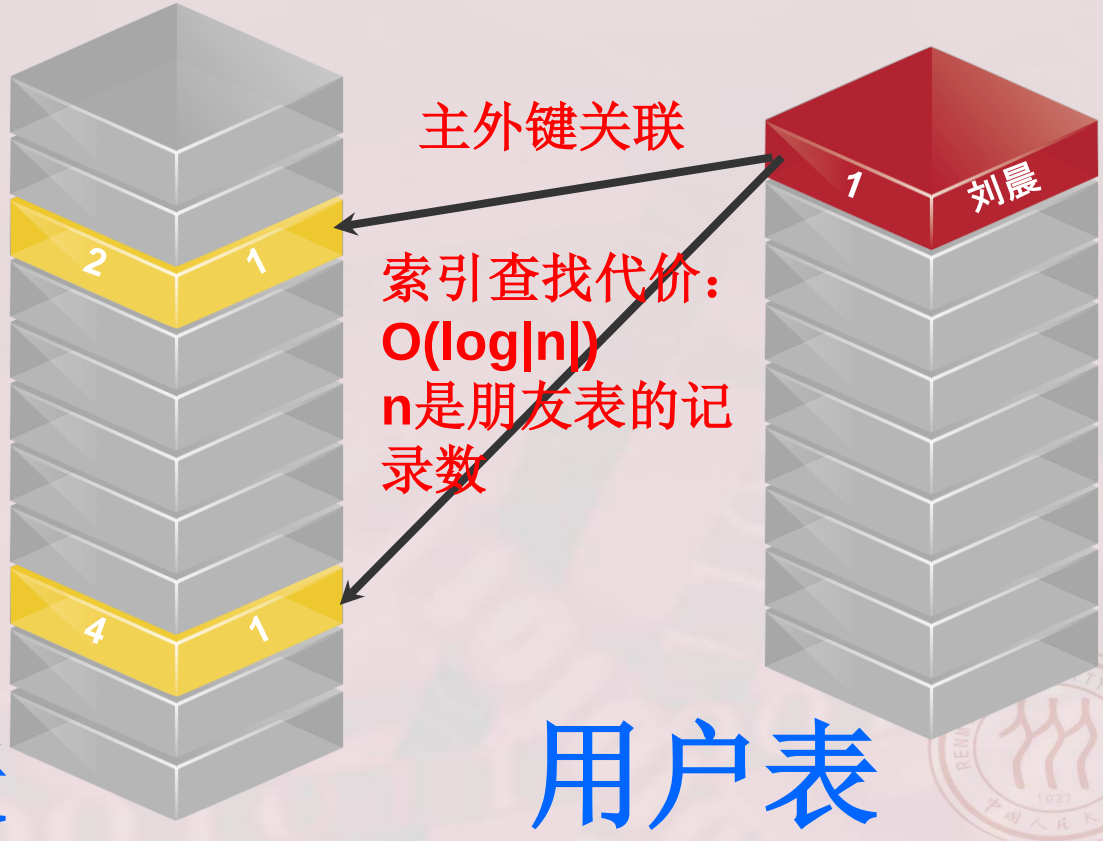
性能（续）-关系数据库 VS 图数据库

- 数据集大小 [Neo4J in Action]
  - 用户表：1百万条记录；朋友表：5千万条记录
  - 测试系统：MySQL VS Neo4J
- 实验结果

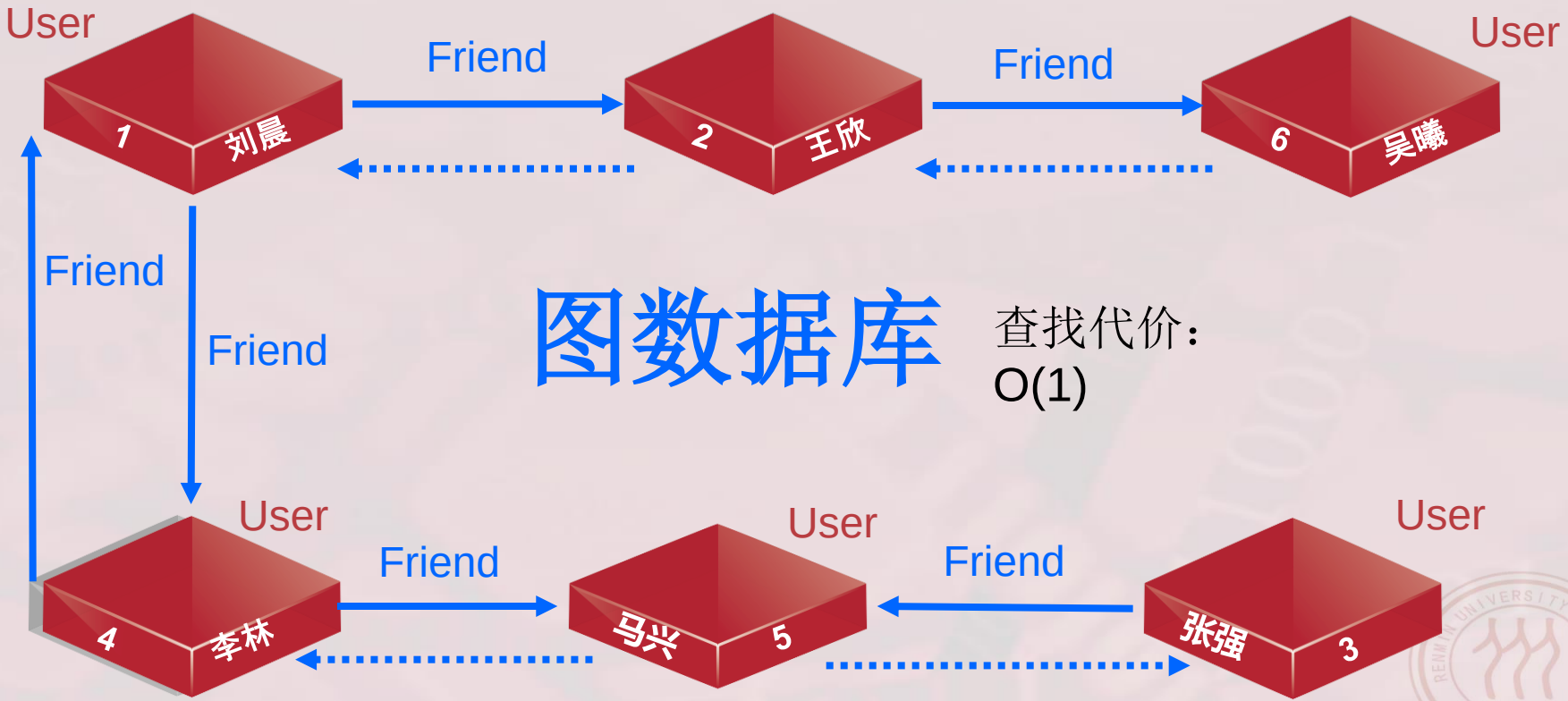
距离	MySQL（s）	Neo4J（s）	返回记录数
2	0.016	0.01	~2,500
3	30.267	0.218	~110,000
4	1543.505	1.359	~600,000
5	未完成	2.132	~800,000

性能（续）-图数据库的性能优势原因

关系数据库

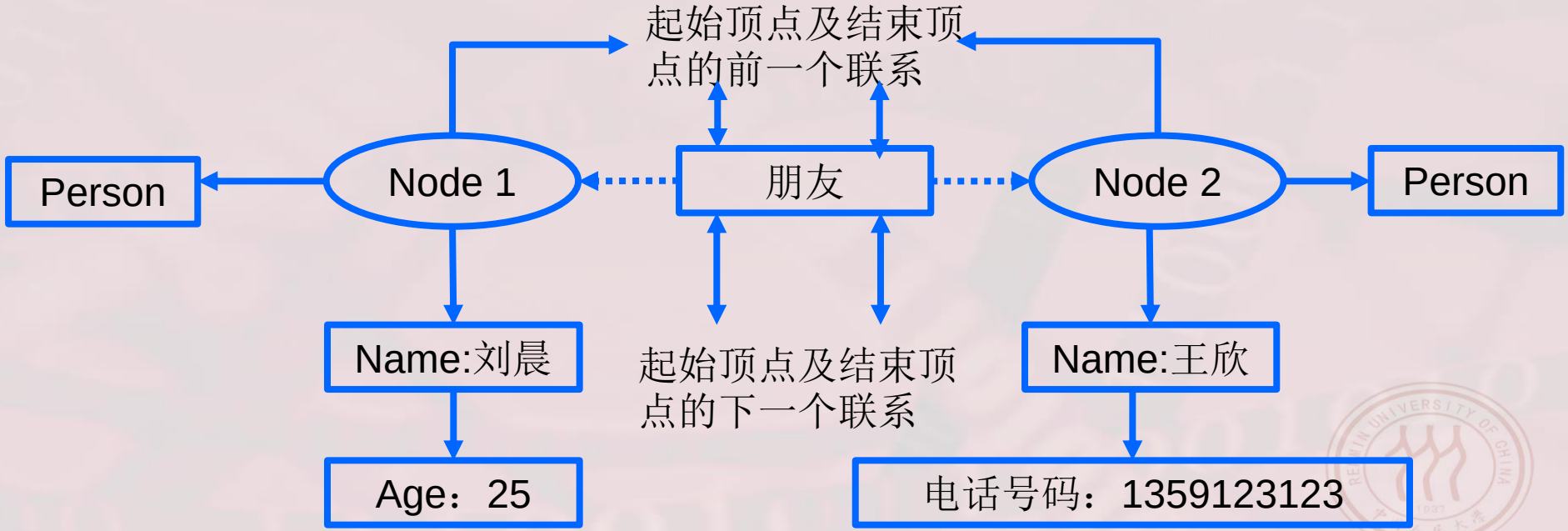


性能（续）-图数据库的性能优势原因



性能（续）-图数据库的存储系统

- 为每一类对象单独存成文件，每个对象的大小固定
  - 顶点文件；联系文件；标签文件；属性文件；等



灵活性

关系数据库

- 事先建立完整关系模式
- 模式修改代价大
- 互联网应用需要频繁修改模式

图数据库

- 无需事先建立模式
- 图结构的变化不影响已有的查询和应用程序

图数据库的主要功能

- 面向事务的图数据库能够支持
  - 数据对象（顶点与边）及其属性的增删改操作
  - 支持ACID事务处理
  - 并提供定制的查询语言来支持管理、分析和检索操作
  - 数据的备份与恢复
  - ……

提纲

- 为什么要有图数据库管理系统
- 如何存储和查询图数据

图数据库的分类

- 根据存储模型的不同，该类数据库又可分为以下四类
  - 基于关系存储模型开发的图数据库
    - 典型的代表是基于**MySQL**开发的**Twitter**公司的**FlockDB**
  - 基于图存储模型的专用图数据库
    - 典型的代表包括**Neo4J**、**Titan**
  - 基于文档存储模型开发的图数据库
    - 典型的代表是**OrientDB,**
  - 基于键值对存储模型开发的图数据库
    - 典型的代表包括**Apache Accumulo**

图数据库产品比较

❖ https://db-engines.com/en/ranking/graph+dbms

27 systems in ranking, June 2017

Rank Jun 2017	Rank May 2017	Rank Jun 2016	DBMS	Database Model	Score Jun 2017	Score May 2017	Score Jun 2016
1.	1.	1.	Neo4j	Graph DBMS	37.87	+1.73	+3.84
2.	↑ 4.	↑ 4.	Microsoft Azure Cosmos DB	Multi-model	6.38	+1.54	+4.03
3.	↓ 2.	↓ 2.	OrientDB	Multi-model	5.81	+0.07	-0.11
4.	↓ 3.	↓ 3.	Titan	Graph DBMS	5.26	+0.41	+0.18
5.	5.	↑ 6.	ArangoDB	Multi-model	3.08	+0.13	+1.43
6.	6.	↓ 5.	Virtuoso	Multi-model	2.00	-0.06	-0.12
7.	7.	7.	Giraph	Graph DBMS	1.05	-0.04	+0.15
8.	8.	↑ 9.	AllegroGraph	Multi-model	0.60	+0.01	+0.13
9.	9.	↓ 8.	Stardog	Multi-model	0.54	+0.03	+0.01
10.	10.	↑ 13.	GraphDB	Multi-model	0.52	+0.02	+0.38
11.	11.	↓ 10.	Sqrrl	Multi-model	0.47	+0.01	+0.17
12.			TinkerGraph	Graph DBMS	0.34		
13.	13.	↓ 11.	InfiniteGraph	Graph DBMS	0.30	+0.02	+0.10
14.	↓ 12.		Dgraph	Graph DBMS	0.29	+0.00	
15.	↓ 14.	↑ 19.	Blazegraph	Multi-model	0.29	+0.01	+0.20
16.	16.		Graph Engine	Multi-model	0.26	+0.08	
17.	↓ 15.	↓ 14.	FlockDB	Graph DBMS	0.19	+0.00	+0.06

Neo4J 简介

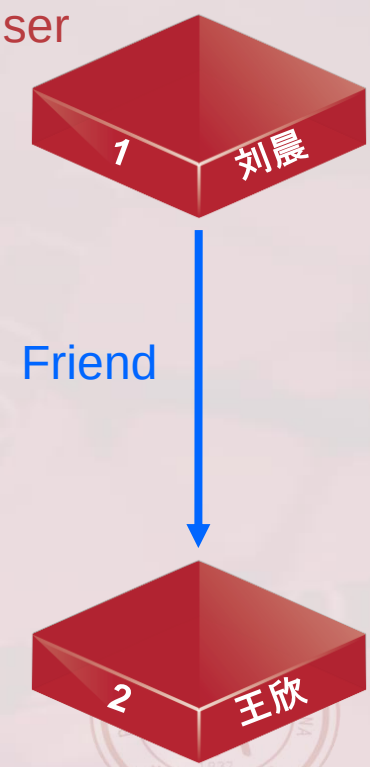
- 查询处理引擎
  - 基于十字链表的遍历
- 存储系统
  - 图模型
- 查询语言：
  - 类似于SQL，Neo4J的查询语言是语法接近于SQL的Cypher

Cypher ： Neo4J的查询语言

- 简单地说，Cypher的工作原理就是通过基于图的模式匹配的方式找到满足条件的子图
- Cypher基本语法结构
  - CREATE: 创建数据对象
  - START：通过顶点的ID或者属性的值，找到图中的起始点
  - MATCH：图的模式匹配
  - WHERE：过滤条件
  - RETURN：返回所需要的信息
  - 注意：有些关键字，比如START、WHERE不是必选的

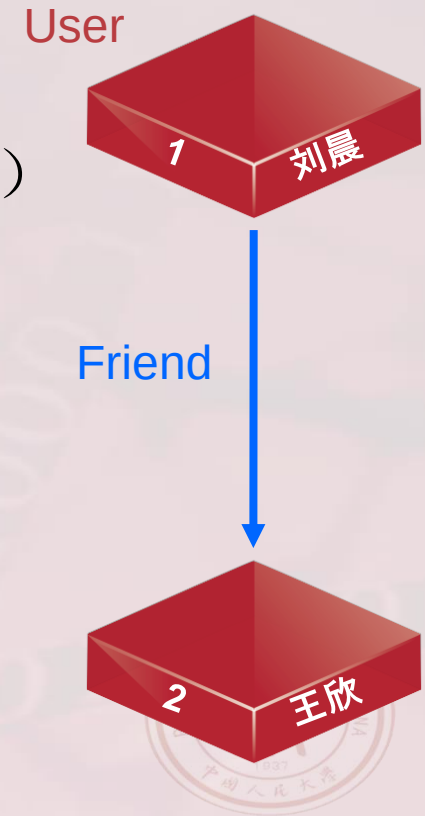
Cypher ： Create语句

- 创建顶点刘晨
  - create(:person {name:“刘晨”,ID:1}) ;
- 一次创建多个顶点
  - create(:person {name:“刘晨”,ID:1}) , (:person {name:“王欣”,ID:2})
- 创建顶点刘晨、边friend、顶点王欣
  - 方法1：create (:person {name:“刘晨”,ID:1}) –[:friend]-> (:person {name:“王欣”,ID:2})
  - 方法2：
    - create(l:person {name:“刘晨”,ID:1}) ;
    - create(r:person {name:“王欣”,ID:2}) ;
    - Create l-[:friend]->r



Cypher： Create语句（续）

- 创建顶点刘晨、边friend、顶点王欣
  - 方法3（假设图中顶点刘晨和王欣已经被创建）
    - 找到刘晨顶点： Match(l:User(ID:1))
    - 找到王欣顶点： Match(r:User(ID:2))
    - 创建边friend：Create l-[:friend]->r

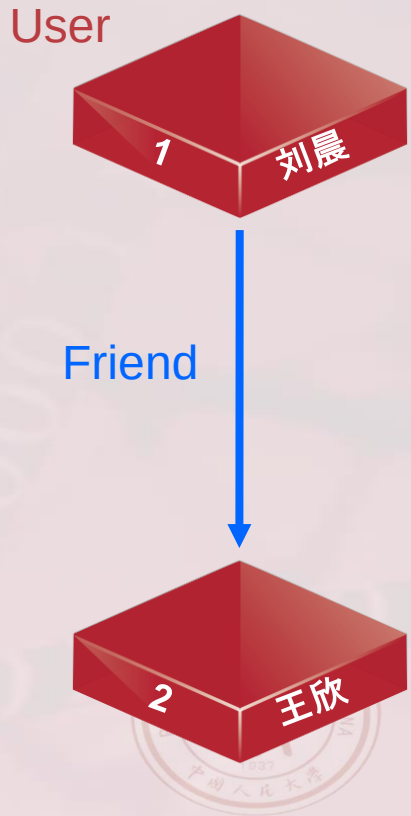


Cypher： Create语句（续）

- 创建面向类别属性的索引
  - Create index on :user(Name)
- 创建面向类别属性的复合索引
  - Create index on :user(Name,Affiliation)
- 创建完整性约束
  - CREATE CONSTRAINT ON (C:User) ASSERT C.ID IS UNIQUE

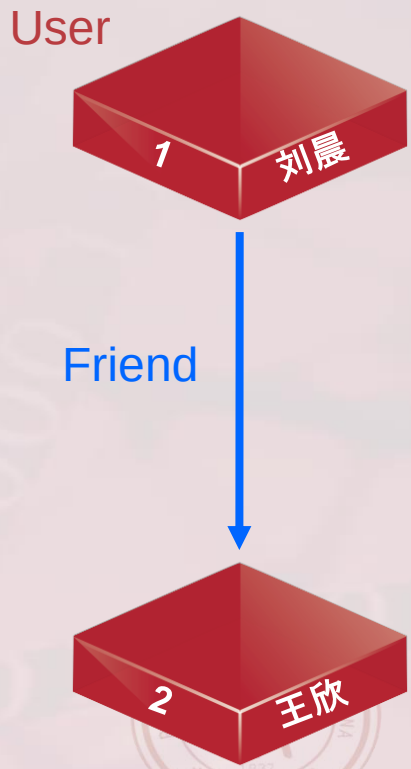
Cypher：查询

- 顶点查询
  - 通过编号查找顶点
    - START L = NODE(0)
    - RETURN L
  - 通过属性：ID查找顶点
    - MATCH (L:User(ID:1))
    - RETURN L
  - 通过WHERE条件
    - match (L:User)
    - where ID=1
    - return L



Cypher： Neo4J的查询语言（续）

- 边查询
  - 方法1
    - **START L =** NODE(0)
    - **START R =** NODE(1)
    - **MATCH(L-[e:friend]->R)**
  - 方法2
    - **MATCH (L:User(Name:‘刘晨’)-[e:friend]->R:User(Name:’王欣’))**
    - **RETURN e**



Cypher ： Neo4J的查询语言（续）

- 路径查询
  - 查找对刘晨发的帖子点赞过的用户的信息
  - **MATCH(:User(Name:’刘晨’))-[:Post]->(:Message)<-[:Likes]-(L:User)**
  - **Return L**

小结

- 谁在用图数据库


Walmart
ebay
hp
pitney bowes
53
cisco
CGI
dun & bradstreet
ALERT LOGIC
LinkedIn
Medium
Our Customers
CenturyLink
CAMBRIDGE UNIVERSITY PRESS
Cerved