# 第9章 知识图谱的存储与检索

王 泉

中国科学院信息工程研究所
中国科学院大学网络空间安全学院

# 提纲

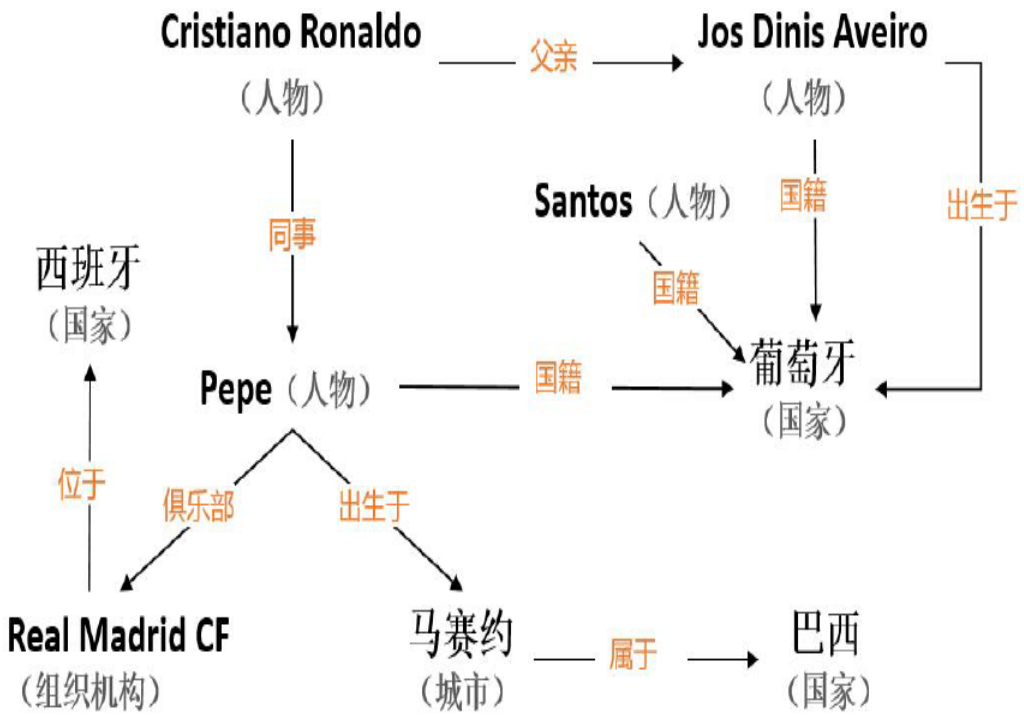
- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 知识图谱

- 知识图谱是一种有向图结构，描述了现实世界中存在的实体、事件或者概念以及它们之间的相关关系。



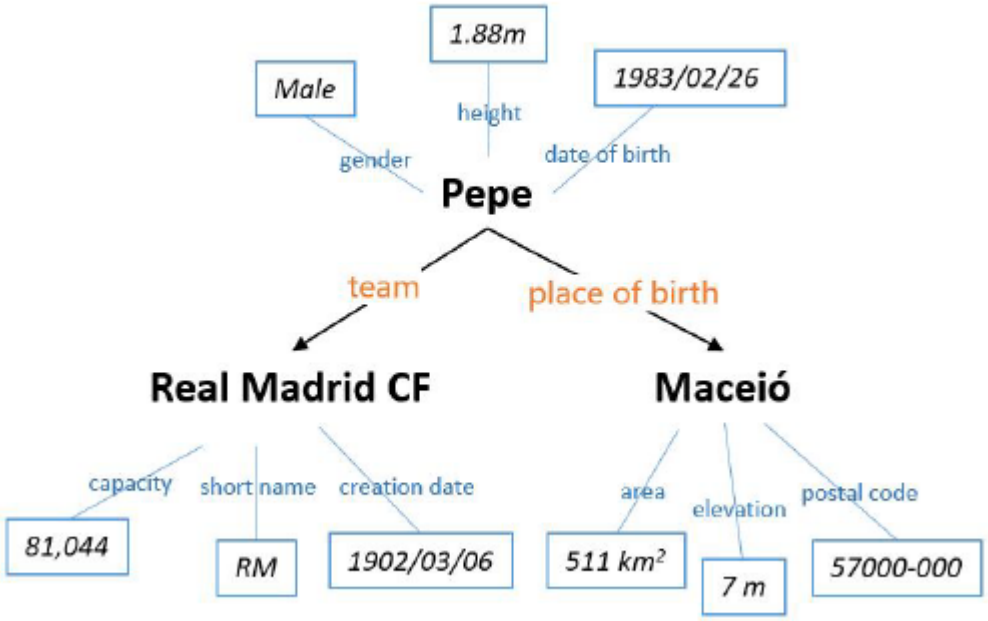

- **实体：**Cristiano Ronaldo, Jos Dinis Aveiro, Real Madrid CF、马赛约等
- **实体类型：**人物、国家、城市、组织机构等
- **属性：**人物有姓名、性别、出生日期、兴趣爱好、职业等属性；国家有国庆日、国家代码、货币、时区、等属性
- **关系：**人物和人物间的同事关系、人物和国家间的国籍关系、城市和国家间的属于关系等

# 知识图谱中的知识表示

- 知识图谱中的知识是通过RDF的结构进行表示的，其基本构成单元是事实，每个事实被表示为一个形如<subject, predicate, object>的三元组。
  - subject: 主体（也称主语），其取值通常是实体、事件或者概念中的任何一个
  - predicate：谓词（也称谓语），其取值通常是关系或者属性
  - object：客体（也称宾语），其取值既可以是实体、事件、概念，也可以是普通的值（如数字、字符串等）

# 知识图谱中的知识表示

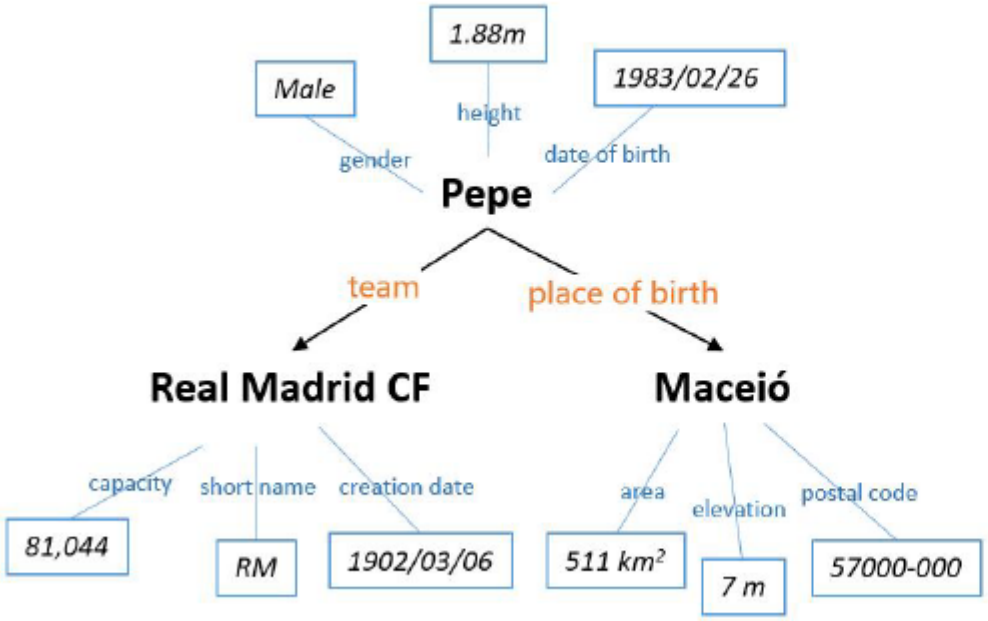
- 知识图谱中的知识是通过RDF的结构进行表示的，其基本构成单元是事实，每个事实被表示为一个形如<subject, predicate, object>的三元组。




- **实体：**Pepe, Real Madrid CF、Maceio
- **属性：**gender, height, date of birth, capacity, short name, creation date, area, elevation, postal code
- **属性值：**male, 1.88m, 1983/02/26, 81,044, RM, 1902/03/06, 511km², 7m, 57000-000
- **关系：**team, place of birth

# 知识图谱中的知识表示

- 知识图谱中的知识是通过RDF的结构进行表示的，其基本构成单元是事实，每个事实被表示为一个形如<subject, predicate, object>的三元组。




**RDF Data**

| <S, | P, | O> |
|---|---|---|
| <Pepe, | gender, | male> |
| <Pepe, | height, | 1.88m> |
| <Pepe, | date of birth, | 1983/02/06> |
| <Pepe, | team, | Ream Madrid CF> |
| <Pepe, | place of birth, | Maceió> |
| <Maceió, | area, | 511km$^2$> |
| <Maceió, | elevation, | 7m> |
| <Maceió, | postal code, | 57000-000> |
| . . . | | |
| . . . | | |

知识图谱的目标是构建一个能够刻画现实世界的知识库，为自动问答、信息检索等应用提供支撑。因此，对知识的持久化存储并提供对目标知识的高效检索是合格的知识图谱必须具备的基本功能。

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 知识图谱的存储

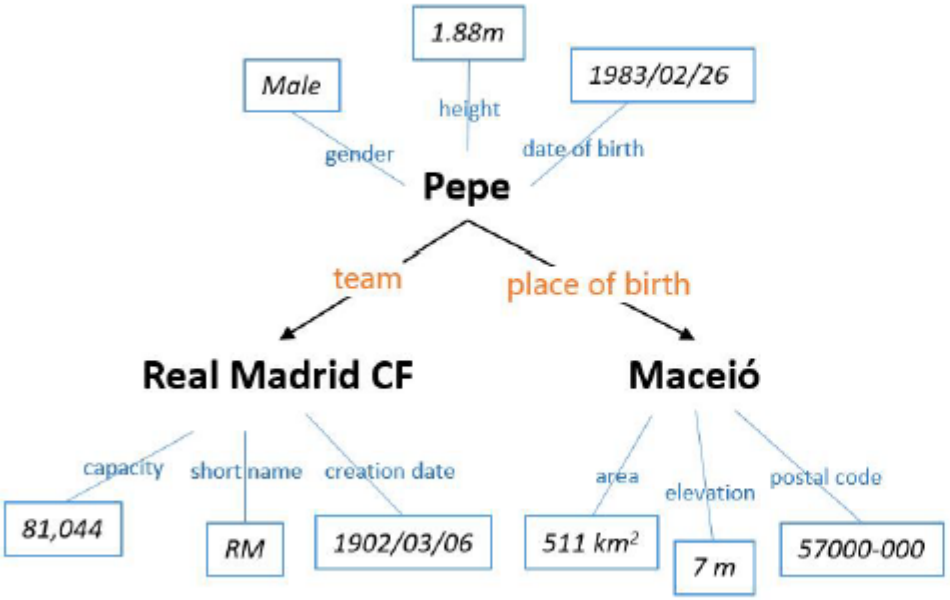
- 按照存储方式的不同，知识图谱的存储可以分为基于表结构的存储和基于图结构的存储。
  - 基于表结构的存储：利用二维的数据表对知识图谱中的数据进行存储
    - 三元组表、类型表、关系数据库
  - 基于图结构的存储：利用图的方式对知识图谱中的数据进行存储
    - 图数据库

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 三元组表

- 知识图谱中的事实是一个个的三元组，一种最简单直接的存储方式是设计一张三元组表用于存储知识图谱中所有的事实。




**三元组表**

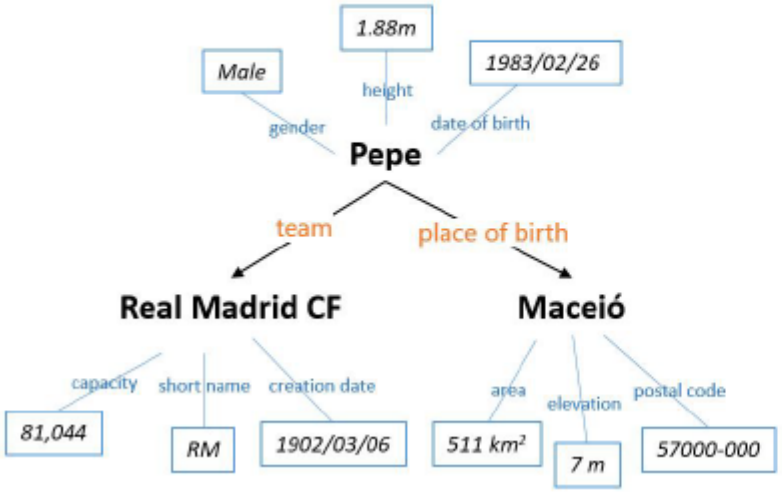
| S. | P. | O. |
|---|---|---|
| Pepe | gender | male |
| Pepe | height | 1.88m |
| Pepe | date of birth | 1983/02/06 |
| Pepe | team | Ream Madrid CF |
| Pepe | place of birth | Maceió |
| Maceió | area | 511km$^2$ |
| Maceió | elevation | 7m |
| Maceió | postal code | 57000-000 |
| . . . | . . . | . . . |

# 三元组表特性

- 基于三元组表的存储方式的优点是简单直接，易于理解；然而缺点也非常明显，主要有以下两点：
  - 整个知识图谱都存储在一张表中，导致单表的规模太大。对大表进行查询、插入、删除、修改等操作的开销很大，这将导致知识图谱的实用性大打折扣。
  - 复杂查询在这种存储结构上的开销巨大。由于数据表只包括三个字段，因此复杂的查询只能拆分成若干简单查询的复合操作，大大降低了查询的效率。例如，查询“佩佩的身高和性别是什么？”需要拆分为“佩佩的身高是多少？”和“佩佩的性别是什么？”

# 类型表

- 为每种类型构建一张表，同一类型的实例存放在相同的表中。表的每一列表示该类实体的一个属性，每一行存储该类实体的一个实例。




**类型表**

City Table

| subject | area | elevation | postal code |
|---|---|---|---|
| Maceió | 511km² | 7m | 57000-000 |

Person Table

| subject | gender | height | date of birth | team | place of birth |
|---|---|---|---|---|---|
| Pepe | male | 1.88m | 1983/02/06 | Ream Madrid CF | Maceió |

# 类型表的不足

- 大量数据字段的冗余存储。假设知识图谱中既有“演员”也有“歌手”，那么同属于这两个类别的实例将会同时被存储在这两个表中，其中它们共有的属性会被重复存储。

Actor Table

| subject | 性别 | 出生日期 | 出生地 | 艺名 | 经纪公司 |
|---|---|---|---|---|---|
| 刘德华 | 男 | 1961/09/27 | 香港 | 刘德华 | 映艺娱乐 |

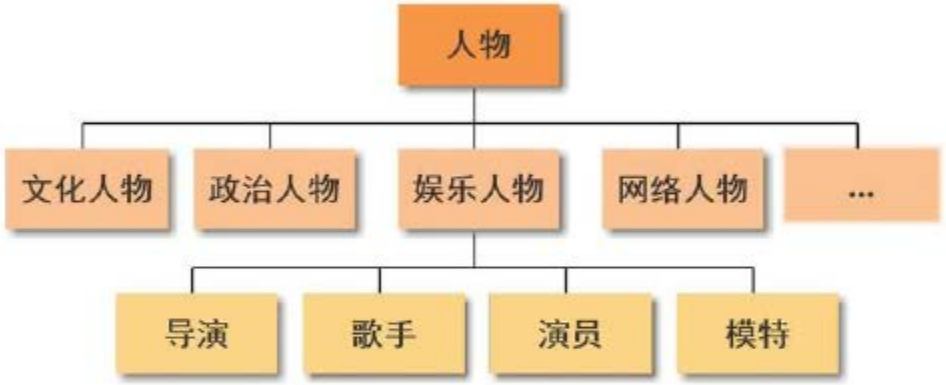
Singer Table

| subject | 性别 | 出生日期 | 出生地 | 艺名 | 经纪公司 | 唱片公司 |
|---|---|---|---|---|---|---|
| 刘德华 | 男 | 1961/09/27 | 香港 | 刘德华 | 映艺娱乐 | 东亚唱片 |

- 大量的数据列为空值。通常知识图谱中并非每个实体在所有属性或关系上都有值，这种存储方式会导致表中存在大量的空值。

# 考虑层级关系的类型表

- 构建数据表时，将知识图谱的类别体系考虑进来。具体来说，每个类型的数据表只记录属于该类型的特有属性，不同类别的公共属性保存在上一级类型对应的数据表中，下级表继承上级表的所有属性。




Person (人物) Table

| subject | 性别 | 出生日期 | 出生地 |
|---|---|---|---|
| 刘德华 | 男 | 1961/09/27 | 香港 |

Entertainer (娱乐人物) Table

| subject | 艺名 | 经纪公司 |
|---|---|---|
| 刘德华 | 刘德华 | 映艺娱乐 |

Actor (演员) Table

| subject |
|---|
| 刘德华 |

Singer (歌手) Table

| subject | 唱片公司 |
|---|---|
| 刘德华 | 东亚唱片 |

# 类型表特性

- 类型表克服了三元组表面临的单表过大和结构简单的问题，但是也有明显的不足之处：
  - 由于数据表是和具体类型对应的，不同的数据表具有不同的结构，因此在查询之前必须知道目标对象的类型才能确定查找的数据表。
  - 当查询涉及到不同类型的实体时，需要进行多表的连接，这一操作开销巨大，限制了知识图谱对复杂查询的处理能力。
  - 知识图谱通常包含丰富的实体类型，因此需要创建大量的数据表，并且这些数据表之间又具有复杂的关系，这为数据的管理增加了很大的难度。

# 关系数据库基本概念

- 关系数据库以二维表结构对数据进行组织和存储。
  - 属性 (attribute)：表中的每一列称为一个属性（也称字段），用来描述实体集的某个特征。每个属性都有自己的取值范围，称为域。
  - 元组 (tuple)：表中的每一行由一个实体的相关属性取值构成，称为元组（也称记录），它相对完整地描述了一个实体。元组中的一个属性值称为分量。

| 学号 | 姓名 | 性别 | 出生日期 | 身份证号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 31010319950922317 |
| 1202 | 张红 | 女 | 1995-11-07 | 41010419951107243 7 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 51010519960408222 3 |

# 关系数据库基本概念

- 上述二维表格有以下限制：
  - 每一属性必须是基本的、不能再拆分的数据类型，如整型、实型、字符型等。结构或数组不能作为属性的类型。

# 关系数据库基本概念

- 上述二维表格有以下限制：
  - 每一属性必须是基本的、不能再拆分的数据类型，如整型、实型、字符型等。结构或数组不能作为属性的类型。

| 姓名 | 籍贯 | |
|---|---|---|
| | 省 | 市 |
| 李强 | 黑龙江 | 齐齐哈尔 |
| 张红 | 浙江 | 杭州 |
| 王鹏 | 江西 | 南昌 |

| 姓名 | 省 | 市 |
|---|---|---|
| 李强 | 黑龙江 | 齐齐哈尔 |
| 张红 | 浙江 | 杭州 |
| 王鹏 | 江西 | 南昌 |

# 关系数据库基本概念

- 上述二维表格有以下限制：
  - 每一属性必须是基本的、不能再拆分的数据类型，如整型、实型、字符型等。结构或数组不能作为属性的类型。
  - 属性的所有值必须是同类型、同语义的。如果某一列包含学生的学号，则该表中所有行的此列都必须是学生的学号。
  - 属性的值只能是域中的值。例如学生的性别只能在“男”或“女”中取值。
  - 属性必须有唯一的名字，但不同的属性可以出自相同的域。列在表中的顺序可以任意交换。

# 关系数据库基本概念

- 上述二维表格有以下限制：
  - 每一属性必须是基本的、不能再拆分的数据类型，如整型、实型、字符型等。结构或数组不能作为属性的类型。
  - 属性的所有值必须是同类型、同语义的。如果某一列包含学生的学号，则该表中所有行的此列都必须是学生的学号。
  - 属性的值只能是域中的值。例如学生的性别只能在“男”或“女”中取值。
  - 属性必须有唯一的名字，但不同的属性可以出自相同的域。列在表中的顺序可以任意交换。

| 学号 | 姓名 | 性别 | 出生日期 | 入学日期 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 2017-09-01 |
| 1202 | 张红 | 女 | 1995-11-07 | 2017-09-01 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 2017-09-01 |

# 关系数据库基本概念

- 上述二维表格有以下限制：
  - 每一属性必须是基本的、不能再拆分的数据类型，如整型、实型、字符型等。结构或数组不能作为属性的类型。
  - 属性的所有值必须是同类型、同语义的。如果某一列包含学生的学号，则该表中所有行的此列都必须是学生的学号。
  - 属性的值只能是域中的值。例如学生的性别只能在“男”或“女”中取值。
  - 属性必须有唯一的名字，但不同的属性可以出自相同的域。列在表中的顺序可以任意交换。
  - 任意两个元组的值不能完全相同，即不允许有重复的行。行在表中的顺序可以任意交换。

# 关系数据库属性分类

- 候选码：能够唯一标识元组的最小的属性集合。
  - 唯一性：候选码在整个表的范围内必须具有唯一的值，不同元组不能具有相同的候选码值。
  - 最小性：候选码所包括的属性必须都是必不可少的，缺少其中的任何一个都不再具备唯一性。

| 学号 | 姓名 | 性别 | 出生日期 | 身份证号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 31010319950922317 |
| 1202 | 张红 | 女 | 1995-11-07 | 41010419951107243 7 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 51010519960408222 3 |

  - 例如，在上表中学生的学号、身份证号分别都可以作为候选码；但是（学号, 身份证号）这一属性组不能作为候选码，因为去掉其中任何一个剩下的属性仍然可以作为候选码，因此不满足最小性。

# 关系数据库属性分类

- 主码：一个数据表可以包含多个候选码，从中任意选取一个作为主码。虽然理论上所有的候选码都可以作为主码，但是实际操作中一般选择单属性组成的候选码作为主码。

| 学号 | 姓名 | 性别 | 出生日期 | 身份证号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 31010319950922317 |
| 1202 | 张红 | 女 | 1995-11-07 | 41010419951072437 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 51010519960408223 |

  - 在该表中有三个候选码：学号、身份证号、姓名（不重名）。
  - 姓名作主码还是有些限制（不能重名）；
  - 身份证号太长，使用不方便；
  - 学号简洁，使用方便，在实际生活中也被用来唯一区分学生，所以选学号作主码最合适。

# 关系数据库属性分类

- 外码：如果数据表中的某个属性或属性组是其它表的候选码，那么称该属性或属性组是当前表的外码。外码能够保证不同数据表之间数据的一致性。

学生表（外表）

| 学号 | 姓名 | 性别 | 出生日期 | 班号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 11 |
| 1202 | 张红 | 女 | 1995-11-07 | 12 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 13 |

班级表（主表）

| 班号 | 班名 | 专业 |
|---|---|---|
| 11 | 数学01 | 应用数学 |
| 12 | 物理01 | 理论物理 |
| 13 | 化学01 | 材料化学 |

  - 班号是班级表的候选码，所以班号是学生表的外码。

- 主属性与非主属性：包含在任何候选码中的属性称为主属性；相反地，不包含在任何候选码中的属性称为非主属性。

# 关系数据库完整性约束

- 关系数据库通过关系完整性约束条件来保证数据的正确性和一致性。所谓关系完整性约束条件主要是指在表和属性（列）上定义的规则，数据库管理系统会依据这些约束条件维护数据完整性。
  - 域完整性规则
  - 实体完整性规则
  - 参照完整性规则

# 关系数据库完整性约束

- 域完整性：在关系数据模型定义时，由用户对属性值的数据类型、长度、单位、精度、格式、值域范围、是否允许为空值等进行限定，规定属性取值必须在其值域中。

| 学号 | 类型定义 | 属性限定 |
|---|---|---|
| 学号 | char(4) | Primary Key |
| 姓名 | nvarchar(20) | Not Null |
| 性别 | nchar(1) | Not Null, 男/女 |
| 出生日期 | datetime | >=1900-01-01 |
| 班号 | char(2) | Not Null |

| 学号 | 姓名 | 性别 | 出生日期 | 班号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 11 |
| 1202 | 张红 | 女 | 1995-11-07 | 12 |
| 1303 | 王鹏 | 21 | 1996-04-08 | 13 |

# 关系数据库完整性约束

- 实体完整性：元组（记录）在构成主码的属性上不能有空值且主码的值不能相同。实体完整性主要是为了确保主码能唯一标识元组。

| 学号 | 姓名 | 性别 | 出生日期 | 班号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 11 |
| 1202 | 张红 | 女 | 1995-11-07 | 12 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 13 |
| √1102 | 孙丽 | 女 | 1996-01-29 | 11 |
| X1202 | 赵刚 | 男 | 1995-10-20 | 12 |

# 关系数据库完整性约束

- 参照完整性：一个外表的外码取值必须是其主表主码的存在值或空值。

学生表（外表）

| 学号 | 姓名 | 性别 | 出生日期 | 班号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 11 |
| 1202 | 张红 | 女 | 1995-11-07 | 12 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 13 |

班级表（主表）

| 班号 | 班名 | 专业 |
|---|---|---|
| 11 | 数学01 | 应用数学 |
| 12 | 物理01 | 理论物理 |
| 13 | 化学01 | 材料化学 |

  - 学生表中班号的取值必须是空值或在班级表中班号属性中出现过的值。如果学生表的班号字段允许为空，它取空值表示该学生还没有确定班级；否则只能取班级表出现过的班号值。

# 关系数据库操作语言

- 关系数据库的基本操作语言是SQL (Structured Query Language)。
- SQL语言以简洁的语法支持关系数据库的各类操作（插入/修改/删除/查询）。
- SQL语言独立于关系数据库本身，独立于使用的机器、网络和操作系统。

如果选择关系数据库作为知识图谱的存储引擎，那么对知识图谱的所有操作都需要转换为SQL语句才能执行。

# 关系数据库基本操作

- 关系数据库通过SQL语言为用户提一系列的操作接口，其核心功能包括插入、修改、删除、查询四种操作。
  - 插入 (INSERT)：在一个表中插入一条或多条新记录。
  - 修改 (UPDATE)：在一个表中修改满足条件的记录的某些字段的值。
  - 删除 (DELETE)：从一个表中删除一条或多条满足条件的记录。
  - 查询 (SELECT)：从一个或多个表中提取满足条件的数据、生成计算列或汇总数据。

# 常见的关系数据库存储系统

- DB2：DB2是IBM公司于1983年推出的数据库管理系统，能够运行于各种不同的操作系统平台上，如Windows、Linux、VMS、OS/2等。该数据库的特色之处包括支持面向对象的编程、支持多媒体应用、支持分布式数据库的访问等。更多信息请参考DB2官网：https://www.ibm.com/analytics/us/en/technology/db2/
- Oracle：Oracle是甲骨文公司于1979年推出的关系数据库管理系统，是目前最流行的数据库之一。该数据库遵循标准SQL语言，支持多种数据类型。和DB2相同，Oracle同样具有跨平台的特点，可以运行于不同的操作系统之上。更多信息请参考Oracle官网：https://www.oracle.com/database/index.html

# 常见的关系数据库存储系统

- Microsoft SQL Server：Microsoft SQL Server是微软公司推出的应用于Windows操作系统的数据库产品，该数据库实际上是微软基于Sybase公司提供的技术开发而成的，因此完全兼容Sybase数据库。更多信息请参考Microsoft SQL Server官网：https://www.microsoft.com/zh-cn/sql-server/sql-server-2016
- PostgreSQL：PostgreSQL是一个自由软件系统，和上面几个数据库系统相比，该系统具有开源、免费的特点。由于PostgreSQL可以免费地被使用、修改和分发，因此其具有非常完善的功能，不仅支持大部分的SQL标准，还提供外键、复杂查询、触发器等特性。更多信息请参考PostgreSQL官网：https://www.postgresql.org/

# 常见的关系数据库存储系统

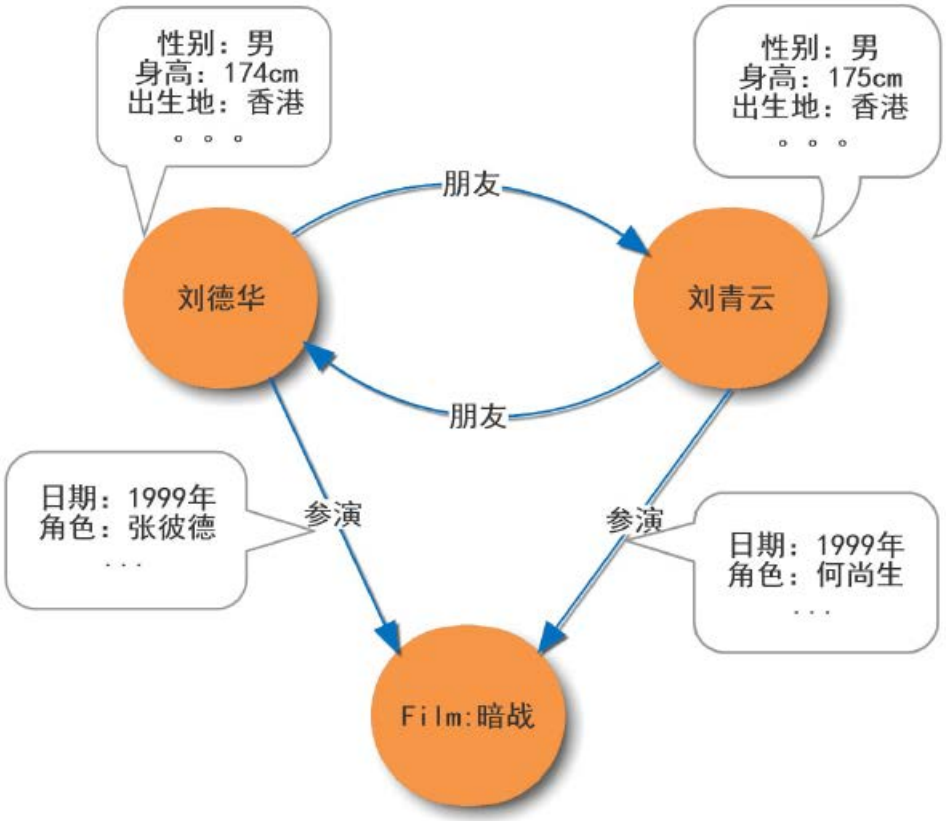
- MySQL：MySQL 也是一个自由软件系统，最早是由瑞典的MySQL AB公司开发的小型关系数据库管理系统，在2008年被Sun公司收购。该系统由于具有体积小、速度快、跨平台、免费开源等特点，是中小型网站最为青睐的数据库。更多信息请参考MySQL官网：https://www.mysql.com/

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 基于图结构的存储模型

- 将实体看做节点，关系看做带有标签的边，那么知识图谱的数据很自然地满足图模型结构。




- **用节点表示实体：**刘德华、刘青云、Film：暗战
- **用边表示实体间的关系：**刘德华和暗战之间的参演关系、刘德华和刘青云之间的朋友关系等
- **节点可以定义属性：**刘德华性别男、身高174cm、出生地香港等
- **边上也可以定义属性：**刘德华参演暗战的时间是1999年，参演角色是张彼得等
- **无向关系需要转化为两条对称的有向关系：**刘德华和刘青云之间互为朋友关系

# 图数据库核心概念

- 图数据库基于有向图，其理论基础是图论。节点、边和属性是图数据库的核心概念。
  - 节点：节点用于表示实体、事件等对象，可以类比于关系数据库中的记录或数据表中的行数据。例如人物、地点、电影等都可以作为图中的节点。
  - 边：边是指图中连接节点的有向线条，用于表示不同节点之间的关系。例如人物节点之间的夫妻关系、同事关系等都可以作为图中的边。
  - 属性：属性用于描述节点或者边的特性。例如人物的姓名、夫妻关系的起止时间等都是属性。

# 常见的图数据库存储系统

- Neo4j：Neo4j是一个开源的图数据库系统，它将结构化的数据存储在图上而不是表中。Neo4j基于Java实现，它是一个具备完全事务特性的高性能数据库，具有成熟数据库的所有特性。Neo4j是一个本地数据库（又称基于文件的数据库），这意味着不需要启动数据库服务器，应用程序不用通过网络访问数据库服务，而是直接在本地对其进行操作，因此访问速度快。因其开源、高性能、轻量级等优势，Neo4j 受到越来越多的关注。更多信息请参考官网：https://neo4j.com/

# 常见的图数据库存储系统

- OrientDB：OrientDB是一个开源的文档-图混合数据库，兼具图数据库对数据强大的表示及组织能力和文档数据库的灵活性及很好的可扩展性。OrientDB具有多种模式可选，包括全模式、无模式和混合模式。全模式要求数据库中的所有类别都必须有严格的模式，所有字段都强制约束；无模式则相反，不需要为类别定义模式，存储的数据记录可以有任意的字段；混合模式则允许为类别定义若干字段，同时支持自定义字段。该数据库同样是本地的，支持许多数据库的高级特性，如事务、快速索引、SQL查询等。更多信息请参考官网：http://orientdb.com/

# 常见的图数据库存储系统

- HyperGraphDB：HyperGraphDB同样是开源的存储系统，并依托于BerkeleyDB数据库。和上述图数据库相比，HyperGraphDB最大的特点在于HyperGraph，即超图。从数学角度讲，有向图的一条边只能指向一个节点，而超图则可以指向多个节点。实际上，HyperGraphDB在超图的基础上更进了一步，不仅允许一条边指向多个节点，还允许其指向其它边。如此以来，HyperGraphDB相较于其它图数据库具有更强大的表示能力。更多信息请参考官网：http://www.hypergraphdb.org/

# 常见的图数据库存储系统

- InfiniteGraph：InfiniteGraph是一个基于Java语言开发的分布式图数据库系统。和MySQL等传统关系数据库类似，InfiniteGraph需要作为服务项目进行安装，应用程序只能通过访问数据库服务对数据库进行操作。InfiniteGraph借鉴了面向对象的概念，将图中的每个节点及每条边都看做一个对象。具体地，所有的节点都继承自基类BaseVertex，所有的边则都继承自基类BaseEdge。更多信息请参考官网：http://www.objectivity.com/products/infinitegraph/

# 常见的图数据库存储系统

- InfoGrid：InfoGrid是一个开源的互联网图数据库，提供了很多额外的组件，可以很方便地构建基于图结构的网络应用。InfoGrid实际上是一个基于Java语言的开源项目集，其中InfoGrid图数据库项目是其核心，其它的项目包括InfoGrid存储项目、InfoGrid用户接口项目等。更多信息请参考官网：http://infogrid.org

# 常见的图数据库存储系统

- InfoGrid：InfoGrid是一个开源的互联网图数据库，提供了很多额外的组件，可以很方便地构建基于图结构的网络应用。InfoGrid实际上是一个基于Java语言的开源项目集，其中InfoGrid图数据库项目是其核心，其它的项目包括InfoGrid存储项目、InfoGrid用户接口项目等。更多信息请参考官网：http://infogrid.org

和成熟的关系数据库相比，图数据库的发展较晚，相关的标准及技术尚不完善，在实际使用中可能会遇到一些棘手的问题，因此在选用数据库时除了需要考虑数据库本身的特性、性能等因素以外，还需要考虑数据库的易用性、技术文档的完整性等因素。

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 知识图谱的检索

- 知识图谱的知识实际上是通过数据库系统进行存储的，大部分数据库系统通过形式化的查询语言为用户提供访问数据的接口。
  - 关系数据库：标准查询语言SQL
  - 图数据库：标准查询语言SPARQL
- 由于知识图谱中的数据在逻辑上本身就是一种图结构，因此也可以通过图查询技术来完成特定查询图的查找，其核心问题是判断查询图是否是图数据集的子图，因此也叫子图匹配问题。

  《A Performance Comparison of Five Algorithms for Graph Isomorphism》

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# SQL的产生与发展

- SQL（Structured Query Language，结构化查询语言）是介于关系代数和元组演算之间的一种语言，用于管理关系数据库，目前已经成为关系数据库的标准语言。
  - 1972年，IBM公司开始研制实验型关系数据库管理系统SYSTEM R，其配备的查询语言称为SQUARE（Specifying Queries As Relational Expression）语言，语言中使用了较多的数学符号。
  - 1974年，Boyce和Chamberlin把SQUARE修改为SEQUEL (Structured English Query Language) 语言。后来SEQUEL简称为SQL (Structured Query Language)，即结构化查询语言，SQL的发音仍是“sequel”。现在SQL已经成为一个标准。

# SQL的组成

- SQL语言从功能上可以分为四个部分：数据查询、数据操纵、数据定义和数据控制。
  - 数据查询语言，即SQL DQL，用来对已存在的数据库中的数据按照指定的组合、条件表达式或排序进行检索。
  - 数据操纵语言，即SQL DML，用来改变数据库中的数据，分为插入、修改、删除三种操作。
  - 数据定义语言，即SQL DDL，用来创建数据库中的各种对象，包括数据库模式、表、视图、索引、同义词、聚簇等。
  - 数据控制语言，即SQL DCL，用来授予或回收访问数据库的某种特权，控制数据操纵事务的发生时间及效果、对数据库进行监视等。

# SQL的组成

- SQL语言功能极强，但由于设计巧妙，语言十分简洁，完成数据定义、数据操纵、数据控制、数据查询的核心功能只用了9个动词。并且SQL语言语法简单，接近英语口语，因此易学易用。

| 功　能 | 动　词 |
| --- | --- |
| 数据查询 | SELECT |
| 数据操纵 | INSERT, UPDATE, DELETE |
| 数据定义 | CREATE, DROP, ALTER |
| 数据控制 | GRANT, REVOKE |

# 数据查询

- 数据查询是指从数据表中选取满足条件的数据，查询结果存储在一个临时的结果表中。SQL通过SELECT语句完成该功能，其基本语法为：

  SELECT 目标表的列名或列表达式序列
  FROM 基本表名和（或）视图序列
  [ WHERE 行条件表达式 ]
  [ GROUP BY 列名序列 ]
  [ HAVING 组条件表达式 ]
  [ ORDER BY 列名 [ ASC | DESC ], … ]

  - 主语句**SELECT-FROM-WHERE**的含义是：根据WHERE子句的条件表达式，从FROM子句指定的基本表或视图中找出满足条件的元组，再按SELECT子句中的目标列表达式，选出元组中的属性值形成结果表。

# 数据查询

- 完整的**SELECT**语句执行过程如下：
  1. 读取FROM子句中基本表、视图的数据；
  2. 选取满足WHERE子句中给出的条件表达式的元组；
  3. 按GROUP BY子句中指定列的值分组，同时提取满足HAVING子句中组条件表达式的那些组；
  4. 按SELECT子句中给出的列名或列表达式求值输出；
  5. ORDER BY子句对输出的目标表进行排序，按附加说明ASC升序排列，或按DESC降序排列。

# 单表查询

- 查询仅涉及一个表，是一种最简单的查询操作：
  - 选择表中的若干列
  - 选择表中的若干元组
  - 对查询结果排序
  - 使用聚合函数
  - 对查询结果分组

# 查询表中的若干列

- 查询指定列：在很多情况下，用户只对表中的一部分属性感兴趣，这时可以通过在SELECT子句的<目标列表达式>中指定要查询的属性列。

例9.1 查询以下学生表中全体学生的学号与姓名。

```
SELECT number, name
FROM Student;
```

| number | name | sex | age | birthplace |
|---|---|---|---|---|
| 20141001 | 朱晓明 | 男 | 26 | 江苏徐州 |
| 20141002 | 王月华 | 女 | 25 | 山西太原 |
| 20141003 | 刘子华 | 男 | 26 | 四川成都 |

Student

| number | name |
|---|---|
| 20141001 | 朱晓明 |
| 20141002 | 王月华 |
| 20141003 | 刘子华 |

# 查询表中的若干列

- 查询全部列：将表中的所有属性列都选出来，可以有两种方法。一种方法就是在SELECT关键字后面列出所有列名；另一种方法是如果列的显示顺序与其在基表中的顺序相同，也可以简单地将<目标列表达式>指定为“*”。

例9.2 查询Student表中全体学生的详细记录。

```
SELECT *
FROM Student;
```

等价于：

```
SELECT number, name, sex, age, birthplace
FROM Student;
```

# 查询表中的若干列

- 查询经过计算的值：SELECT子句的<目标列表达式>不仅可以是表中的属性列，也可以是表达式。

例9.3 查询Student表中全体学生的姓名及其出生年份。

```
SELECT name, 2017-age
FROM Student;
```

| number | name | sex | age | birthplace |
|---|---|---|---|---|
| 20141001 | 朱晓明 | 男 | 26 | 江苏徐州 |
| 20141002 | 王月华 | 女 | 25 | 山西太原 |
| 20141003 | 刘子华 | 男 | 26 | 四川成都 |

Student

| name | |
|---|---|
| 朱晓明 | 1991 |
| 王月华 | 1992 |
| 刘子华 | 1991 |

# 查询表中的若干元组

- 消除取值重复的行：两个本来并不完全相同的元组，投影到指定的某些列上后，可能变成相同的行了，这时可以用DISTINCT消除它们。

例9.4 查询Student表中全体学生的年龄。

| number | name | sex | age | birthplace |
|---|---|---|---|---|
| 20141001 | 朱晓明 | 男 | 26 | 江苏徐州 |
| 20141002 | 王月华 | 女 | 25 | 山西太原 |
| 20141003 | 刘子华 | 男 | 26 | 四川成都 |

```
SELECT age
FROM Student;
```

| age |
|---|
| 26 |
| 25 |
| 26 |

```
SELECT DISTINCT age
FROM Student;
```

| age |
|---|
| 26 |
| 25 |

# 查询表中的若干元组

- 查询满足条件的元组：查询满足指定条件的元组可以通过WHERE子句实现。

| 查询条件 | 谓　词 |
| --- | --- |
| 比较 | =, >, <, >=, <=, !=, <>, !>, !<, NOT+上述运算符 |
| 确定范围 | BETWEEN AND, NOT BETWEEN AND |
| 确定集合 | IN, NOT IN |
| 字符匹配 | LIKE, NOT LIKE |
| 空值 | IS NULL, IS NOT NULL |
| 逻辑运算 | AND, OR, NOT |

# 查询表中的若干元组

- 查询满足条件的元组：查询满足指定条件的元组可以通过WHERE子句实现。

例9.5 查询Student表中出生于江苏徐州的学生的学号和姓名。

```
SELECT number, name
FROM Student
WHERE birthplace='江苏徐州';
```

| number | name |
|---|---|
| 20141001 | 朱晓明 |

例9.6 查询Student表中年龄大于25岁的学生的学号和出生地。

```
SELECT number, birthplace
FROM Student
WHERE age>25;
```

| number | birthplace |
|---|---|
| 20141001 | 江苏徐州 |
| 20141003 | 四川成都 |

# 对查询结果排序

- 使用ORDER BY子句可以按一个或多个属性列排序。ASC（缺省值）表示按升序排列，DESC表示按降序排列。当排序列含空值时，ASC排序列为空值的元组最后显示，DESC排序列为空值的元组最先显示。

例9.7 查询选修了3号课程的学生的学号及其成绩，查询结果按分数的降序排列。

```
SELECT studentNo, grade
FROM StudentCourse
WHERE courseNo='3'
ORDER BY grade DESC;
```

| studentNo | grade |
|---|---|
| 95010 | |
| 95007 | 92 |
| 95003 | 82 |
| 95010 | 82 |
| 95009 | 75 |
| 95014 | 61 |

# 使用聚合函数

- 聚合函数是涉及整个关系的另一类运算操作。通过聚合函数，可以把某一列中的值形成单个值。
  - 计数：COUNT ( [ DISTINCT | ALL ] * )
    COUNT ( [ DISTINCT | ALL ] <列名> )
  - 计算总和： SUM ( [ DISTINCT | ALL ] <列名> )
  - 计算平均值： AVG ( [ DISTINCT | ALL ] <列名> )
  - 求最大值： MAX ( [ DISTINCT | ALL ] <列名> )
  - 求最小值： MIN ( [ DISTINCT | ALL ] <列名> )

# 使用聚合函数

- 聚合函数是涉及整个关系的另一类运算操作。通过聚合函数，可以把某一列中的值形成单个值。

例9.8 查询选修了课程的学生人数。

```
SELECT COUNT ( DISTINCT studentNo )
FROM StudentCourse;
```

例9.9 计算选修了1号课程的学生的平均成绩。

```
SELECT AVG ( grade )
FROM StudentCourse
WHERE courseNo='1';
```

# 对查询结果分组

- 使用GROUP BY子句分组，细化聚合函数的作用对象：未对查询结果分组，聚合函数将作用于整个查询结果；对查询结果分组后，聚合函数将分别作用于每个组。

例9.10 查询每门课程的选课人数和平均成绩。

```
SELECT courseNo, COUNT (studentNo), AVG (grade)
FROM StudentCourse
GROUP BY courseNo;
```

# 数据操纵

- 数据插入是指向数据表中添加新的数据记录，SQL通过INSERT语句完成该功能。其基本语法为：

  INSERT
  INTO 表名 [(<属性列1> [,<属性列2>, … ])]
  VALUES (<常量1> [,<常量2>, … ])

  - 该语句的功能是将新元组插入指定表中，其中新元组的属性列1的值为常量1，属性列2的值为常量2，依此类推。INTO子句中没有出现的属性列，新元组在这些列上将取空值。如果INTO子句没有指明任何属性列名，则新插入的元组必须在每个属性列上均有值。

# 数据操纵

例9.11 将一个新学生元组 (学号: 20141004; 姓名: 陈东; 性别: 男; 年龄: 24; 出生地: 广东深圳) 插入到Student表中。

```
INSERT
INTO Student
VALUES ('20141004', '陈东', '男', 24, '广东深圳');

INSERT
INTO Student (number, name, sex, age, birthplace)
VALUES ('20141004', '陈东', '男', 24, '广东深圳');
```

| number | name | sex | age | birthplace |
|---|---|---|---|---|
| 20141001 | 朱晓明 | 男 | 26 | 江苏徐州 |
| 20141002 | 王月华 | 女 | 25 | 山西太原 |
| 20141003 | 刘子华 | 男 | 26 | 四川成都 |
| 20141004 | 陈　东 | 男 | 24 | 广东深圳 |

# 数据操纵

- 数据修改是指修改数据表中已有记录某些列的值，SQL通过UPDATE语句完成该功能。其基本语法为：

  UPDATE 表名
  SET <列名>=<表达式> [,<列名>=<表达式>,…]
  [WHERE 行条件表达式]

  - 该语句中SET子句给出<表达式>的值用于取代相应的属性列值。如果省略WHERE子句，则表示要修改表中的所有元组。

# 数据操纵

例9.12 将学号为20141003的学生的年龄改为25岁。

```
UPDATE Student
SET age=25
WHERE number='20141003';
```

例9.13 将所有学生的年龄增加1岁。

```
UPDATE Student
SET age=age+1;
```

# 数据操纵

- 数据删除是指从数据表中删除指定的数据记录，SQL通过DELETE语句完成该功能。其基本语法为：

```
DELETE
FROM 表名
[WHERE 行条件表达式]
```

例9.14 删除学号为20141003的学生记录。

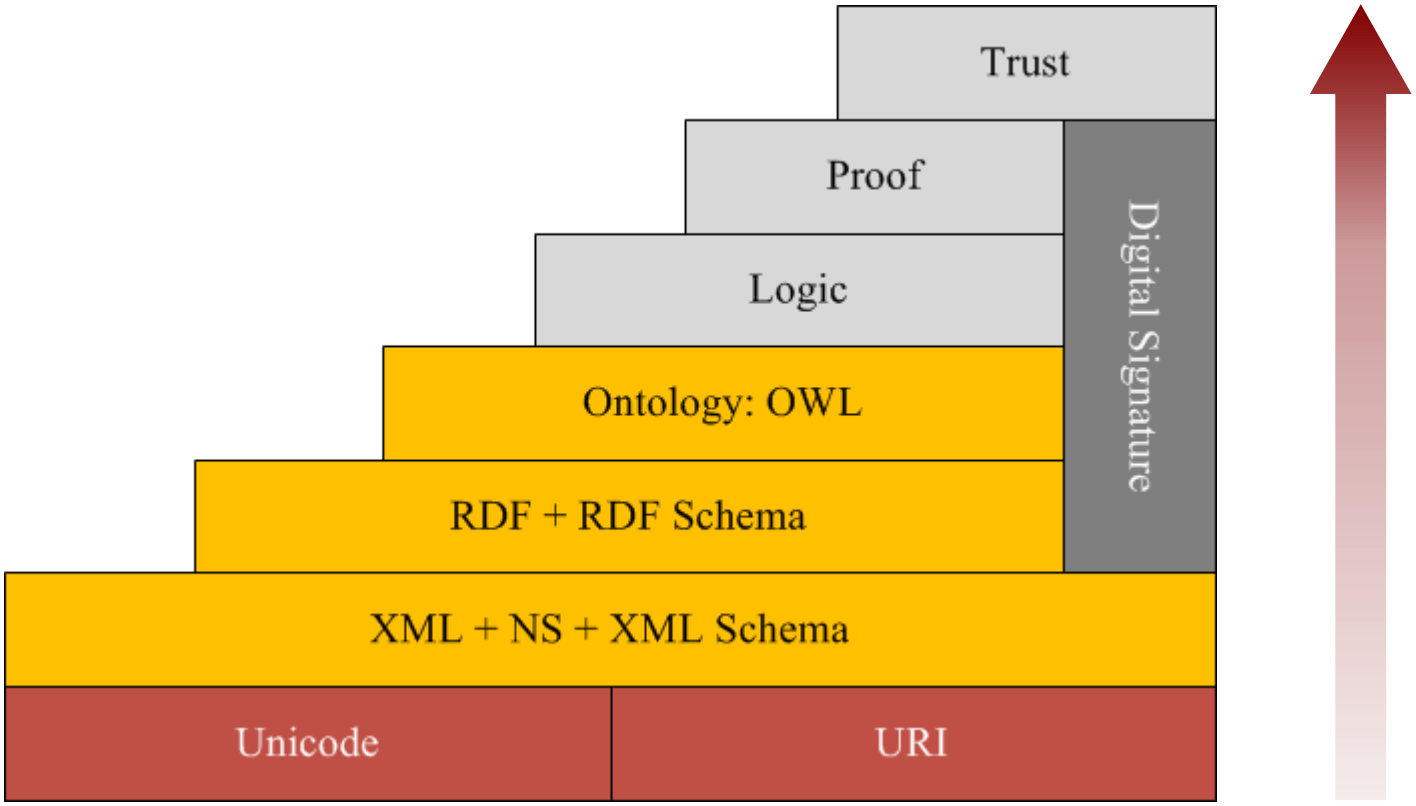
```
DELETE
FROM Student
WHERE number='20141003';
```

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
- 本章小结

# 语义网回顾

- 本质：以Web数据的内容（即语义）为核心，用机器能够理解和处理的方式链接起来的海量分布式数据库。



# RDF回顾

- RDF（Resource Description Framework，资源描述框架）是一种资源描述语言，其核心思想是利用Web标识符（URI）来标识事物，通过指定的属性和相应的值描述资源的性质或资源之间的关系。

```
# Default Graph
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>  ns:director  <http://example/person1> .
```

图1 只包含一个三元组的RDF图示例

# SPARQL语言

- SPARQL是Simple Protocol and RDF Query Language的缩写，是由W3C为RDF数据开发的一种查询语言和数据获取协议，是被图数据库广泛支持的查询语言。和SQL类似，SPARQL也是一种结构化的查询语言，用于对数据的获取与管理。
  - 数据操纵（插入、删除、更新）
  - 数据查询

# 数据操纵

- 数据插入：将新的三元组插入到已有的RDF图中。SPARQL通过INSERT DATA语句完成该功能，其基本语法为：

  INSERT DATA 三元组数据

  - 三元组数据可以是多条三元组，不同的三元组通过“;” 分隔。如果待插入的三元组在RDF图中已经存在，那么系统会忽略该三元组。

例9.15 将以下两条三元组插入到图1所示的RDF图中。

<http://example/movie1> ns:producer <http://example/person2>
<http://example/movie1> ns:writer <http://example/person3>

# 数据操纵

其对应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

INSERT DATA
{
<http://example/movie1>  ns:producer  <http://example/person2>;
                         ns:writer  <http://example/person3> .
}
```

执行该语句后，图1中的RDF图变为：

```
# Default Graph
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>  ns:director <http://example/person1> .
<http://example/movie1>  ns:producer  <http://example/person2>.
<http://example/movie1>  ns:writer <http://example/person3> .
```

图2 SPARQL插入数据结果示例

# 数据操纵

- 数据删除：从RDF图中删除一些三元组。SPARQL通过DELETE DATA语句完成该功能，其基本语法为：

  DELETE DATA 三元组数据

  - 其中三元组数据可以是多条三元组，不同的三元组通过“;” 分隔。对于给定的每个三元组，如果其在RDF图中，则将其从图中删除，否则忽略该三元组。

例9.16 将以下三元组从图2所示的RDF图中删除。

<http://example/movie1> ns:producer <http://example/person2>

# 数据操纵

其对应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

DELETE DATA
{
<http://example/movie1>  ns:producer  <http://example/person2>.
}
```

执行该语句后，图2中的RDF图变为：

```
# Default Graph
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>  ns:director <http://example/person1> .
<http://example/movie1>  ns:writer <http://example/person3> .
```

图3 SPARQL删除数据结果示例

# 数据操纵

- 数据更新：更新RDF图中指定三元组的值。和SQL不同，SPARQL没有定义UPDATE操作，也就是说SPARQL语言没有直接更新已有数据的方法。但是，可以通过组合INSERT DATA语句和DELETE DATA语句来实现该功能。
  - DELETE DATA：删除待修改的三元组；
  - INSERT DATA：插入修改后的三元组。

例9.17 将图3中三元组 (<http://example/movie1> ns:director <http://example/person1>) 的客体<http://example/person1>修改为<http://example/person4>。

# 数据操纵

其对应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

DELETE DATA
{
<http://example/movie1> ns:director  <http://example/person1>.
};

INSERT DATA
{
<http://example/movie1> ns:director  <http://example/person4>.
}
```

执行该语句后，图3中的RDF图变为：

```
# Default Graph
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>   ns:director  <http://example/person4> .
<http://example/movie1>   ns:writer  <http://example/person3> .
```

图4 SPARQL更新数据结果示例

# 数据查询

- SPARQL提供了丰富的数据查询功能，包括四种形式：
  - SELECT：最为常用的查询语句，其功能和SQL中的SELECT语句类似，从知识图谱中获取满足条件的数据。
  - ASK：用于测试知识图谱中是否存在满足给定条件的数据，如果存在则返回“yes”，否则返回“no”，该查询不会返回具体的匹配数据。
  - DESCRIBE：用于查询和指定资源相关的RDF数据，这些数据形成了对给定资源的详细描述。
  - CONSTRUCT：则根据查询图的结果生成RDF。

# SELECT语句

- SELECT语句的基本语法为：

  SELECT 变量1 变量2 …

  WHERE 图模式

  [ 修饰符 ]

  - SELECT子句中的“变量1 变量2 …”和SQL中的“列1，列2，…”的含义类似，表示要查询的目标。不同之处在于，SQL处理的数据存储于结构化的二维表中，语句中的“列1，列2，…”和数据表中的列完全对应，因此查询的目标具有明确的语义；而SPARQL处理的数据则具有更加灵活的存储结构，“变量1 变量2 …”在知识图谱中没有直接的对应。

# SELECT语句

- SELECT语句的基本语法为：

  SELECT 变量1 变量2 …
  WHERE 图模式
  [ 修饰符 ]

  - WHERE子句用于为SELECT子句中的变量提供约束，查询结果必须完全匹配该子句给出的图模式。图模式主要由两类元素组成，一类是三元组，例如“?x a Person”表示变量?x必须是Person的一个实例；另一类是通过FILTER关键字给出的条件限制，这些条件包括数字大小的限制、字符串格式的限制等。
  - 修饰符是可选项，用于对查询结果做一些特殊处理，例如常见的修饰符有用于表示排序的ORDER子句、用于限制结果数量的Limit子句等。

# SELECT语句

例9.18 在图4所示的RDF图中查询资源<http://example/movie1>的导演和编剧。

相应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

SELECT ?director ?writer
WHERE
{
<http://example/movie1> ns:director  ?director.
<http://example/movie1> ns:writer  ?writer.
}
```

执行该语句的查询结果为：

| director | writer |
| --- | --- |
| <http://example/person4> | <http://example/person3> |

# ASK语句

- ASK语句用于测试知识图谱中是否存在满足给定图模式的数据，其基本语法为：

  ASK 图模式

例9.19 在图4所示的RDF图中测试是否存在由<http://example/person4>导演的电影。

相应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

ASK
{
?movie ns:director <http://example/person4>.
}
```

该语句返回的结果为“yes”；如果查询是否存在<http://example/person5>导演的电影，查询的结果将会返回“no”。

# DESCRIBE语句

- DESCRIBE语句用于获取和给定资源相关的数据，其基本语法为：

  DESCRIBE 资源或变量
  [WHERE 图模式]

  - DESCRIBE后既可以直接跟确定的资源标识符，也可以跟变量；
  - WHERE子句是可选的，用于限定变量需要满足的图模式。

例9.20 在图4所示的RDF图中获取由<http://example/person4>导演的所有电影的相关信息。

# DESCRIBE语句

其对应的SPARQL语句为：

```
prefix ns: <http://example.org/ns#>

DESCRIBE ?movie
WHERE
{
?movie ns:director <http://example/person4>.
}
```

该语句的执行结果为：

```
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>  ns:director  <http://example/person4> .
<http://example/movie1>  ns:writer  <http://example/person3> .
```

# CONSTRUCT语句

- CONSTRUCT语句用于生成满足图模式的RDF图，其基本语法为：

  CONSTRUCT RDF图模板

  WHERE 图模式

  - RDF图模板确定了生成的RDF图所包含的三元组类型，它由一组三元组构成，每个三元组既可以是包含变量的三元组模板，也可以是不包含变量的事实三元组；图模式则和上述SELECT 等语句中介绍的相同，用于约束语句中的变量。
  - 该语句产生结果的基本流程为：首先执行WHERE子句，从知识图谱中获取所有满足图模式的变量取值；然后针对每一个变量取值，替换RDF图模板中的变量，生成一组三元组。

# CONSTRUCT语句

例9.21 在图4所示的知识图谱上执行如下SPARQL语句：

```
prefix ns: <http://example.org/ns#>

CONSTRUCT
{
?movie a ns:Movie .
?movie ns:director ?director .
}
WHERE
{
?movie ns:writer <http://example/person3> .
?movie ns:director ?director .
}
```

?movie = <http://example/movie1>

?director = <http://example/person4>

# CONSTRUCT语句

该语句的执行结果为：

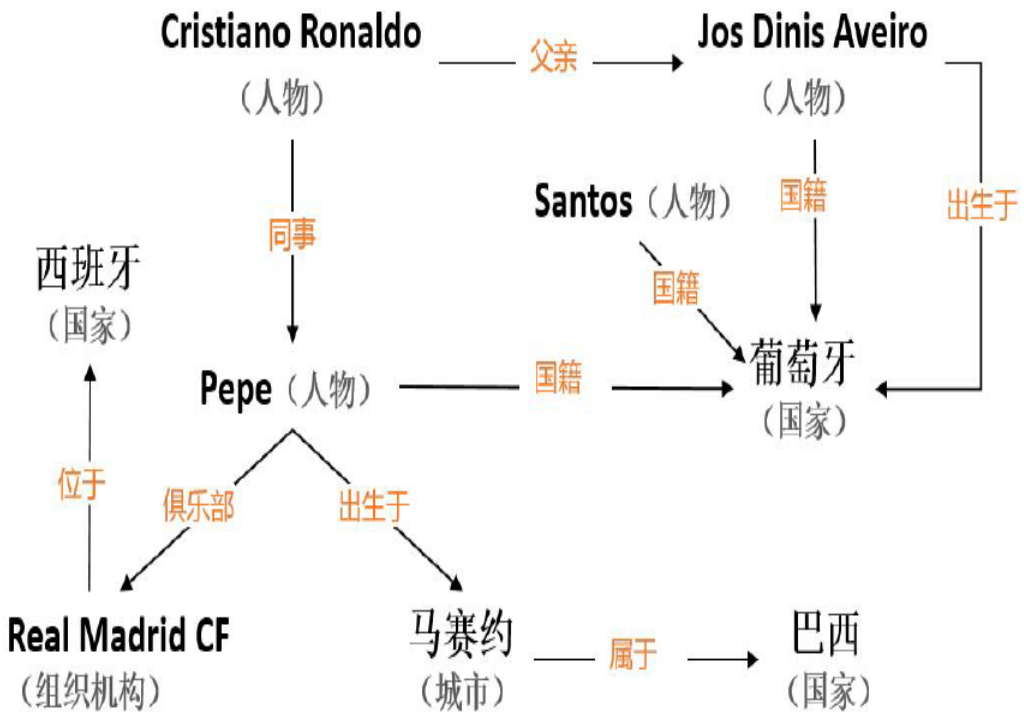
```
@prefix ns: <http://example.org/ns#> .

<http://example/movie1>  a  ns:Movie.
<http://example/movie1>  ns:director  <http://example/person4> .
```

# 提纲

- 概述
- 知识图谱的存储
  - 基于表结构的存储
  - 基于图结构的存储
- 知识图谱的检索
  - 关系数据库查询：SQL语言
  - 图数据库查询：SPARQL语言
  - 图查询技术
- 本章小结

# 知识图谱

- 知识图谱是一种有向图结构，描述了现实世界中存在的实体、事件或者概念以及它们之间的相关关系。




- **实体：**Cristiano Ronaldo, Jos Dinis Aveiro, Real Madrid CF、马赛约等
- **实体类型：**人物、国家、城市、组织机构等
- **属性：**人物有姓名、性别、出生日期、兴趣爱好、职业等属性；国家有国庆日、国家代码、货币、时区、等属性
- **关系：**人物和人物间的同事关系、人物和国家间的国籍关系、城市和国家间的属于关系等

# 知识图谱

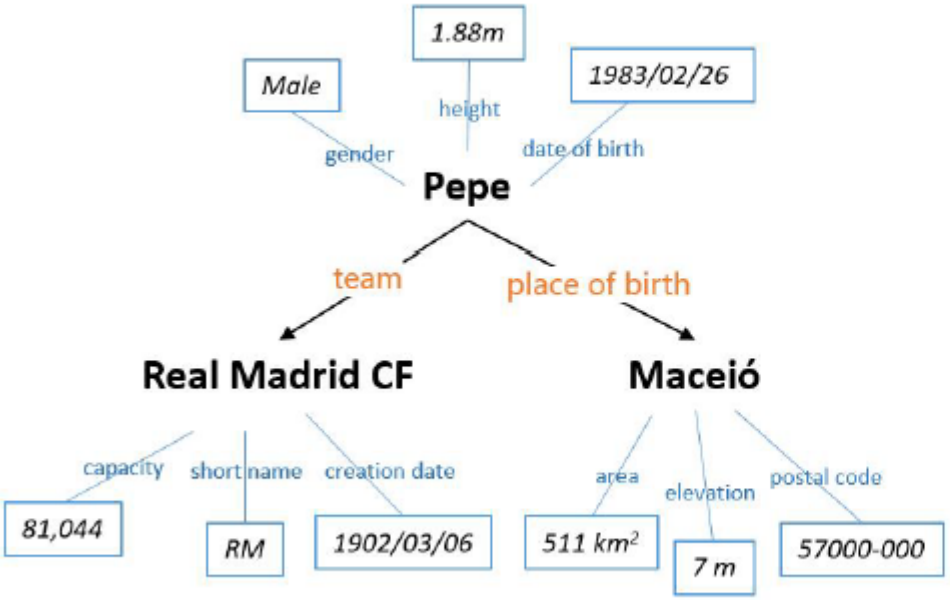
- 知识图谱中的知识是通过RDF的结构进行表示的，其基本构成单元是事实，每个事实被表示为一个形如<subject, predicate, object>的三元组。
- 知识图谱的目标是构建一个能够刻画现实世界的知识库，为自动问答、信息检索等应用提供支撑。因此，对知识的持久化存储并提供对目标知识的高效检索是合格的知识图谱必须具备的基本功能。

# 知识图谱的存储

- 按照存储方式的不同，知识图谱的存储可以分为基于表结构的存储和基于图结构的存储。
  - 基于表结构的存储：利用二维的数据表对知识图谱中的数据进行存储
    - 三元组表、类型表、关系数据库
  - 基于图结构的存储：利用图的方式对知识图谱中的数据进行存储
    - 图数据库

# 三元组表

- 知识图谱中的事实是一个个的三元组，一种最简单直接的存储方式是设计一张三元组表用于存储知识图谱中所有的事实。

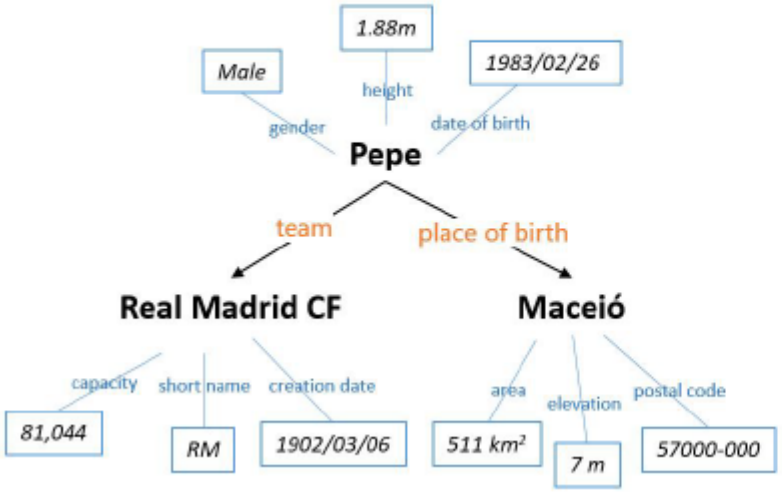



**三元组表**

| S. | P. | O. |
|---|---|---|
| Pepe | gender | male |
| Pepe | height | 1.88m |
| Pepe | date of birth | 1983/02/06 |
| Pepe | team | Ream Madrid CF |
| Pepe | place of birth | Maceió |
| Maceió | area | 511km$^2$ |
| Maceió | elevation | 7m |
| Maceió | postal code | 57000-000 |
| . . . | . . . | . . . |

# 类型表

- 为每种类型构建一张表，同一类型的实例存放在相同的表中。表的每一列表示该类实体的一个属性，每一行存储该类实体的一个实例。




**类型表**

City Table

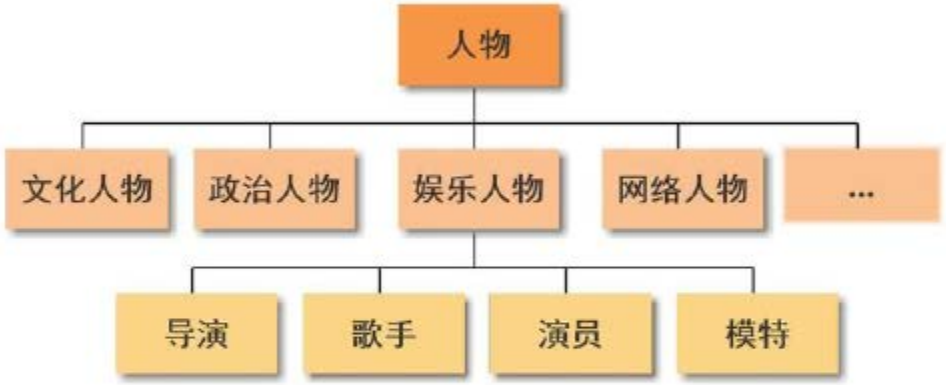
| subject | area | elevation | postal code |
|---|---|---|---|
| Maceió | 511km² | 7m | 57000-000 |

Person Table

| subject | gender | height | date of birth | team | place of birth |
|---|---|---|---|---|---|
| Pepe | male | 1.88m | 1983/02/06 | Ream Madrid CF | Maceió |

# 考虑层级关系的类型表

- 构建数据表时，将知识图谱的类别体系考虑进来。具体来说，每个类型的数据表只记录属于该类型的特有属性，不同类别的公共属性保存在上一级类型对应的数据表中，下级表继承上级表的所有属性。




Person (人物) Table

| subject | 性别 | 出生日期 | 出生地 |
|---|---|---|---|
| 刘德华 | 男 | 1961/09/27 | 香港 |

Entertainer (娱乐人物) Table

| subject | 艺名 | 经纪公司 |
|---|---|---|
| 刘德华 | 刘德华 | 映艺娱乐 |

Actor (演员) Table

| subject |
|---|
| 刘德华 |

Singer (歌手) Table

| subject | 唱片公司 |
|---|---|
| 刘德华 | 东亚唱片 |

# 关系数据库

- 关系数据库以二维表结构对数据进行组织和存储。
  - 属性 (attribute)：表中的每一列称为一个属性（也称字段），用来描述实体集的某个特征。每个属性都有自己的取值范围，称为域。
  - 元组 (tuple)：表中的每一行由一个实体的相关属性取值构成，称为元组（也称记录），它相对完整地描述了一个实体。元组中的一个属性值称为分量。

| 学号 | 姓名 | 性别 | 出生日期 | 身份证号 |
|---|---|---|---|---|
| 1101 | 李强 | 男 | 1995-09-22 | 31010319950922317 |
| 1202 | 张红 | 女 | 1995-11-07 | 41010419951107243 7 |
| 1303 | 王鹏 | 男 | 1996-04-08 | 51010519960408222 3 |

# 关系数据库

- 每张数据表可以由任意数量的属性组成，这些属性按照各自的特点可以分为如下几类：
  - 候选码：能够唯一标识元组的最小的属性集合。
  - 主码：一个数据表可以包含多个候选码，从中任意选取一个作为主码。
  - 外码：如果数据表中的某个属性或属性组是其它表的候选码，那么称该属性或属性组是当前表的外码。
  - 主属性与非主属性：包含在任何候选码中的属性称为主属性；相反地，不包含在任何候选码中的属性称为非主属性。

# 关系数据库完整性约束

- 通过完整性约束条件来保证数据的正确性和一致性。
  - 域完整性：在关系数据模型定义时，由用户对属性值的数据类型、长度、单位、精度、格式、值域范围、是否允许为空值等进行限定，规定属性取值必须在其值域中。
  - 实体完整性：元组（记录）在构成主码的属性上不能有空值且主码的值不能相同。实体完整性主要是为了确保主码能唯一标识元组。
  - 参照完整性：一个外表的外码取值必须是其主表主码的存在值或空值。

# 常见的关系数据库存储系统

- DB2 (https://www.ibm.com/analytics/us/en/technology/db2/)
- Oracle (https://www.oracle.com/database/index.html)
- Microsoft SQL Server (https://www.microsoft.com/zh-cn/sql-server/sql-server-2016)
- PostgreSQL (https://www.postgresql.org/)
- MySQL (https://www.mysql.com/)

# 图数据库

- 图数据库基于有向图，其理论基础是图论。节点、边和属性是图数据库的核心概念。
  - 节点：节点用于表示实体、事件等对象，可以类比于关系数据库中的记录或数据表中的行数据。例如人物、地点、电影等都可以作为图中的节点。
  - 边：边是指图中连接节点的有向线条，用于表示不同节点之间的关系。例如人物节点之间的夫妻关系、同事关系等都可以作为图中的边。
  - 属性：属性用于描述节点或者边的特性。例如人物的姓名、夫妻关系的起止时间等都是属性。

# 常见的图数据库存储系统

- Neo4j (https://neo4j.com/)
- OrientDB (http://orientdb.com/)
- HyperGraphDB (http://www.hypergraphdb.org/)
- InfiniteGraph (http://www.objectivity.com/products/infinitegraph/)
- InfoGrid (http://infogrid.org)

# 知识图谱的检索

- 知识图谱的知识实际上是通过数据库系统进行存储的，大部分数据库系统通过形式化的查询语言为用户提供访问数据的接口。
  - 关系数据库：标准查询语言SQL
  - 图数据库：标准查询语言SPARQL
- 由于知识图谱中的数据在逻辑上本身就是一种图结构，因此也可以通过图查询技术来完成特定查询图的查找，其核心问题是判断查询图是否是图数据集的子图，因此也叫子图匹配问题。

# SQL的组成

- SQL语言从功能上可以分为四个部分：数据查询、数据操纵、数据定义和数据控制。

| 功　能 | 动　词 |
| --- | --- |
| 数据查询 | SELECT |
| 数据操纵 | INSERT, UPDATE, DELETE |
| 数据定义 | CREATE, DROP, ALTER |
| 数据控制 | GRANT, REVOKE |

# 数据查询

- 数据查询是指从数据表中选取满足条件的数据，查询结果存储在一个临时的结果表中。SQL通过SELECT语句完成该功能，其基本语法为：

  SELECT 目标表的列名或列表达式序列
  FROM 基本表名和（或）视图序列
  [ WHERE 行条件表达式 ]
  [ GROUP BY 列名序列 ]
  [ HAVING 组条件表达式 ]
  [ ORDER BY 列名 [ ASC | DESC ], … ]

  - 主语句**SELECT-FROM-WHERE**的含义是：根据WHERE子句的条件表达式，从FROM子句指定的基本表或视图中找出满足条件的元组，再按SELECT子句中的目标列表达式，选出元组中的属性值形成结果表。

# 数据操纵

- 数据插入是指向数据表中添加新的数据记录，SQL通过INSERT语句完成该功能。其基本语法为：

```
INSERT
INTO 表名 [(<属性列1> [,<属性列2>, … ])]
VALUES (<常量1> [,<常量2>, … ])
```

- 数据修改是指修改数据表中已有记录某些列的值，SQL通过UPDATE语句完成该功能。其基本语法为：

```
UPDATE 表名
SET <列名>=<表达式> [,<列名>=<表达式>, … ]
[WHERE 行条件表达式]
```

# 数据操纵

- 数据删除是指从数据表中删除指定的数据记录，SQL通过DELETE语句完成该功能。其基本语法为：

```
DELETE
FROM 表名
[WHERE 行条件表达式]
```

# SPARQL语言

- SPARQL是Simple Protocol and RDF Query Language的缩写，是由W3C为RDF数据开发的一种查询语言和数据获取协议，是被图数据库广泛支持的查询语言。和SQL类似，SPARQL也是一种结构化的查询语言，用于对数据的获取与管理。
  - 数据操纵（插入、删除、更新）
  - 数据查询

# 数据操纵

- 数据插入：将新的三元组插入到已有的RDF图中。SPARQL通过INSERT DATA语句完成该功能。
- 数据删除：从RDF图中删除一些三元组。SPARQL通过DELETE DATA语句完成该功能。
- 数据更新：更新RDF图中指定三元组的值。和SQL不同，SPARQL没有定义UPDATE操作，也就是说SPARQL语言没有直接更新已有数据的方法。但是，可以通过组合INSERT DATA语句和DELETE DATA语句来实现该功能。

# 数据查询

- SPARQL提供了丰富的数据查询功能，包括四种形式：
  - SELECT：最为常用的查询语句，其功能和SQL中的SELECT语句类似，从知识图谱中获取满足条件的数据。
  - ASK：用于测试知识图谱中是否存在满足给定条件的数据，如果存在则返回“yes”，否则返回“no”，该查询不会返回具体的匹配数据。
  - DESCRIBE：用于查询和指定资源相关的RDF数据，这些数据形成了对给定资源的详细描述。
  - CONSTRUCT：则根据查询图的结果生成RDF。

谢谢！

wangquan@iie.ac.cn