问题一： 说明关系数据库规范化理论对于学习数据库系统课程有什么指导作用？

关系数据库逻辑设计的好坏与其所含的各个关系模式设计的好坏相关。如果各个关系模式结构合理、功能简单明确、规范化程度高，就能确保所建立的数据库具有较少的数据冗余、较高的数据共享度、较好的数据一致性，并为数据库系统能够很好的应用于实际打下良好的基础。 关系规范化的目的是解决关系模式中存在的数据冗余、插入和删除异常、更新繁琐等问题。其基本思想是消除数据依赖中的不合适部分，使各关系模式达到某种程度的分离，使一个关系描述一个概念、一个实体或实体间的一种联系。因此，规范化的实质是概念的单一化。

针对一个具体问题，应该如何构造适合于它的数据模式是关系数据库逻辑设计的核心问题，也正是关系数据库规范化理论的研究内容。一个关系通常是由赋予它的元组语义来确定的。凡符合元组语义的那部分元素的全体就构成了该关系模式的关系。现实世界随着时间不断变化，在不同时刻，关系模式的关系也会有所变化。但现实世界的许多已有事实限定了关系模式所有可能的关系必须满足一定的完整性约束条件，这些约束或者通过对属性取值范围的限定，或者通过属性值之间的相互关联反映出来。后者称为数据依赖，它是现实世界属性间相互关系的抽象，是数据内在的性质和语义的体现，也是数据模式设计求精的关键。数据库概念设计得到的是一个关系的集合与一组完整性约束，这是进行数据库设计的开端，而更严格的设计过程必须通过更加完整地考虑完整性约束来对数据库模式的初始设计进行求精。

数据库的模式规范化需要解决的主要问题是从关系模式的初始状态开始，不断修改模式结构，以减少或消除冗余，同时还应充分考虑修改而引起的相关问题。给定一个关系模式，规范化理论需要确定它是否是一个良好的设计，或者是否需要进行模式分解。“范式”可以帮助我们做出上述决定。如果一个关系满足某种范式，那么就肯定不会发生诸如插入或删除异常此类特定的问题。如果不能满足某种范式，则需要判定其是否能通过模式分解，将原关系模式转化成为满足要求的范式，从而避免出现一些不良问题。范式的定义以函数依赖为基础。按其规定的严格性从低到高的顺序为：第一范式（１ＮＦ），第二范式（２ＮＦ），第三范式（３ＮＦ），Ｂｏｙｃｅ－Ｃｏｄｄ范式（ＢＣ－ＮＦ），以及第四范式（４ＮＦ），第五范式（５ＮＦ）等等。其中比较重要的是３ＮＦ，ＢＣＮＦ等范式类型。直观地说，在一个３ＮＦ的关系中，每一个非主属性既不部分依赖于码也不传递依赖于码，而一个满足ＢＣＮＦ的关系排除了任何属性对码的传递依赖和部分依赖。ＢＣＮＦ可以确保只使用函数依赖不能再检测出冗余，即在函数依赖范畴下，ＢＣＮＦ是能达到的最好的范式。

关系模型因其严格的数学理论基础，使关系数据库的规范化设计获得了空前的成功。自Ｃｏｄｄ提出这一理论以来，关系数据库的规范化理论研究取得了丰富的成果。这些研究工作基于这样的主线来开展：以数据依赖为核心，讨论依赖关系的各种合宜形式，试图减少或消除不合理的依赖关系，使数据库模式达到符合不同级别要求的范式，从而使数据库系统冗余少，一致性高，进而获得更高的查询和检索效率。

数据库构建过程中规范化设计具有明显的优势，要以规范化的模式进行研究，积极制定更加高效的数据库设计方案，并有效的进行实践运用，才能保证数据库运行质量。

问题二：描述一个具体的数据库系统应用开发需求，使用课程中所学的关系数据库设计方法，设计该系统的数据库。具体要求如下：

详细描述数据库的设计需求。 使用实体-联系（E-R）模型进行数据库的概念设计，使用实体-联系（E-R）图清晰表示该数据库的E-R模型。要求E-R图绘制规范。 将E-R模型转换为关系数据库模式，使用SQL语言创建该数据库中的所有关系及约束。 为了方便数据库系统应用程序的开发，可以在该数据库上创建哪些视图和索引？

设计需求：

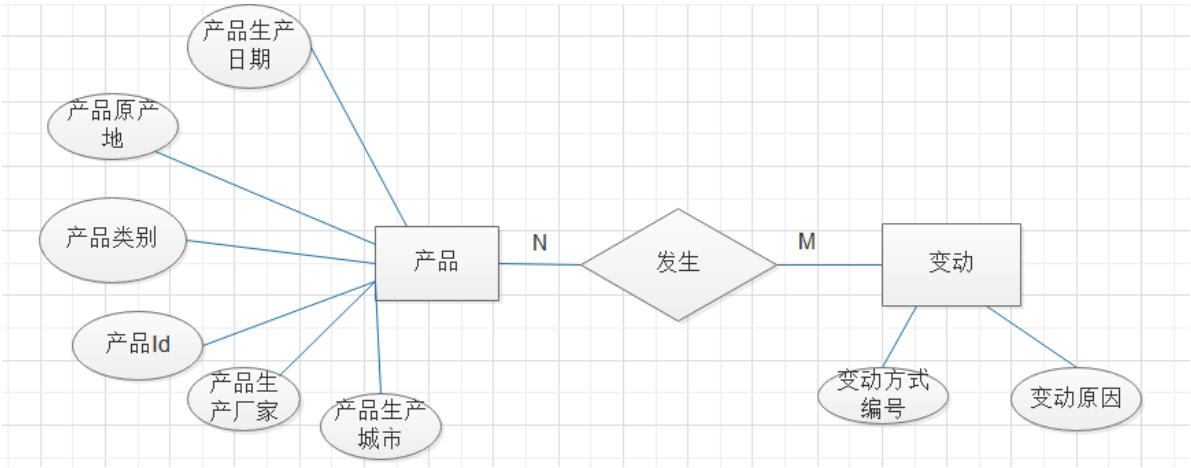
我希望制作的是一个仓库产品管理系统，其中产品信息包括产品Id,产品名称，产品类别，产品生产厂家编号，产品生产城市编号，产品生产日期，产品原产地等七个信息。另外还有产品生产方式变动信息，其中包括Id，产品Id，变动方式编号，变动时间，变动原因。此外，还有国内运输信息以及国际运输信息，他们都包含了产品Id，运送时间，相对应的省份编号以及国家编号，以及对应的城市。在这些信息里面，为产品生产厂家编号、产品生产城市编号、省份编号、国家编号、变动方式编号单独建立各自对应的表，里面包含各自编号所对应的具体信息，例如产品生产厂家、产品生产城市、国内运输的省份、国际运输的国家以及产品变动方式。

该应用为仓库产品管理系统，可以提前输入产品信息，也可以之后手动输入产品信息，为了应用的使用，产品生产厂家编号、产品生产城市编号、变动方式编号、省份编号以及国家编号均需提前进行输入，当插入产品变动信息的时候，就要调用产品变动方式编号，当输入产品国内运输信息的时候就要调用省份编号，当输入产品国际运输信息的时候就要调用国家编号，输入产品信息的时候就要调用产品生产厂家编号以及产品生产城市编号。

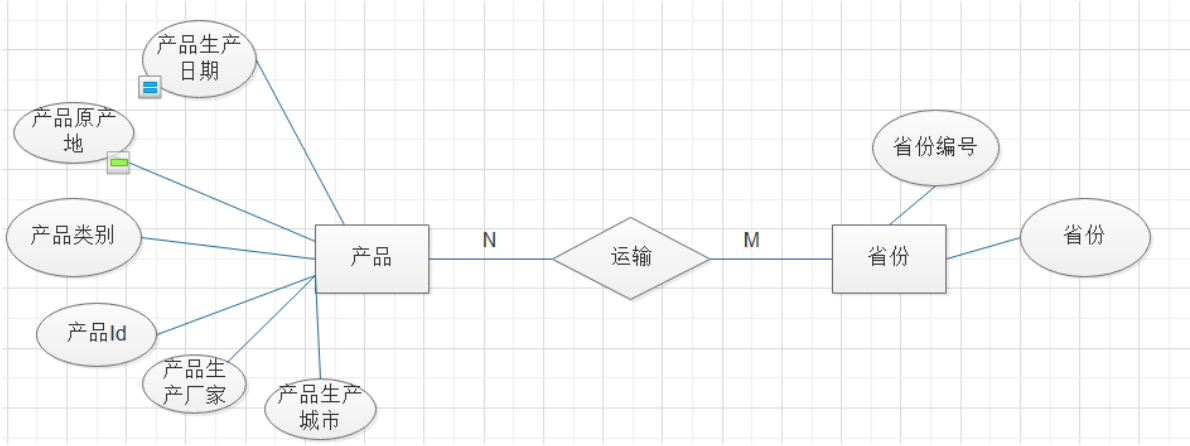
E-R图：

在数据库系统中设计了如下E-R模型：

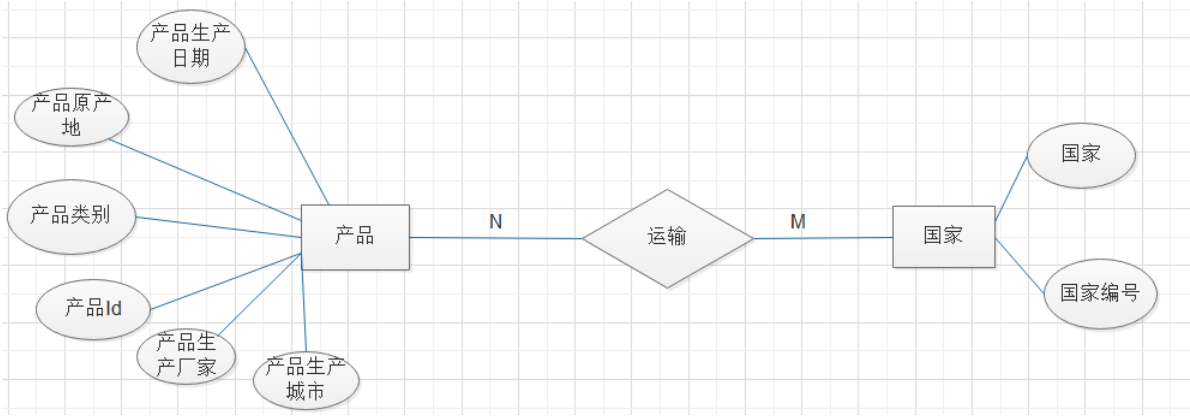
产品变动ER模型：



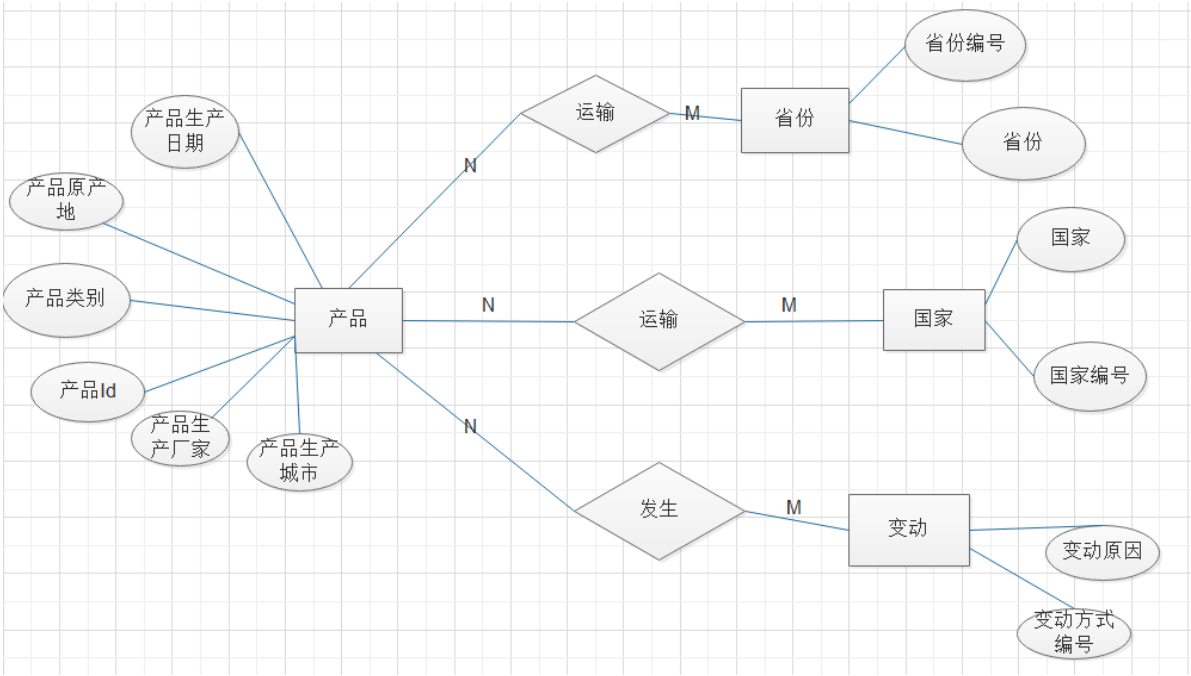
产品国内运输ER模型：



产品国际运输ER模型：



使用ER图表：



关系数据库模式：

将ER模型转换为关系数据库模式。

Proinfo(proid,proname,protype,factory,city,prodate,originplace)

Domestictransport(proid,proname,protype,factory,city,prodate,originplace,pid,city)

Internationaltransport(proid,proname,protype,factory,city,prodate,originplace,cid,city)

Productchange(proid,proname,protype,factory,city,prodate,originplace,changecode,way)

Pcode(pid,prov)

Ccode(cid,country)

Wayofchange(changecode,way)

运用关系数据库规范化理论，对数据库模式进行规范化。

规范化后数据库模式如下

Proinfo(proid,proname,protype,facid,cityid,prodate,originplace)

Domestictransport(id,proid,proname,pid,city)

Internationaltransport(id,proid,proname,cid,city)

Productchange(id,proid,proname,changecode,way)

Pcode(pid,prov)

Ccode(cid,country)

Wayofchange(changecode,way)

Factory(facid,facname)

City(cityid,cityname)

使用MySql进行数据库的构建，sql语言如下：

```
create database productmanage;

create table productchange(
id int(10) not null,
proid int(10) not null,
changecode int(10) not null,
changetime date,
changeresult varchar(20),
primary key(id),
foreign key(proid) references proinfo(proid),
foreign key(changecode) references wayofchange(changecode));

create table wayofchange(
changecode int(10) not null,
way varchar(20),
```

```
primary key(changecode));

create table factory(
facid int(10) not null,
facname varchar(20),
primary key(facid));

create table city(
cityid int(10) not null,
cityname varchar(20),
primary key(cityid));

create table domestictransport(
id int(10) not null,
proid int(10) not null,
pid int(10) not null,
ptime date,
city varchar(20),
primary key(id),
foreign key(proid) references proinfo(proid),
foreign key(pid) references pcode(pid));

create table internationaltransport(
id int(10) not null,
proid int(10) not null,
cid int(10) not null,
ptime date,
city varchar(20),
primary key(id),
foreign key(proid) references proinfo(proid),
foreign key(cid) references ccode(cid));

create table pcode(
pid int(10) not null,
```

```
prov varchar(20),
primary key(pid));

create table ccode(
cid int(10) not null,
country varchar(20),
primary key(cid));

create table proinfo(
proid int(10) not null,
proname varchar(20),
protype varchar(20),
facid int(10) not null,
cityid int(10) not null,
prodate date,
originplace varchar(20),
primary key(proid),
foreign key(facid) references factory(facid),
foreign key(cityid) references city(cityid));
```

在数据库的应用当中，有四个查询要求，一共建立了四个视图，sql语句如下：

第一个是查询产品的本身信息（产品编号，产品名称，生产工厂编号，生产国家编号，生产时间，原产地城市）

```
create VIEW PROINFO_view(proid, proname, protype, facid,
cityid, prodate, originplace)
AS SELECT * FROM proinfo
```

第二个是查询产品变化信息（产品编号，变化方式，变化时间，变化原因）

```
create VIEW ProChange_view AS
select productchange.proid,
wayofchange.way,
productchange.changetime,
productchange.changeresult
from productchange
join wayofchange
on wayofchange.changecode = productchange.changecode
```

第三个是查询产品国内运输信息（产品编号，省份，时间，城市）

```
create VIEW domestictransport_view AS
select domestictransport.proid,
pcode.prov ,
domestictransport.ptime ,
domestictransport.city
from domestictransport
join pcode
on pcode.pid=domestictransport.pid
```

第四个是查询产品国际运输信息（产品编号，国家，时间，城市）

```
create VIEW internationaltransport_view AS
select internationaltransport.proid,
ccode.country ,
internationaltransport.ptime ,
internationaltransport.city
from internationaltransport
join ccode
on ccode.cid=internationaltransport.cid
```

使用SQL定义数据库索引。

根据数据库应用中的四个查询可以进行索引的选择，由于四个查询的查询均是根据产品编号进行查询的，所以在4个关键信息上的proid属性上建立B+树索引，SQL语句如下：

```
create INDEX idx ON proinfo(proid) USING BTREE
create INDEX idx1 ON productchange(proid) USING BTREE
create INDEX idx2 ON domestictransport(proid) USING BTREE
create INDEX idx3 ON internationaltransport(proid) USING BTREE
```

视图和索引：

根据数据库工作负载，运用反规范化和分表等方法，可以调整数据库模式。

根据数据库工作负载，设计数据库索引。

根据应用当中的四个频繁出现的查询语句可以选择索引方式。

在应用当中，有四个查询要求，一个是查询产品的本身信息；第二个是查询产品变化信息；第三个是查询产品国内运输信息；第四个是查询产品国际运输信息。

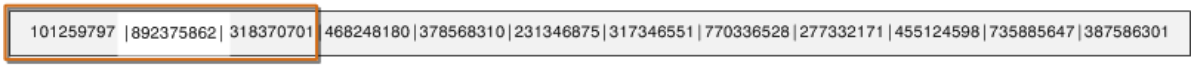
进行这四个查询的时候，选择where的表达式均是对proid的值进行判断，故应该在proid上建立索引，又因为proid只有一个属性，所以不会受到B+树索引的弊端的限制，故最终决定使用B+树索引来对proid进行索引的建立。

根据四个查询可知要在四个关系上建立proid的B+树索引，四个关系分别是productchange，domestictransport，internationaltransport，proinfo。具体的SQL语句实现见下一个板块。

问题三：自行查阅资料，了解一种广泛应用在分析型数据库系统中的存储方法——按列存储，对比课上所学的按行存储方法，说明两种存储方法的优点和缺点。

大部分传统的关系型数据库，都是面向行来组织数据的。如 Mysql，Postgresql。近几年，也越来越多传统数据库加入了列存储的能力。虽然列存储的技术在十几年前就已经出现，却从来没有像现在这样成为一种流行的存储组织方式。分析型数据库（OLAP）的兴起，使得列式存储这一概念又变得流行。列式数据库是以列相关存储架构进行数据存储的数据库，主要适合于批量数据处理和即时查询。相对应的是行式数据库，数据以行相关的存储体系架构进行空间分配，主要适合于小批量的数据处理，常用于联机事务型数据处理。

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL



Block 1

列式存储的概念如上图所示。同一列的数据被一个接一个紧挨着存放在一起，表的每列构成一个长数组。这种存储方式经常被用于分析数据库当中。由于机械磁盘受限于磁头寻址过程，读写通常都以一块（block）为单位，故在操作系统中被抽象为块设备，与流设备相对。这能帮助上层应用是更好地管理储存空间、增加读写效率等。这一特性直接影响了数据库储存格式的设计：数据库的 Page 对应一个或几个物理扇区，让数据库的 Page 和扇区对齐，提升读写效率。大多数服务于在线查询的 DBMS 采用 NSM (N-ary Storage Model) 即按行存储的方式，将完整的行（即关系 relation）从 Header 开始依次存放。页的最后有一个索引，存放了页内各行的起始偏移量。由于每行长度不一定是固定的，索引可以帮助我们快速找到需要的行，而无需逐个扫描。NSM 的缺点在于，如果每次查询只涉及很小的一部分列，那多余的列依然要占用掉宝贵的内存以及 CPU Cache，从而导致更多的 IO；为了避免这一问题，很多分析型数据库采用 DSM (Decomposition Storage Model) 即按列分页：将 relation 按列拆分成多个 sub-relation。类似的，页的尾部存放了一个索引。

无论对于磁盘还是内存数据库，IO 相对于 CPU 通常都是系统的性能瓶颈，合理的压缩手段不仅能节省空间，也能减少 IO 提高读取性能。列式存储在数据编码和压缩上具有天然的优势。编码之后，还可以对数据进行压缩。由于一列的数据本身具有相似性，即使不做特殊编码，也能取得相对较好的压缩效果。通常采用 Snappy 等支持流式处理、吞吐量高的压缩算法。最后，编码和压缩不仅是节约空间的手段，更多时候也是组织数据的手段。在 PowerDrill、Dremel 等系统中，我们会看到很多编码本身也兼具了索引的功能，例如在扫描中跳过不需要的分区，甚至完全改表查询执行的方式。

在现代的大数据架构中，GFS、HDFS 等分布式文件系统已经成为存放大规模数据集的主流方式。分布式文件系统相比单机上的磁盘，具备多副本高可用、容量大、成本低等诸多优势，但也带来了一些单机架构所没有的问题：1.读写均要经过网络，吞吐量可以追平甚至超过硬盘，但是延迟要比硬盘大得多，且受网络环境影响很大。2.可以进行大吞吐量的顺序读写，但随机访问性能很差，大多不支持随机写入。为了抵消网络的 overhead，通常写入都以几十 MB 为单位。上述缺点对于重度依赖随机读写的 OLTP 场景来说是致命的。所以我们看到，很多定位于 OLAP 的列式存储选择放弃 OLTP 能力，从而能构建在分布式文件系统之上。

基于列模式的存储，天然就会具备以下几个优点：1.自动索引。因为基于列存储，所以每一列本身就相当于索引。所以在做一些需要索引的操作时，就不需要额外的数据结构来为此列创建合适的索引。2.利于数据压缩。利于压缩有两个原因。一来你会发现大部分列数据基数其实是重复的，拿上面的数据来说，因为同一个 author 会发表多篇博客，所以 author 列出现的所有值的基数肯定是小于博客数量的，因此在 author 列的存储上其实是不需要存储博客数量这么大的数据量的；二来相同的列数据类型一致，这样利于数据结构填充的优化和压缩，而且对于数字列这种数据类型可以采取更多有利的算法去压缩存储。而行式数据库存储更容易实现事务性、一致性控制。

问题四：人工智能在自然语言处理、图像处理、计算机视觉等领域发挥了重要作用。请你查阅资 料，结合所学的数据库系统知识，说明人工智能未来对数据库系统的发展有哪些帮助。

从传统研究内容来看，人工智能大部分倾向于理论方面的研究，而数据库技术则更为倾向于实际应用。近年来，数据库技术的持续发展，当前已经提出DBMS可以自动有效的管理超大规模数据库，同时又可以用数据驱动的模式自动提供有关的决策，即利用DBMS，能够针对数据进行更为智能化的管理。上述背景下，有关数据库技术与人工智能的融合逐渐成为研究的重要内容。对电力行业来说，近年来有关技术研究的持续深入，数据库技术与人工智能融合也逐渐成为很多行业研究的重要内容。

从目前数据库的发展现状来看，数据库与人工智能有着紧密的结合。人工智能的核心问题主要分为提出问题、解答问题、学习问题三种形式，在提出问题方面主要表现为人工智能理解知识的范围，由于人工智能存在一定的局限性，对知识的掌握不具备充实性，对理解知识的范围存在误区，则需要找出问题的根源，即解答问题，在人工智能的发展阶段，对发现的问题进行及时解答。在人工智能不断学习的过程中，应利用掌握的知识来学习新的知识，把更多的知识运用在数据库的使用中，完善数据库的使用作用。表达知识的最主要特点就是根据人们掌握的程度进行表达，在数据库的使用过程中，根据知识掌握程度成功输送到数据库中，形成数据库中的知识库，在使用数据库的过程中，可充分的利用知识库中的知识，有效方便的为人们提供服务，大大提升数据库的存在价值。数据库与知识库有着紧密的联系，数据库的相应技术可管理知识库，在知识库的使用过程中，为接近人们思维的表达方式，有些表达方式可采用真实模拟性，在表达电影方面，数据库创造出多媒体技术，其有效改善人们的生活质量，多媒体技术不限制空间性，数据存在不规则性，使用空间范围较广。

人工智能的关键在于专家系统，专家系统主要是将专家研究的经验存入到计算机中，将计算机的使用技能得以升华。在人工智能的应用中，智能管理系统起着重要作用，数据库的使用过程中往往会出现违规操作，这需要制定出相应的管理系统，即智能管理系统。智能管理系统需将管理相关条例输入到计算机中，在使用数据库的过程中，出现的违规操作，智能管理系统能及时准确的进行修改，改善数据库存在问题现象的发生。

人工智能技术还可以用于数据库的查询，在人工智能运用下数据库查询功能中，常常需要对大量较为复杂的数据进行查询，对数据查询的便捷性与时间性要求越来越高，要求通过数据查询能够得到较为直接的分析结果。基于用户的数据查询需求，在人工智能数据查询过程中、可以构建多个系统管理与运行模块，同时对模块运行的功能要求高。在系统管理模块，要求能够快速地进行用户注册与登录。在数据查询模块，要求能够实现数据信息的有效查询与输出。在数据订阅模块，要求能够实现对数据信息的订阅。为数据库中的内容建立自然语言解析方式，使得运行中可对数据库中的内容与数据运用SQL语言。在用户权限管理中，对用户进行组下权限的有效分配与管理操作，提升数据信息的真实性与可靠性。为数据库的信息管理与运行设置相关的辅助操作功能，包括数据可视化、语言语音输入以及历史记录查询等人性化的功能。

以2019年IBM发布的Db2 11.5为例。Db2 11.5帮助显著加快运行查询的速度，支持流行的数据科学语言和框架，让实施自然语言查询更加容易。有这些路人工智能结合的技术：1.自然语言查询，提供更加直观的洞察。Db2 11.5提供更加类似于搜索引擎的数据集探索体验。由于包含IBMDb2 Augmented Data Explorer，因此通过自动补全、自然语言查询和分面搜索改进了编写查询。2.系统性能得到优化。Db2 11.5利用Workload Management自动管理资源并安排执行工作负载。这就支持根据服务分类、工作负载特征、所用时间、当日时间等，进行详细的资源分配和对工作负载进行监控和管理。3.无论数据位于何处，其可访问性都有所提高。通用SQL引擎(CSE)发现，在Db2 11.5内，支持更好地访问IBM和非IBM产品中的数据，这样用户就可以提供更完整的洞察。CSE的内置数据联合功能允许用户从Db2系列产品访问数据，比如IBM Db2 Warehouse、IBM Db2 Big SQL、IBM Integrated Analytics System以及现有的IBM Pure Data for Analytics(Netezza)。由于这些联合功能，我们改进了对Oracle、Teradata、Microsoft SQLServer、Amazon Redshift之类的云资源以及Hive之类的开源解决方案的数据访问。4.Db2能够像数据科学家一样工作。Db2 11.5支持流行的数据科学语言和框架，包括：Go、Ruby、Python、PHP、Java、Node.js、Sequelize、IBM Watson Studio和Jupyter Notebooks。5.查询运行速度显著加快。使用Db2 11.5，查询运行速度显著加快，因为它可以监控SQL随时间推移的性能信息，并使用机器学习算法将其与查询关联起来。这就允许针对特定SQL语句创建模型并进行优化。6.基于置信度的查询使结果更加准确。Db2 11.5还使用机器学习来监控SQL查询结果，根据它们的准确性对其进行评分。然后，它利用这些评分划分优先级，对结果进行重新排序。

人工智能的不断发展，各方面都有着突飞猛进的进步。数据库的研制充分证明了人工智能与数据相结合的重要表现，充分发挥出数据库为人们带来的方便性。人工智能已将纯理论性逐步进入实用的发展阶段，使人工智能表现出更深的渗透性，人们不断的学习，将学习的知识与结论充分的注入在人工智能当中，促使人工智能的使用范围更广，把其他领域中的新技术引入到人工智能的研制中，能大力推动人工智能的发展。