## 为什么需要知识图谱

计算机一直面临着这样的困境——无法获取网络文本的语义信息。尽管近些年人工智能得到了长足的发展，在某些任务上取得超越人类的成绩，但离一台机器拥有一个两三岁小孩的智力这样一个目标还有一段距离。这距离的背后很大一部分原因是机器缺少知识。为了让机器能够理解文本背后的含义，我们需要对可描述的事物(实体)进行建模，填充它的属性，拓展它和其他事物的联系，即，构建机器的先验知识。

知识图谱是使用图形化的方式展现知识的内部结构以及外部关系的知识表示形式，是语义网的重要基础设施。当今的互联网正从仅包含网页和网页之间超链接的文档万维网(Web of Document)转变成包含大量描述各种实体和实体之间丰富关系的数据万维网(Web of Data)。

有知识图谱作为辅助，搜索引擎能够洞察用户查询背后的语义信息，返回更为精准、结构化的信息，更大可能地满足用户的查询需求。Google知识图谱的宣传语“things not strings”给出了知识图谱的精髓，即，不要无意义的字符串，而是获取字符串背后隐含的对象或事物。

## 数据库和深度学习有什么关系

1) 以云计算和大数据的技术和资源为基础，深度学习通过简单函数迭代的计算模型和高维复杂函数近似表示方法抽取数据中隐藏的层次结构，在视觉和语言处理等领域取得显著成功；

2)数据库通过关系模型表示对象以及关系，依赖索引技术取得计算效率，支撑数据库系统的广泛应用。

3)关系、结构是两者关注的共同点，很自然地深度学习技术在数据库系统中的应用可以在多个层次和维度上拓展数据系统的模型表示能力、复杂算子表示及其效率方法，能够导致：

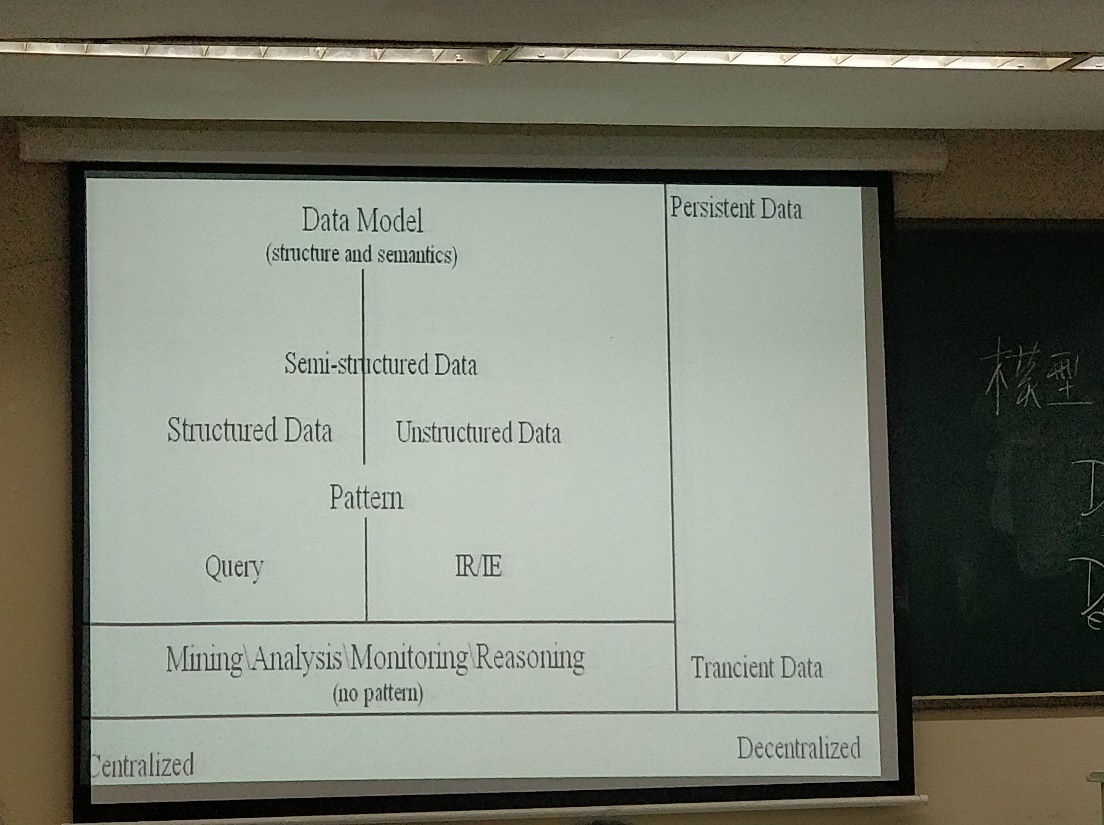
1）扩展数据库系统的复杂相似关系和相似算子的表示，构建统一的多模态数据库系统模型；

2）隐藏复杂关系的发现和深度神经网络模型的表示；

3）通过深度神经网络表示的数据结构分布设计自适应的索引结构，形成数据驱动的索引和查询策略优化。

## 数据库的逻辑模型（从已知到未知）

逻辑模型：是将概念模型转化为具体的数据模型的过程，即按照概念结构设计阶段建立的基本E-R图，按选定的管理系统软件支持的数据模型（层次、网状、关系、面向对象），转换成相应的逻辑模型。这种转换要符合关系数据模型的原则。目前最流行就是关系模型（也就是对应的关系数据库）



* 数据分类
  + 结构化数据：表格数据
  + 半结构化数据：xml数据，知识图谱（谁和谁有关系）
  + 非结构化数据：视频，音频，图片
* 查找方式：
  + 结构化数据：Query，SQL
    - 用结构化查询查找结构化数据，匹配方式是是否满足
  + 非结构化数据：关键词，IE/IR（信息抽取/信息检查）
    - 用非结构化查询查找非结构化数据，匹配方式是相似或相关
* Pattern/No pattern
  + pattern：query, IE/IR 是在数据中查找已知的东西
  + no pattern：mining/analysis/reasoning/monitoring 是从得到的数据中想找出未知的东西
* 数据持久性：
  + persistent data: 数据存在磁盘上，想什么时候看就什么时候看
  + transient data: 瞬时数据，不能扫描第二遍
* 中心化与去中心化：
  + centralized：单节点控制访问
  + decentralized：多节点控制访问

## 什么叫query, search, mining, analysis

数据模型中包含 mining，analysis, monitoring, reasoning

query：能够找出数据库中存在的，最直接的数据，这些数据一般情况下只有开发者能够看懂。

search:在query的基础上，对已有的数据基础上按照一定规则进行重新组织，并且配合一些多媒体数据展现给需要查询的人。

analysis:对数据进行分析，根据分析目的，用适当的统计分析方法及工具，对收集来的数据进行处理与分析，提取有用的信息，发挥数据的作用。数据分析一般是得到一个指标统计量结果。如总和，平均值等，这些指标数据都需要与业务结合进行解读才能发挥出数据的价值与作用。

mining：从大量的数据中，通过统计学、人工智能、机器学习等方法，挖掘出未知的、且具有价值的信息和知识的过程。其结果输出模型和规则，并且可相应得到模型得分或标签。

reasoning:推理，从已知到未知

monitoring：看数据如何变化，变化后报警（如信用卡额度变化）

## 数据库代表性的成就

关系数据库系统的成就

（1）关系数据模型：数据的语法和语义

（2）索引：化解大海捞针的问题，查询导向相关数据

（3）查询优化，代价较小的执行计划

（4）并发调度：吞吐量

（5）分布的可扩展性：垂直扩展性和水平扩展性

## 数据库和其他东西的关系（区块链）

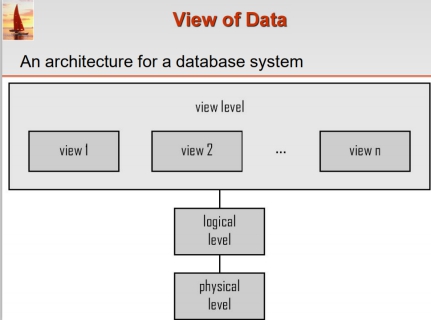
数据库与区块链

1、去中心化，区块链的优点：传统的数据库需要一定的访问权限去管理，也就是说你必须依赖人为的管理，一旦访问权限被黑或者管理员作恶，数据就有很大的危险性，或者存在被人篡改的可能性，区块链用数据加密学和工作量证明以及节点的共识的机制来保证数据的安全。

2、健壮性：区块链的优点：区块链的另外一个优点就是高容错，因为它内在的冗余机制，每个节点对于整个区块链网络而言，都是必不可少的，节点之间相互通信，即使一些节点因为一些原因出错，也不会导致整个网络垮掉，同时失败的节点，重启之后也总是能正确同步新的交互。

3、性能，中心数据库的优点：区块链比传统的数据库要慢，并不是因为区块链技术本来就慢，而是因为区块链本身就是新生事物，依然处于早期阶段，性能优化仍然有一定的空间，同时因为区块链除了像传统数据库那样运行，还要做额外的负载：1、签名机制2、共识机制3、冗余性4、透明性：在区块链的交易都是透明的，区块链在一下场景应该会更擅长：a、公司内部审计b、来源追踪c、轻金融系统

## 数据视图（数据抽象）：物理层，逻辑层和视图层



* 物理层：最底层的抽象，描述数据实际是怎样被存储。（物理层详细描述复杂的底层数据结构，是开发DBMS的数据库供应商应该研究的事情。）
* 逻辑层：比物理层稍高，描述数据库中存储了怎样的数据以及数据间的关系。（因而整个数据库可通过少量相对简单的结构来描述。虽然简单的逻辑层结构的实现涉及到复杂的物理层结构，但逻辑层的用户不必知道这种复杂性。逻辑层抽象是由数据库管理员和数据库应用开发人员是用的，他们必须确定数据库中应该保存哪些信息。）
* 视图层（面向对象）：最高层次的抽象，只描述数据库的某个部分。（尽管在逻辑层是用了比较简单的结构，但由于数据库的规模巨大，所以仍存在一定程序的复杂性。数据库系统的最终用户并不需要关心所有的信息，而只需要访问数据库的一部分。视图抽象层的定义正是为了使用户与系统的交互更简单。系统可以为同一数据库提供多个视图，而视图又保证了数据的安全性。）

## 数据模型

#### 数据模型按不同的应用层次分为3种类型：

**数据模型一般由数据结构，数据操作，完整性约束条件三部分组成**

* + 概念模型：
    1. 面向数据库用户的现实世界的模型，主要用于描述世界的概念化结构，使设计人员集中精力分析数据以及数据间的联系，概念数据模型必须转换成逻辑数据模型才能在DBMS中实现
    2. 用于信息世界的建模，简单清晰，易于用户理解
    3. 常用的是E-R模型（实体-联系模型）：
       1. 把每一类数据对象的个体称为实体，而每一类对象个体的集合称为实体集。
       2. 每个实体集涉及的信息项称为属性，若属性或最小属性组合的值能唯一标识其对应实体，则将该属性（组合）称为码
       3. 实体之间的关系称为“联系”。两个实体集之间有三种联系：一对一，一对多，多对多。
  + 逻辑模型：
    1. 是用户从数据库所看到的模型，是具体的DBMS所支持的数据模型。
    2. 此模型既面向用户，又面向系统
    3. 主要用于DBMS的实现。
       1. 层次模型

将数据组织成一对多关系的结构，采用关键字来访问其中每个层次的每一部分。

* + - 1. 网状模型

具有多对多类型的数据组织方式

* + - 1. 关系数据模型
         1. 关系模型以记录组或二位数据表的形式组织数据，以便利用各种实体与属性之间的关系进行存储和变换
         2. 关系模型由关系数据结构、关系操作集合和关系完整性约束三部分构成。在关系模型中，现实世界实体以及实体间的联系均用关系来表示。
         3. 关系操作的特点是集合操作方式，即操作的对象和结果都是集合。关系操作可以使用两种方式定义：基于代数的定义称为关系代数；基于逻辑的定义称为关系演算。由于使用变量的不同，关系演算又分为元组关系演算和域关系演算。

1.数据结构：把数据库表示为关系的集合，关系（表格）

2.数据操作：关系代数，常用的关系操作包括：

选择、投影、连接、除、并、交、差查询等

增加，删除，修改

3.数据完整性约束：主外键，参照完整性，自定义（assert）

实体完整性（必须满足）：关系的主码不能为空值

参照完整性（必须满足）：外码必须是另一个表中主码的有效值，或者是“空值”

用户定义完整性

* + 物理模型：
    1. 面向计算机物理表示的模型，表示了数据在存储介质上的组织结构，
    2. 它不但与具体的DBMS有关，而且与操作系统的硬件有关
    3. 每一种逻辑模型在实现时都有其对应的物理模型。

## 数据视图与数据模型

数据抽象共有三个层次：物理层、逻辑层和视图层。数据模型通常由数据结构、数据操作和完整性约束三部分组成。

在逻辑层使用的数据模型包括两类：一类是概念数据模型，主要用于数据库设计，它能被一般的用户理解，与人的思维表达方式比较接近。这样的模型有实体-联系模型（ERM）；另一类是逻辑数据模型，按计算机系统的观点对数据建模，使得数据更适合用计算机加以表示。这里模型主要用于DBMS的实现，比如关系模型、面向对象模型、层次模型和网状模型。设计师构建数据库模式的方法通常是首先使用E-R模型在高层对数据建模，然后再将其转变成关系模型。在物理层使用的数据模型称为物理数据模型。

**数据抽象：**是将数据抽象化、逻辑化，是对数据的抽取过程。

**数据模型：**是对数据进行抽象化表示的工具，主要使用逻辑概念来表示数据。

**数据模式：**利用数据模型组织抽取的数据所得的结果，也即是数据抽象的结果

**三者之间的关系：**

数据抽取作为总的过程，利用数据模型，对现实具体系统的数据进行抽取，组织，使其具有结构化的特征，最终得到的结果，即是数据模式。

## 可序列化：

#### 事务

1. 事务是用户定义的一个数据库操作序列，是数据库应用程序的基本单元，是反映现实世界需要以完整单位提交的一项工作。
2. 事务的四个特征：原子性、一致性、隔离性和持久性。（ACID）
3. 事务处理包括数据库恢复和并发控制。数据库恢复有两个目的：保证事务的原子性和使数据库能恢复到正确状态。
4. 事务在并发调度过程中，会产生多种结果，只有串行调度才是正确的结果。并发过程的结果只有与串行调度结果一样的才是正确的。这种并发调度被称为可串行化调度。

**序列化指的是多个事务并发的执行结果与顺序执行的结果一致。一个问题是不是可序列化问题是一个NP-C问题**

* 冲突可串行化：
  + 1. 冲突操作指的是不同事务对于同一数据的读写操作与写写操作
    2. 有些冲突操作是可以交换次序的，有些冲突操作不能交换次序
    3. 不能交换位置的次序为：
       - 1. 不同事务的冲突操作
         2. 同一事务的两个操作
    4. 一个调度Sc在保证冲突操作（即3中的两种情况）次序不变的情况下，通过交换两个事务不冲突操作的次序得到另一个调度Sc'，如果Sc'是串性的，则称调度Sc为冲突可串行化的调度。
* 视图可串行化

1. 如果一个schedule 是视图等价于一个串行化 schedule，那么它是视图可串行化的。
2. 每一个冲突可串行化的 schedule都是视图可串行化的 ，反过来不成立。
3. 如果视图可串行化不是冲突可串行化，那么一定是因为多了1到多个盲目写操作。

* 偶尔冲突的加时间戳timestamp：每一个事务都有一个自己的时间，用于表明先后。每一个数据项也有读和写的时间戳，是最后一个进行这个操作的事务的时间戳。
* 经常冲突的采用加锁：分为加锁和解锁两个过程，加锁阶段不能解锁，一旦解锁就不能再加锁。也就是所有的事务开始时全部申请加锁，然后执行，最后在全部解锁，不能出现解锁 以后又加锁的情况。这个方法可能导致死锁。而且他可以产生序列化，但是有的事务可以序列化，但是却无法通过这个算法产生。

## 数据库模式和数据库实例

* 面向对象>>>class>>>object
* 数据模型>>>数据库模式>>>数据库实例
* 模式是数据库中全体数据的逻辑结构和特征描述，它仅仅涉及型的描述，不涉及具体的值。模式的一个具体的值称为模式的一个实例。同一个模式可以有多个实例。
* 模式的三级结构：
  1. 模式（逻辑模式）：是数据库中全体数据的逻辑结构和特征描述，所有用户的公用数据视图，是模式结构的中间层
  2. 内模式（储存模式）：一个数据库只有一个内模式，它是数据物理结构和存储方式的描述，是数据在内部的组织方式。如按什么方式索引，是否加密。
  3. 外模式（子模式，用户模式）：数据库用户能看见和使用的局部数据的逻辑结构和特征描述，是数据库用户的数据视图

1. **知识图谱和数据库的转化**
2. **连接运算**

连接运算是从两个关系的乘运算结果中选取属性间满足一定条件的元组，构成新的关系。连接运算有两种：等值连接和自然连接。自然连接要求两个关系中进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉。

#### SQL描述性语言

SQL转化为关系代数

## 模型的实现

#### 查询的执行计划：查询优化

A.范围查询，点查询，存在查询

B.吞吐率——>并发

#### 索引（函数，映射）

B+树，散列，位图