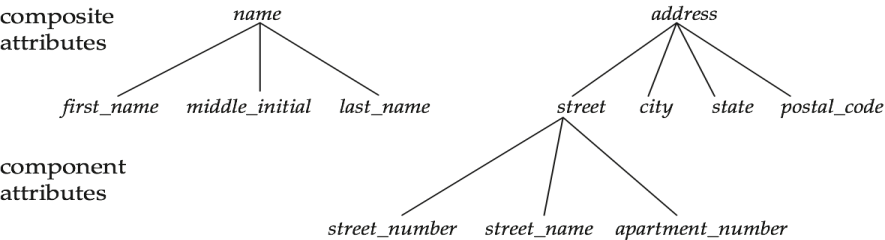
数据库复习

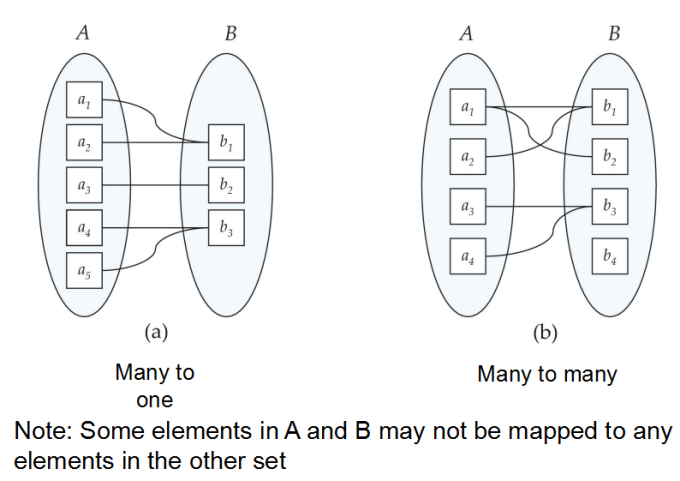
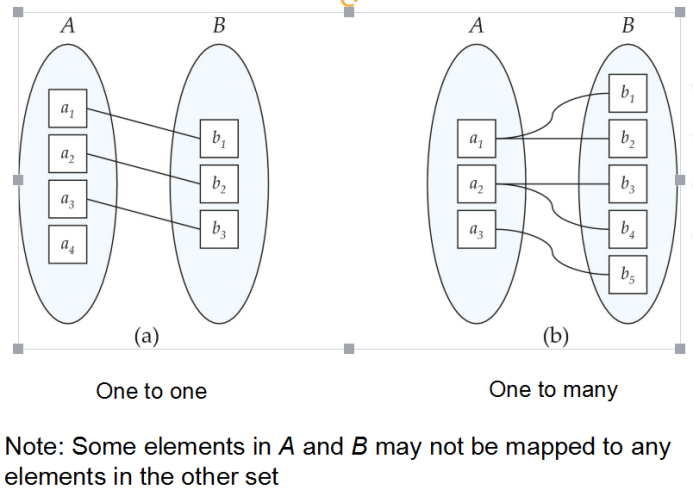
Ch7

1. relationship是许多entity之间的一种关系连接
2. Relationship set是relationship的集合
3. 一种属性也可以是relationship set的property
4. binary relationship:二元关系，数据库中大多实体之间的关系为二元关系，即relationship set为binary relationship，包括两个实体
5. Attributes:实体entity由一组属性表示，这组属性是实体集中所有成员共有的属性。

分类：简单/符合attributes 单值/多值attributes 衍生种类：年龄、日期



1. Domain:域，attributes的可以取到的值
2. binary relationship的映射种类

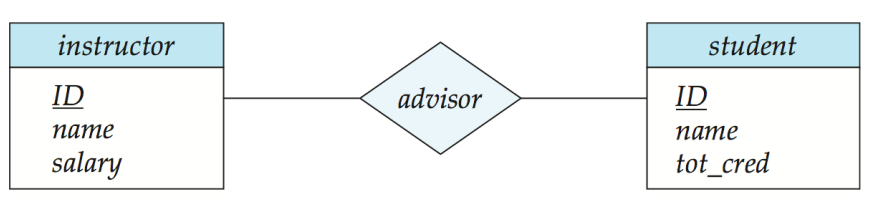


1. superkey:确定实体集中每一个实体的唯一性，可以为一个或多个attribute

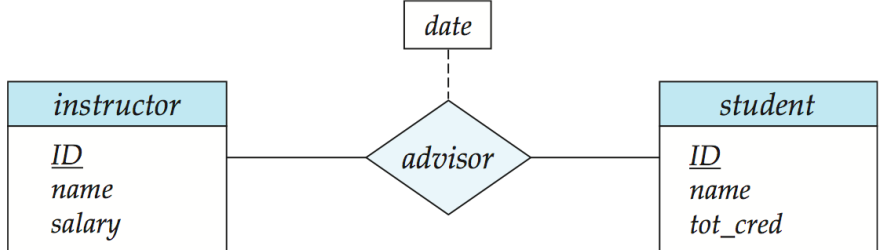
Candidatekey:是superkey中最小的子集

Primarykey：candidatekey中的一个

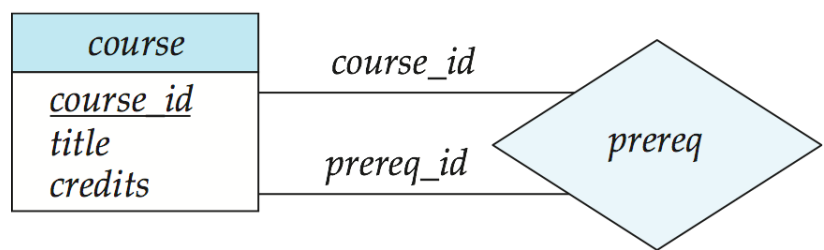
1. relationship set的superkey由参与relationship的两个实体集的primarykey组成。（这意味着两个实体集之间最多有一个relationship set）在确定candidatekey时要考虑relationship set的映射种类。在有多个candidatekey的情况下确定primarykey时要考虑relationship set的语义
2. E-R图 relationship set



1. 有atrribute的relationship set

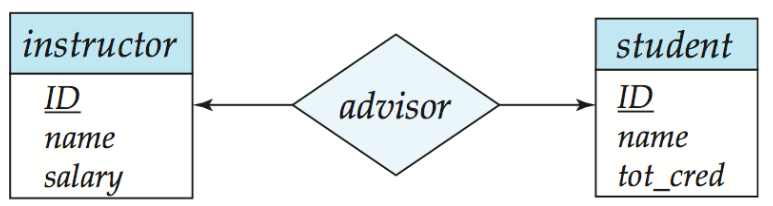


1. roles 有先决条件，只有一个entity与relationship set连接的ER图 course\_id和prereq\_id为角色标识

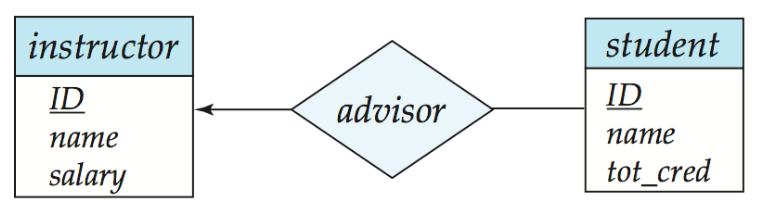


1. relationship set映射规则的ER图 箭头代表1 无箭头代表多

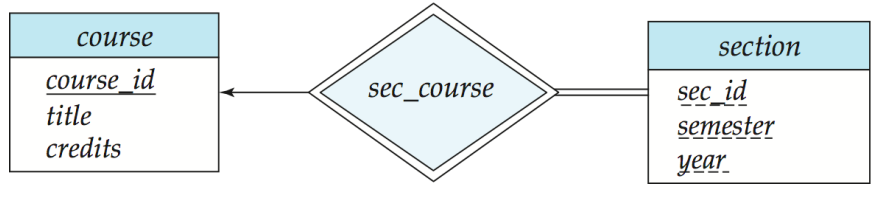
一对一



一对多：一个教授帮助多个学生

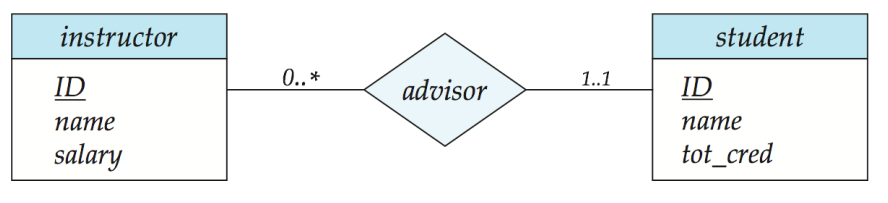


1. Total participation：实体集中所有实体都参与到relation中。每个section必须有一个course

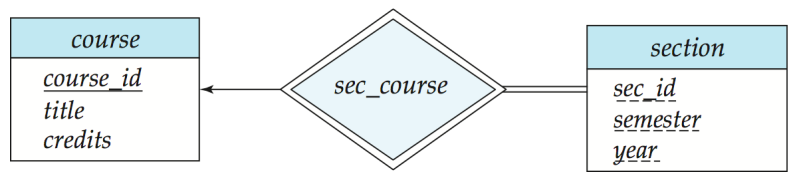


Partial participation与之相反

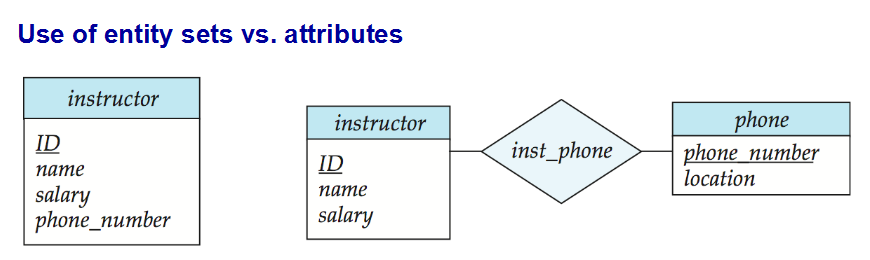
1. ER图也可以用这种方式表达



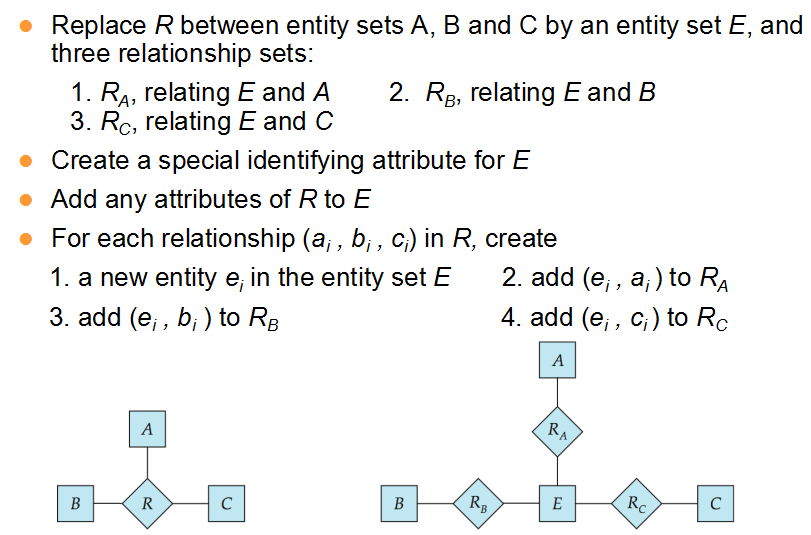
1. weak entity sets：没有primarykey 的entity sets.该实体集必定与一个identifying entity set相关，且relationship是total participation和一对多。只有包含对应的primary key后才会变为strong.
2. Identifying relationship：标识性联系，弱实体集必须与另一个称作标识的实体集关联才有意义，从弱实体集到标识性实体集是多对一的，并且弱实体集在参与中是total。Identifying relationship不应该有任何属性
3. Discriminator分辨符/partial key：相当于weak entity set的primary key。弱实体集的primary key由标识实体集的primarykey和他的discriminator组成。下图section为weak entity



1. 实体集entity set和关系集relationship set可以统一表示为表示数据库内容的关系模式relation schema。就是数据库中的表。其中弱实体集所构成的表，包含与其相关的强实体的primarykey
2. Multivalued attributes需要另一个新的table 包括multivalued 值和primary key，且该attributes不能作为foreign key.
3. Derived attribute：A（）可以从上面数据判断A的值
4. 对于每个实体集和关系集，都有一个惟一的模式，该模式被分配相应的实体集或关系集的名称。
5. 多对多的relationship table为中间的菱形，包含两个primary key。一对多和多对一不需要create table，只需有one的primary key 和many 的 attributes。一对一不需要relation table
6. Relation ship所构成的表包含两个实体的primary key
7. 一般情况下，连接弱实体集合与其所依赖的强实体集的联系集 的模式 是冗余的。
8. 模式的合并，在一对一的情况下，联系集的关系模式可以跟参与联系的任何一个实体集的模式进行合并。
9. 实体集（entity set）VS属性（attribute）的优势：可以记录关于属性的额外信息



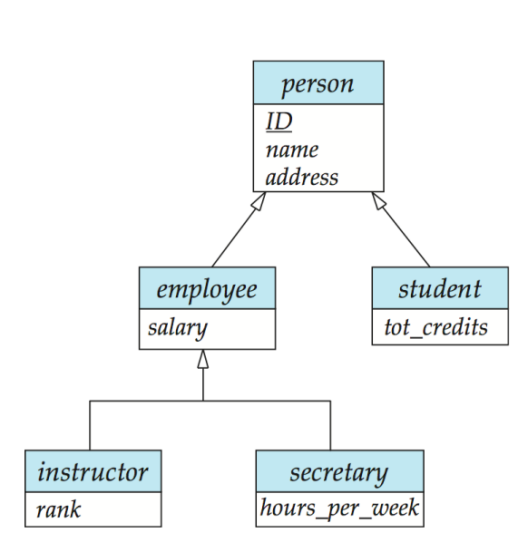
1. 实体集（entity set）VS关系集（relationship set）：描述实体之间的行为采用联系集。
2. 2元（binary versus）VSn元联系集（n-ary relationship sets）：数据库中通常都是二元，一些非二元实际上也可以用二元更好的表示，但会产生一些限制。比如为联系集而创建实体集时，要为其添加标识属性以区别其他同时建立得到实体集；3元约束转变可能无法转为2元；n元联系集可以更清楚表示几个实体集同时参与单个联系集
3. 非二元转二元



1. Extended E-R Features: Specialization（扩展的ER特性：特化）：一个实体集可根据多个可区分的特征进行特化。

Attribute inheritance：属性继承--较低级别的实体集继承了与其链接的更高级别实体集的所有属性和关系参与。

1. Extended ER Features: Generalization（扩展的ER特性：概化）：概化是一个高层实体集与多个低层实体集间的包含关系。
2. Specialization and generalization 概化和特化是彼此的倒置，在同一张ER图中表示。特化是自顶向下，概化是自底向上



1. 概化上的约束：condition-defined条件定义：所有低层实体集基于同一属性进行评估

user-defined用户定义：用户自己给出定义条件

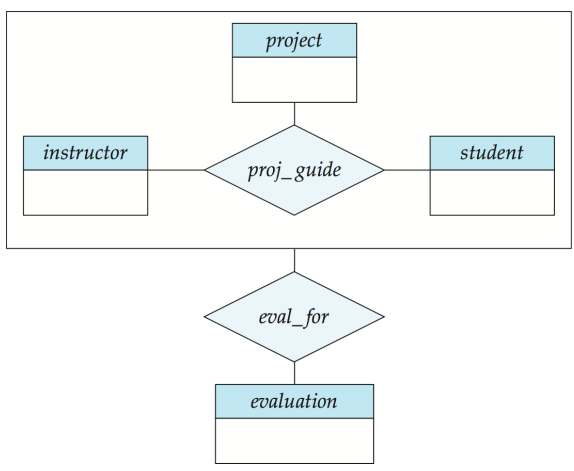
Disjoint：不相交，一个实体集最多有一个低层实体集，即只能满足一个条件，要么是研究生要么是本科生，不能都是

Overlapping：重叠，一个实体集可以属于多个高层实体集。Disjoint的相反

Completeness constraint total：每个高层实体集必须有一个低层实体集

Partial允许有的高层实体集无低层实体集

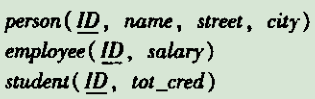
1. Aggregation：聚集：将一部分视为高层实体



1. Representing Specialization via Schemas：概化的表示可以用模式

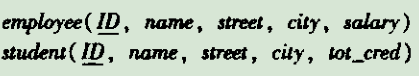
第一种是为每个实体集创建一个模式，包括高层实体集的主码和他们自己的属性。

缺点：获取信息时，雇员需要访问两个关系，一个对应于低级模式，另一个对应于高级模式。



第二种方法，如果概化是不相交而且完全的。即每个高层实体集只有一个低层实体集，则为每个低层实体集创建模式，高层实体集忽略。

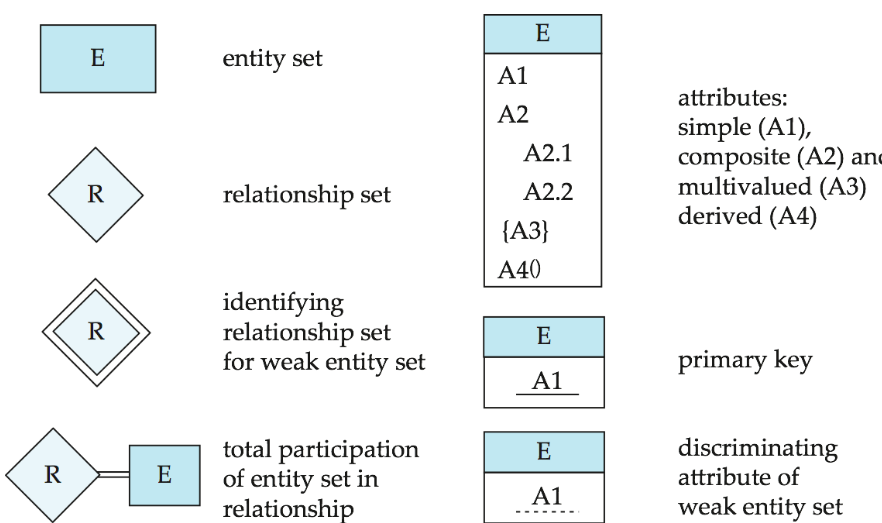
缺点：既是学生又是雇员的人可能会发生数据冗余，又存了一遍。如果既不是雇员又不是学生，则需要一个额外的模式。

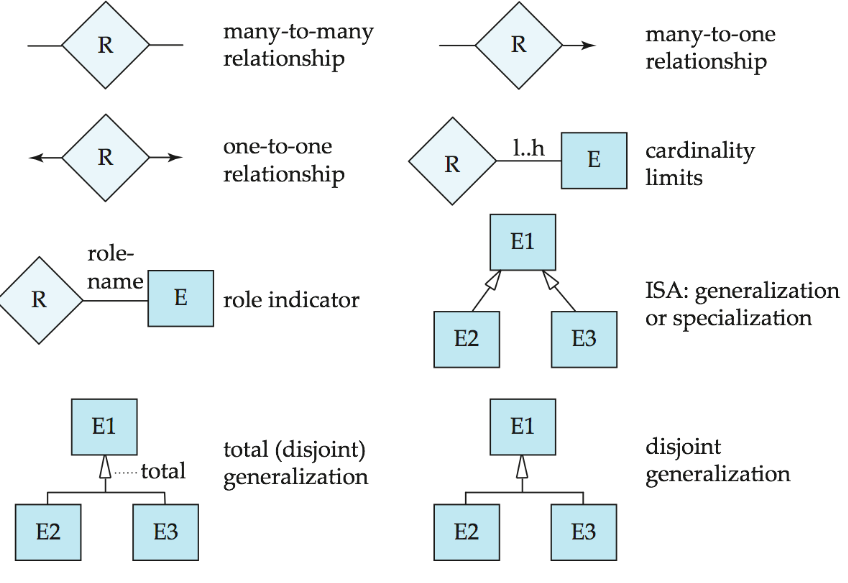


1. 聚集的表示：参见上面的图

eval\_for (s\_ID, project\_id, i\_ID, evaluation\_id)

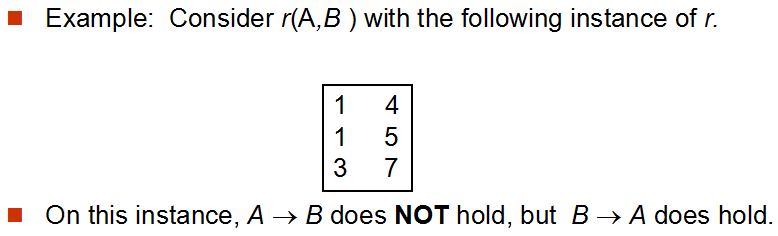
35.

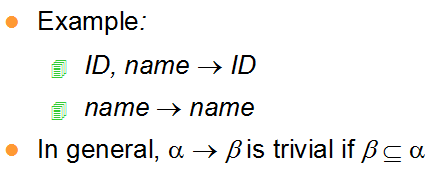


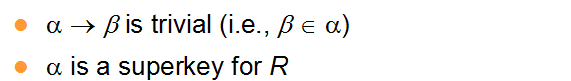


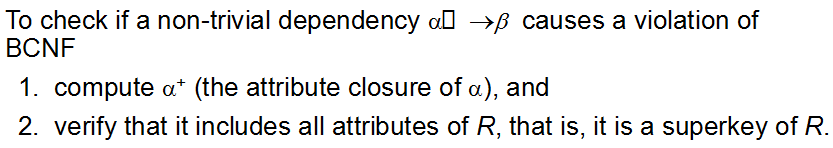
Ch8

1. 如果R的所有属性的域是原子的，则关系模式R是第一范式。如果它的元素被认为是不可分割的单位，则域是原子（atomic）的。
2. Functional Dependencies：某个值唯一确定另一个值



1. K是relation R 的super key当且仅当K→R
2. K是relation R 的 candidate key当且仅当 K→R 且对于任何在K内的属性，都不能→R
3. 如果relation r is legal under a set F of functional dependencies, we say that r satisfies F.
4. We say that F holds on R if all legal relations on R satisfy the set of functional dependencies F.
5. Trivial:存在α→β，它们在所有的关系中都是满足的（基于现实中的事实判断），函数依赖是trivial的。
6. Closure（闭包） of a Set of Functional Dependencies：f逻辑上隐含的所有函数依赖集是f的闭包。记为f+,f+是f的超集
7. BCNF：符合BCNF的relation R的函数依赖集（形如X→Y），函数依赖集中所有的X都是candidate key。所有X→Y满足以下一点即可



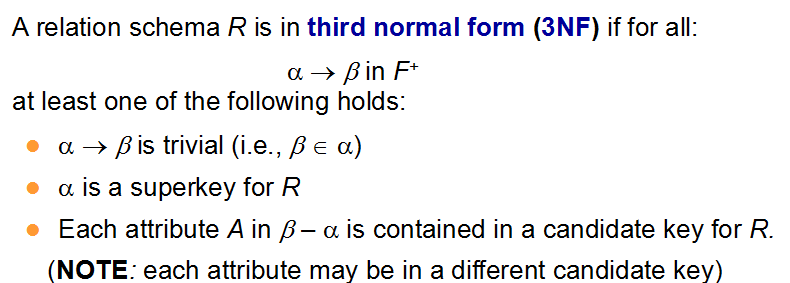


## 分解为BCNF方法：

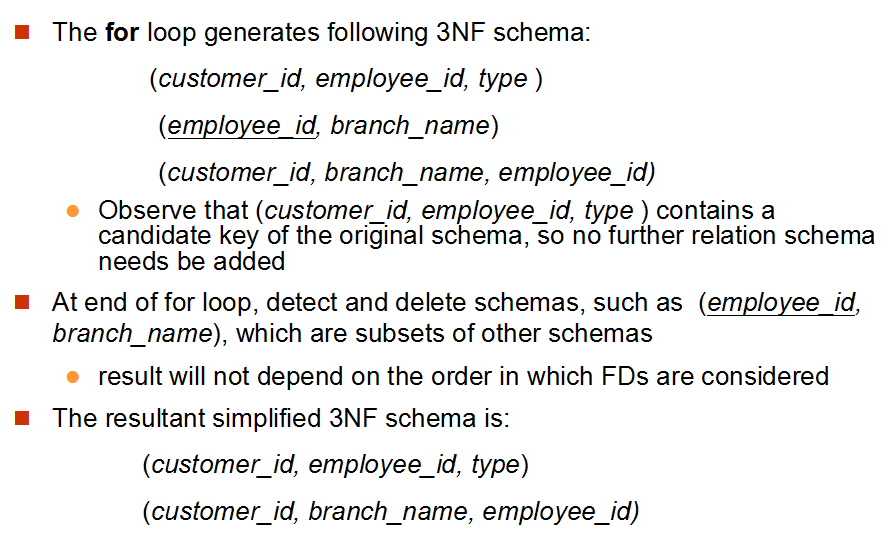
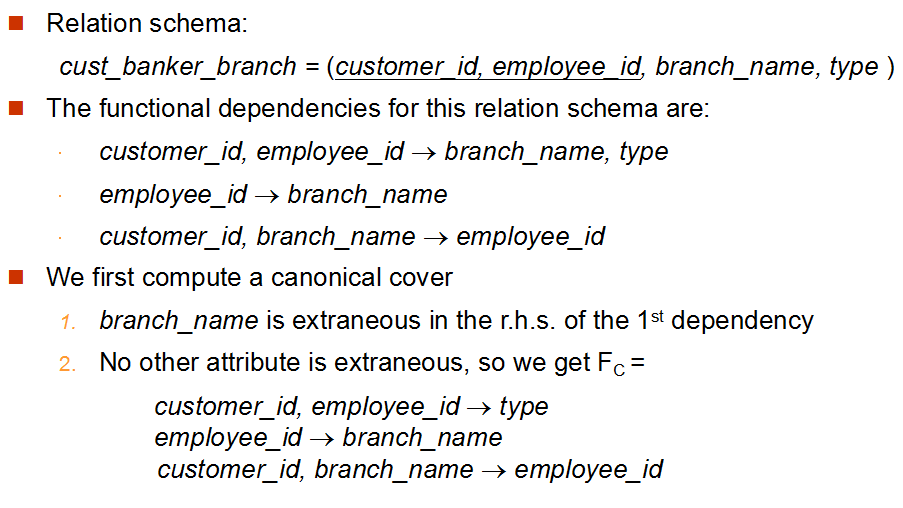
## 

1. 3NF：要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。简而言之，第三范式就是属性不依赖于其它非主属性。 满足第三范式（3NF）必须先满足第二范式（2NF）允许一些冗余

总有一种无损连接，保持依赖分解为3NF。



检测是否是3NF：只检查F中函数依赖，用属性闭包检查每一个α→β，如果α是super key。但如果α不是super key，要确定β中的每个属性包含在R的candidate key中

3NF分解例子：

它始终是可以分解成一组关系的关系，这样在3NF：

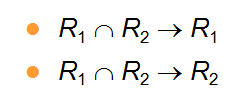
分解是无损的。

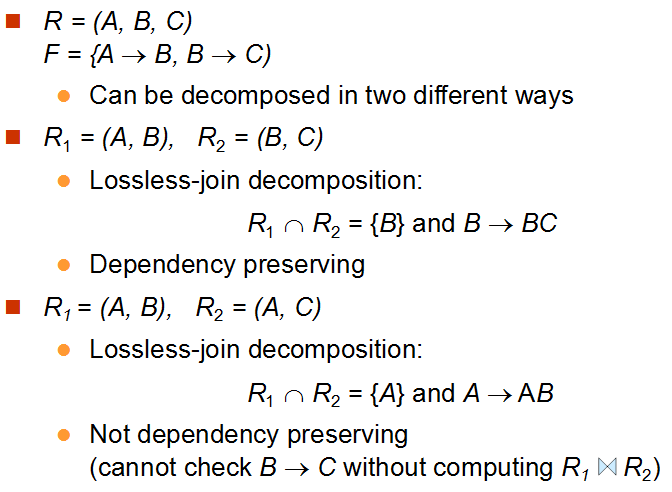
依赖关系被保留。

它始终是可以分解成一组关系的关系，这样在BCNF：

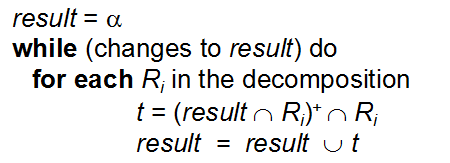
分解是无损的。

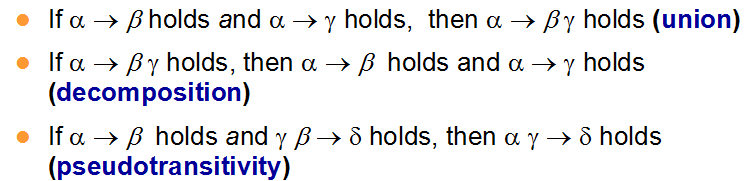
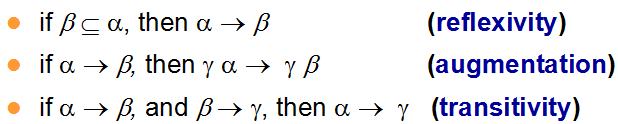
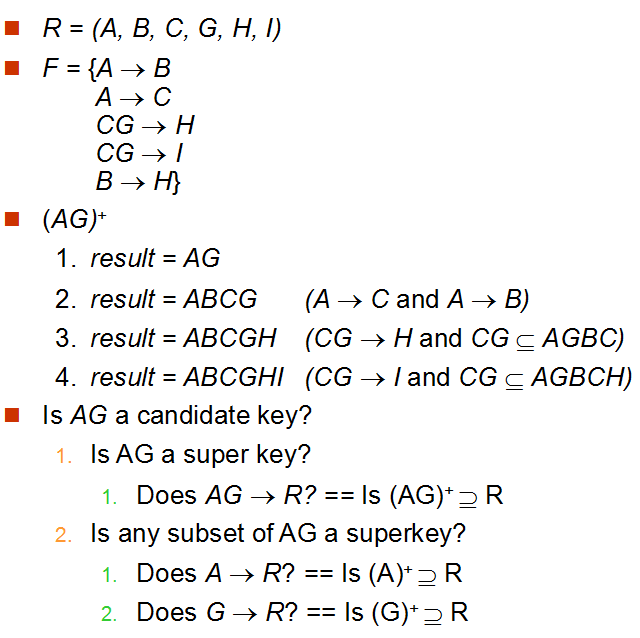
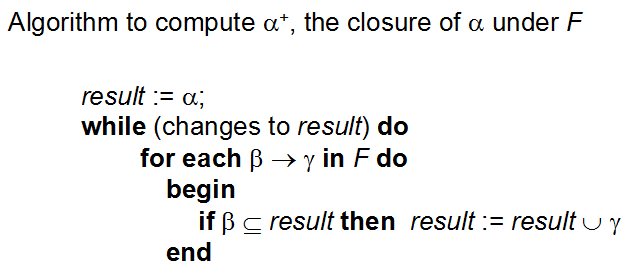
不一定保留依赖关系。

1. 如果一个relation 是BCNF 那它就是3NF
2. 无损分解（lossless decomposition）能确保表的分割不会引起信息丢失。设关系模式R与函数依赖集F，R分解为R1，R2，此分解是无损的必须满足：α=R1∩R2是R1或R2的超码，可利用属性集闭包验证。满足其中一条



1. 依赖-保持分解（dependency-preserving decomposition）能确保联系不丢失。

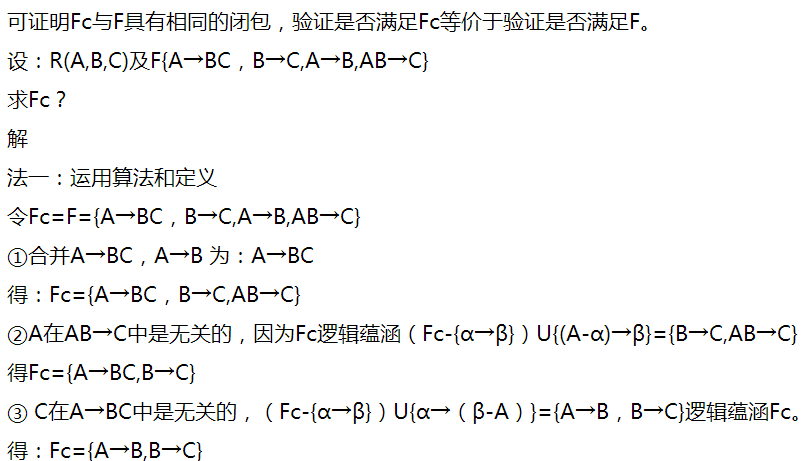
如果F上的每一个函数依赖都在其分解后的某一个关系上成立，则这个分解是保持依赖的（这是一个充分条件）。   
如果上述判断失败，并不能断言分解不是保持依赖的，还要使用下面的通用方法来做进一步判断。对于α→β这里的属性闭包是在函数依赖集F下计算出来的。如果result中包含了β的所有属性，则函数依赖α→β。分解是保持依赖的当且仅当上述过程中F的所有依赖都被保持

1. 计算函数依赖的闭包：
2. 给定一组属性A，定义一个f下的闭包（用+表示）作为一组由f下的函数决定的属性。
3. 属性闭包（attributes closure）的作用：

检验superkey：为了检验α是不是superkey，计算α+，并且看α+是否包含了R所有属性

检验函数依赖：为了检验α→β是否成立，则检验β是否属于α+，就是看α+里面有没有β

1. Canonical Cover（最小函数依赖集）（也叫正则覆盖）：F的正则覆盖Fc是一个依赖集，使得F与Fc相互逻辑蕴涵。此外，还包含如下性质：   
   ●Fc中任何函数依赖都不含无关属性。   
   ●Fc中函数依赖左半部分都是唯一的。即，Fc中不存在α1→β1，α2→β2，满足α1=α2。



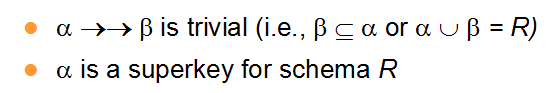
1. &无关项（extraneous attribute）定义：若果去除函数依赖中的属性不会改变函数依赖的闭包，就称该属性是无关的。   
   形式定义：   
   ●如果A∈α并且F逻辑蕴涵（F-{α→β}）U{(A-α)→β},则A在α是无关的   
   ●如果A∈β并且函数依赖{F-{α→β}）U{α→（β-A）}逻辑蕴涵F，则A在β是无关的

检验方法：   
令R为一关系模式，F是在R上成立的给定的函数依赖集。考虑依赖α→β上的属性A。   
●如果A∈α，令λ=α-{A}，并且计算λ→β是否可以由F推出。 能推出则是无关的  
●如果A∈β，F’=（F-{α→β}）U{α→（β-A）}，并检验α→A是否能够由F’推出。能推出则无关

1. Multivalued Dependencies（多值依赖）：α→→β。如果Y→Z 则Y→→Z

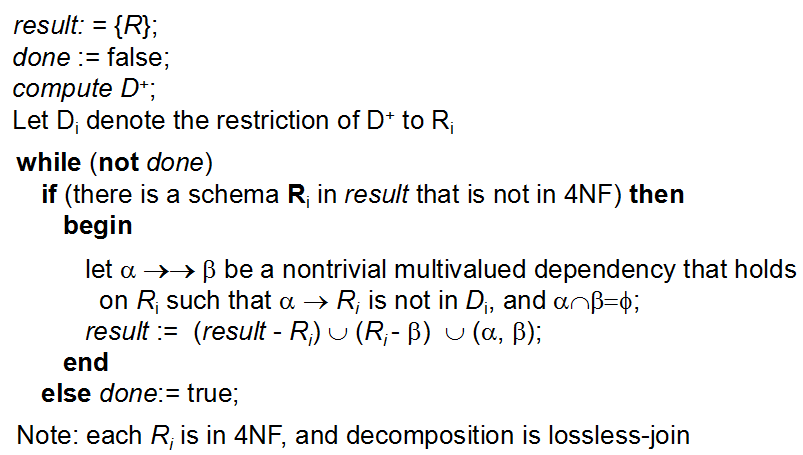
如果一个关系R未能满足给定的多值依赖，我们可以构建一个关系R’也加入元组使R满足多值依赖

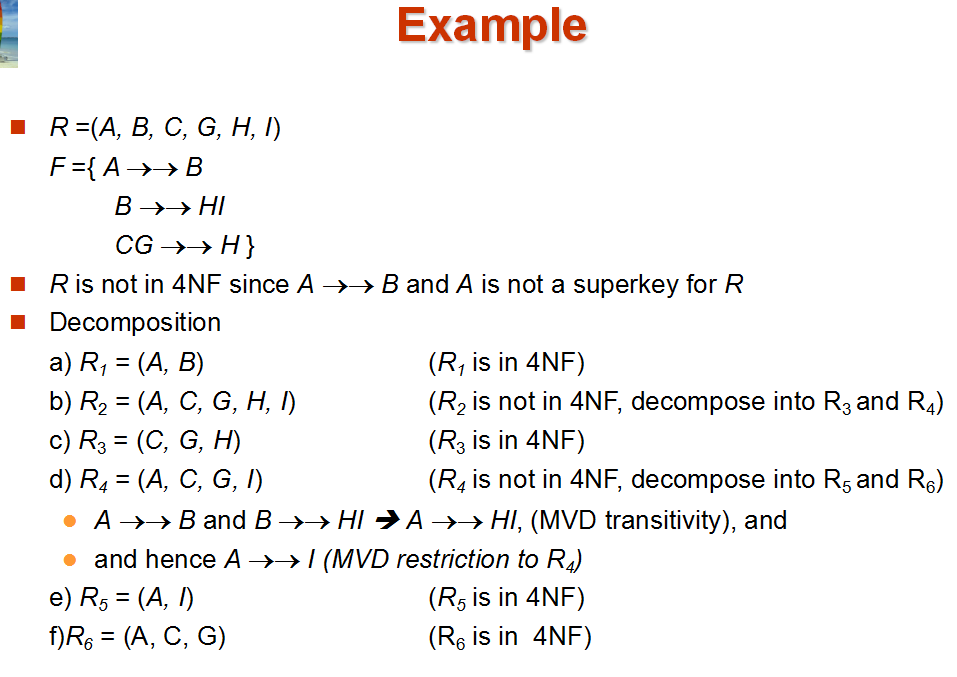
21.4NF：满足以下任意一点。是限制关系模式的属性之间不允许有非平凡且非[函数依赖](http://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E4%BE%9D%E8%B5%96/737239" \t "http://baike.baidu.com/_blank)的多值依赖



如果一个relation是4NF，那她就是BCNF

分解为4NF



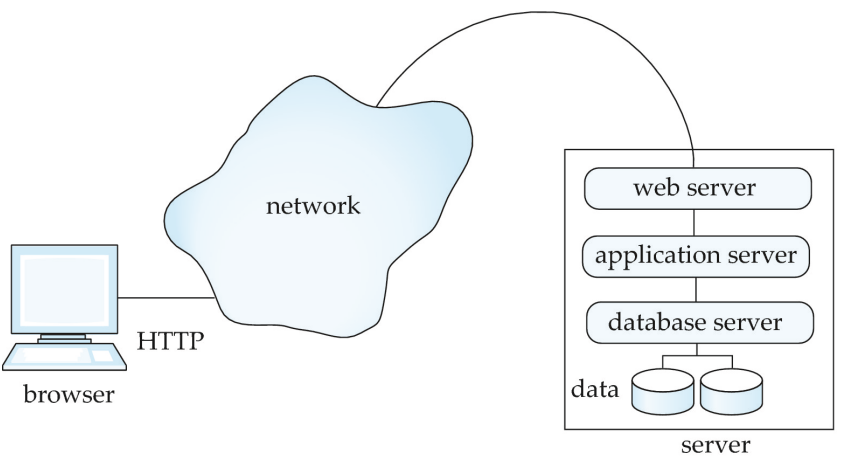


Ch9

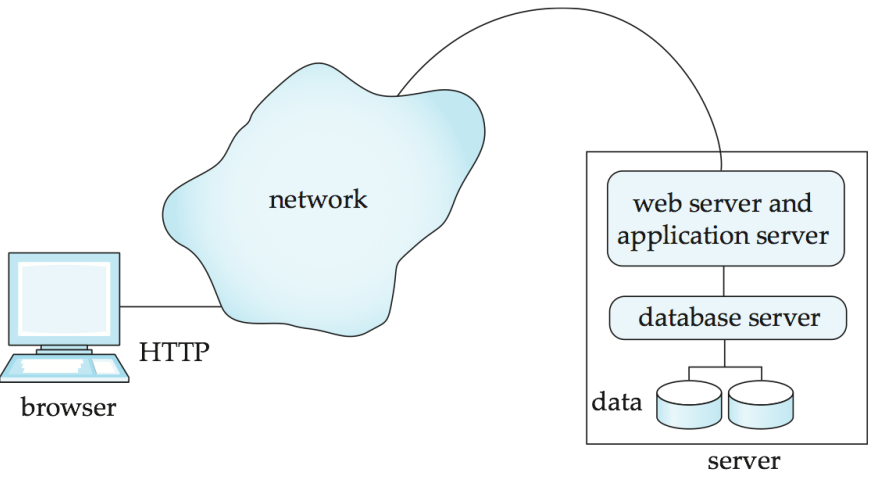
1. 在网络中，功能的指针是由统一资源定位器（URL）提供

例子：IMG_256

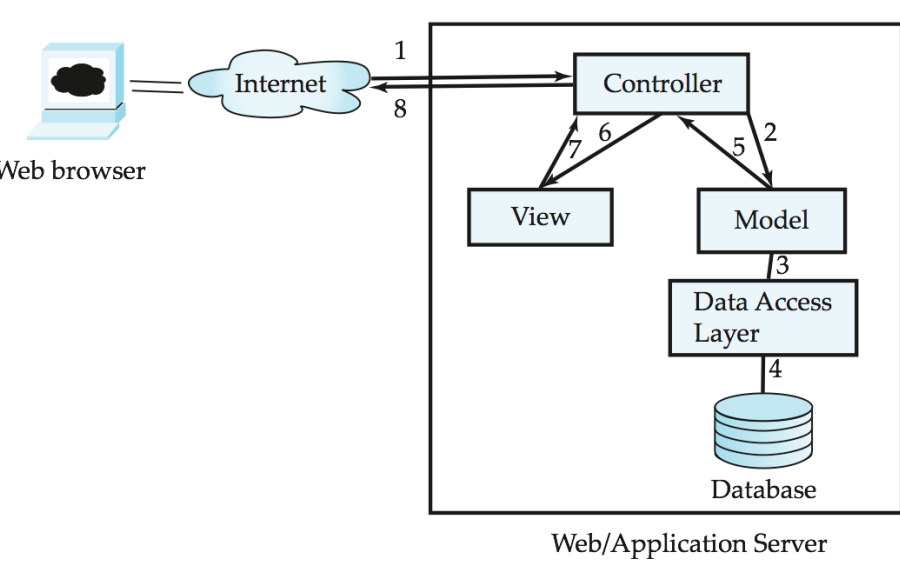
1. URL的第一部分表示文档如何被访问，http表示文档将用超文本传输协议（HyperText Transfer Protocol）第二部分给出一台具有Web服务器的机器的名称。URL的其余部分是文件在机器上的路径名。
2. Web Servers（Web服务器）：是运行于服务器上的程序。它接受来自Web浏览器的请求，并以HTML文档的形式返回结果。浏览器和Web服务器通过HTTP通信。
3. HTML提供了定义将超链接与表单功能相结合的界面的能力，WEB浏览器通过HTTP协议与WEB服务器通信
4. CGI（公共网关接口）：定义了Web服务器如何与应用程序通信。
5. 图：三层 Web architecture



1. 两层WEB architecture.三层的多级服务增加了系统开销，CGI为每个请求开启一个新进程为之服务开销更大。



1. The HTTP protocol is connectionless（HTTP协议是无连接的）：原因在于大多数计算机所能同时容纳的连接数目有限。使用无连接协议，当满足请求后立马断开，为其他请求留出可用连接。
2. Cookie（网络跟踪器）：为了实现会话，需要在客户端存储额外的信息，服务器用这个额外的信息来辨别出请求是会话的一部分。这个额外信息通常以cookie形式保存在客户端。P223
3. Server-Side Scripting（服务器端脚本）：脚本语言提供了可嵌入HTML文档中的结构。
4. Client Side Scripting（客户端脚本）：文档中程序代码的嵌入允许WEB页是活动的，通过在本地执行程序以实现活动。主要用于与用户的灵活交互，加快了交互的速度，但存在安全性问题。
5. Application Architectures：
6. Presentation or user interface（展示或用户界面层）：如MVC（模型-视图-控制器），模型对应于逻辑层，视图为数据的显示，控制器接受事件，在模型上执行操作，并返回一个视图给用户
7. business-logic layer（业务逻辑层）：提供对数据和数据操作的高级视图。
8. data access layer（数据访问层）：提供业务逻辑层和底层数据库的接口。



1. 提高WEB服务性能：利用缓存技术Caching techniques减少开销。
2. Application Security（应用程序安全性）：SQL Injection SQL注入，Cross Site Scripting（跨站点脚本）Password Leakage（密码泄露）Application Authentication（应用程序认证）Application-Level Authorization（应用级授权）Audit Trails（审计追踪）
3. 当数据库授权条款没有提供足够的保护时，数据可能会被加密。
4. 一个好的加密技术有如下特征：对于授权用户，加密和解密数据相对简单；加密模式不应依赖算法的保密，而应依赖被称作为加密秘钥的算法参数，该密钥用于加密数据；对入侵者来说，即使当他已经获得加密数据的访问权限了，确定解密密钥是困难的
5. 数据加密标准（DES）替代字符和重新排列顺序的基础上的加密密钥，这是提供给授权用户通过安全机制。方案不比密钥传输机制更安全，因为密钥必须共享。
6. 高级加密标准（AES）是一种新的标准取代DES，和基于Rijndael算法，但也依赖于共享密钥。

Ch10

1. 多种存储介质：

高速缓冲存储器（cache）：高速缓冲存储器是最快最昂贵的存储介质。一般很小，由计算机系统硬件来管理使用。不稳定volatile

主存储器（main memory）：不稳定，存储很快，但对于存储整个数据库来说还是太小

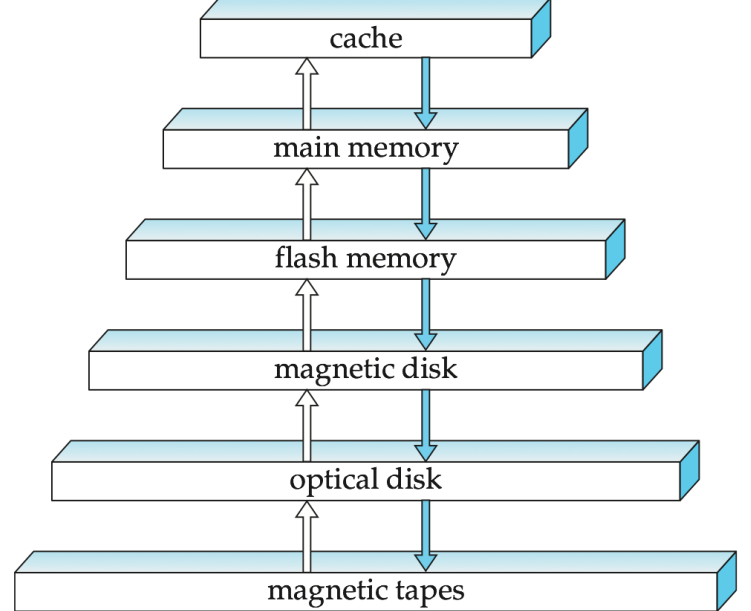
闪存（flash memory）：稳定，电源关闭时数据也可以保存下来。数据只能在一个位置写入，但位置可以被擦除并重新写入。读起来差不多和主存一样快。广泛应用于嵌入式设备，如数码相机、电话、USB钥匙等。

磁盘存储器（magnetic-disk storage）：用于长期联机数据存储的主要介质是磁盘。容量和成本比主存闪存大

光学存储器（optical storage）：光学存储器最流行的方式是光盘。只读光盘是不可写的，读写比磁盘慢。

磁带存储器（Tape storage ）：非易失性，顺序访问，速度比磁盘慢，容量大便宜

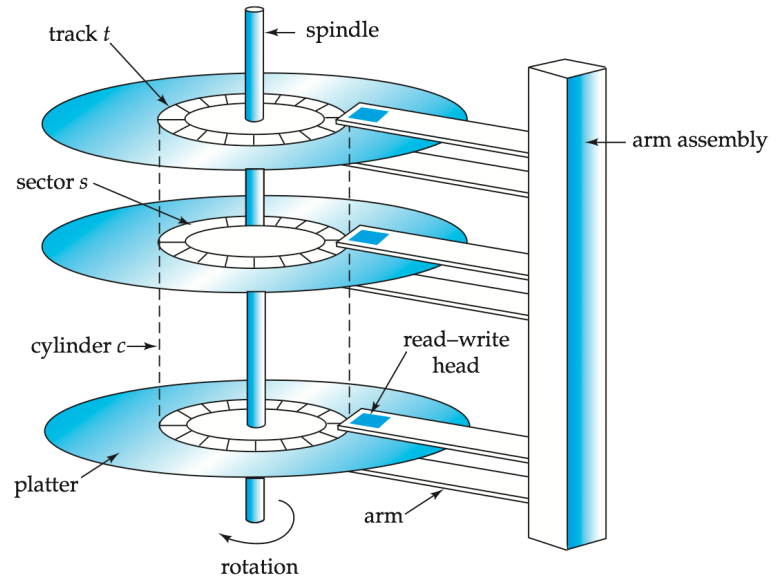
1. 存储层次Storage Hierarchy：层次越高，速度越快，价格越高



1. primary storage（基本存储）：最快的存储介质（如cache和主存）

下一层为secondary storage或online storage（辅助存储或联机存储）磁盘

最底层介质为tertiary storage（脱机存储）磁带、光盘

1. 

Read-write head读写头：位置非常接近盘片表面（几乎触摸它）读取或写入磁编码信息。

Tracks 磁道 盘表面为磁道

Spindle 主轴

Sector 扇区 磁道被分为多个扇区 扇区是可以读取或写入的最小数据单位。

Cylinder 柱面

盘面 platter

读/写扇区：磁盘臂摆动到位置头在正确的轨道上，盘片不断旋转；数据读取/写入扇区通过磁头下。

1. 磁盘控制器（disk controller）：计算机系统与磁盘驱动器硬件之间的接口。接受高级命令读取或写入扇区。然后启动动作，例如将磁盘臂移动到正确的轨道上，并实际读取或写入数据。计算校验和与各扇区确认数据回读正确，如果数据被损坏，以很高的概率计算校验和存储的校验和不匹配
2. 磁盘通常可以通过电缆直接与计算机系统的磁盘接口相连，也可以放置到远端通过高速网络与磁盘控制器相连。
3. 在存储区域网络（SAN）中，大量磁盘通过高速网络连接到若干服务器。
4. 在网络附加存储（NAS）网络化存储中，使用网络化文件系统协议提供文件系统接口，而不是提供磁盘系统接口。
5. 磁盘性能度量：访问时间（access time）平均故障时间Mean time to failure (MTTF)数据传输率（Data-transfer rate）
6. 访问时间（access time）：从读取或写入请求发出到数据传输开始时所花费的时间。包括寻找时间（seek time）-把磁盘臂重新定位在正确的轨道上的时间。平均查找时间为1/2最坏情况下查找时间。

旋转延迟Rotational latency：读写头到达所需的磁道，等待访问扇区出现在读写头下所花费的时间。平均延迟是最坏情况潜伏期的1/2。

1. 数据传输率（Data-transfer rate）：从磁盘获得数据或向磁盘存储数据的速率。
2. 平均故障时间Mean time to failure (MTTF)：系统无故障连续运行的时间
3. Optimization of Disk-Block Access磁盘访问的优化：磁盘臂调度算法Disk-arm-scheduling、文件组织File organization非易失性写缓冲Nonvolatile write buffers日志磁盘Log disk
4. 块（block）-单一磁道中连续的扇区序列，数据以块形式在磁盘和主存之间传输
5. 磁盘臂调度算法Disk-arm-scheduling：特定轨道访问，使磁盘手臂运动最小化。通常用电梯算法
6. 文件组织File organization：为了减少访问块的时间，可以按照与预期的数据访问方式最接近的方式来组织磁盘上的块。
7. Nonvolatile write buffers 非易失性写缓冲：通过将块写入非易失性ram缓冲区来加快磁盘写入速度。

非易失性RAM：电池备份RAM或闪存，即使电源失败，数据也是安全的，当电源返回时将写入磁盘。

18日志磁盘Log disk：专门用于编写块更新的连续日志的磁盘。与非易失性ram完全相同

写入日志磁盘非常快，因为不需要查找。不需要特殊的硬件（NV-RAM）

19.NAND闪存：广泛用于存储，快闪中的页和磁盘中的扇区十分相似，因为它比NOR闪存便宜得多，拥有更高存储量。

固态磁盘：使用多个闪存设备，提供更高的传输速率100到200兆字节/秒。

擦除速度很慢（1到2 millisecs）

擦除块包含多个页面，映射逻辑地址到物理地址的网页页面避免等待擦除

1. 数据库作为文件集合存储。每个文件都是一系列记录。记录是一系列字段。
2. Free list(空闲列表)：将第一个已删除记录的地址存储在文件头中。使用这个第一个记录来存储第二个删除记录的地址，等等，可以将这些存储的地址看作指针，因为它们指向记录的位置。
3. Variable-Length Records（变长记录）：可变长度记录以多种方式出现在数据库系统中：多个记录类型在一个文件中存储。

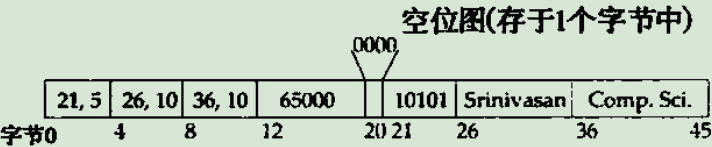
允许一个或多个字段是变长的记录类型。

允许重复字段的记录类型（在一些旧的数据模型中使用）。

属性按顺序存储。

有定长属性，如日期、数字，分配存储他们的值所需的字节数。

用（偏移量，长度）表示可变长度属性。偏移量表示在记录中该属性的数据开始的位置，长度表示属性数据的字节长度、

NULL值位图表示的空值。

23.Organization of Records in Files文件中记录的组织：

Heap堆-一个记录可以放在文件中任何有空间的地方。记录没有顺序

Sequential顺序-按每个记录的搜索键的值按顺序存储记录。

Hashing散列——在每个记录的某个属性上计算散列函数；散列函数结果指定记录应该放在哪个文件块中。

每个关系的记录可以存储在一个单独的文件中。在一个多表聚簇文件组织，几个不同关系的记录存储在同一个文件中。

1. 顺序文件组织在插入删除后，可能造成物理搜索码顺序与物理顺序不同，这时需要重组，使得他再一次在物理上顺序存放，但代价很高。
2. Multitable Clustering File Organization（多表聚类文件组织）：在一个块中存储两个或更多个关系的相关记录的文件结构。
3. Data Dictionary Storage（数据字典存储）:数据字典（也称系统目录）存储元数据(metadata)；也就是关于数据的数据.如信息的关系、名字的关系、每个关系属性的名称、类型和长度、视图的名称和定义、完整性约束、统计和描述性数据、每个关系中的元组数、物理文件组织信息、关系是如何存储的（顺序/散列/…）、关系的物理位置

Ch11

1. 索引机制Indexing：用于加速访问所需数据
2. Search key搜索键：用于在文件中查找记录的属性或属性集
3. 索引文件由表单的记录（称为索引条目）组成。
4. 两种索引结构：顺序索引Ordered index 散列索引Hash index
5. 顺序索引：基于值的顺序排序
6. 散列索引：基于将值平均分布到若干散列桶中。一个值所属的散列桶由散列函数决定。
7. 索引结构评价指标：访问类型（access type）访问时间（access time）插入时间（intsertion

Time）删除时间（deletion time）空间开销（space overhead）

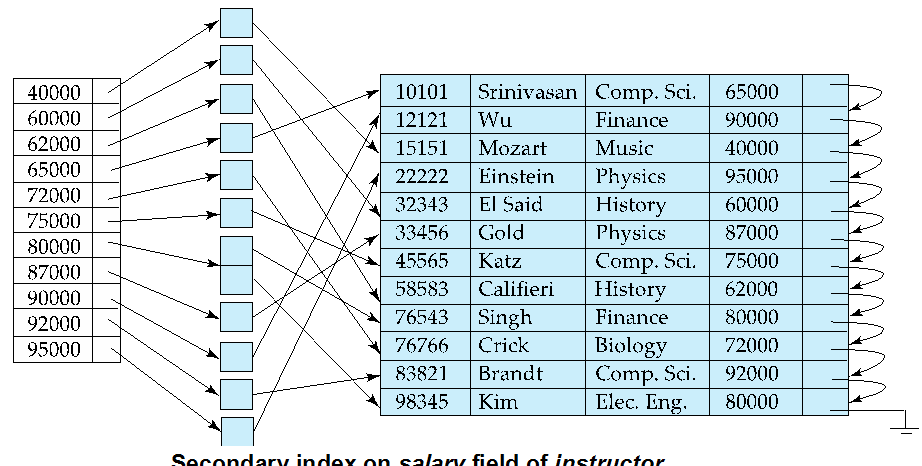
8.顺序索引：在顺序索引中，索引项按搜索键值存储排序。例如，图书馆的作者目录。

主索引（primary index）：在顺序排序的文件中，索引的搜索键指定文件的顺序顺序。也称聚类索引

主索引的搜索键通常是主键，但不一定是主键。

二级索引Secondary index：其搜索键指定与文件的顺序不同的顺序。也称为非聚集索引non-clustering index。

索引顺序文件：带有主索引的有序顺序文件

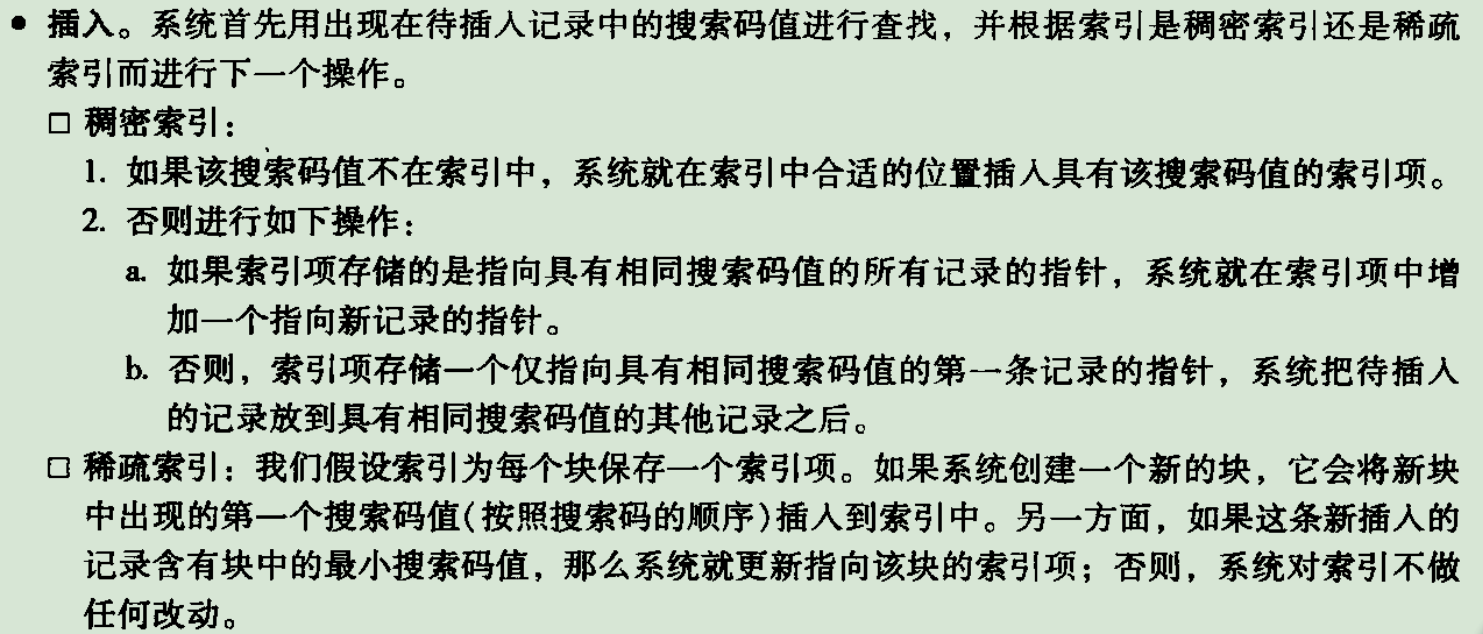
1. 两类顺序索引：稠密索引Dense Index、稀疏索引Sparse Index
2. 稠密索引Dense Index：文件中每个搜索码值都有一个索引项。在稠密聚集索引中，索引项包括索引码值以及具有该值的第一条数据记录的指针，具有相同搜索码值的其余记录顺序存储在第一条记录之后。稠密非索引中，索引必须指向所有具有相同搜索码值的记录。P278
3. 稀疏索引：Sparse Index只为搜索码的某些值建立索引
4. 稀疏索引与稠密索引相比：插入和删除占用更少的空间和较少的维护开销。通常比密集索引定位记录慢。
5. 次级索引Secondary indices：索引记录指向一个桶，其中包含指向具有特定搜索键值的所有实际记录的指针。次级索引必须是稠密的。
6. 多级索引Multilevel Index：如果主索引不适合于内存，访问将变得昂贵。解决方案：将主要索引保存在磁盘上作为顺序文件，并在其上构建稀疏索引。

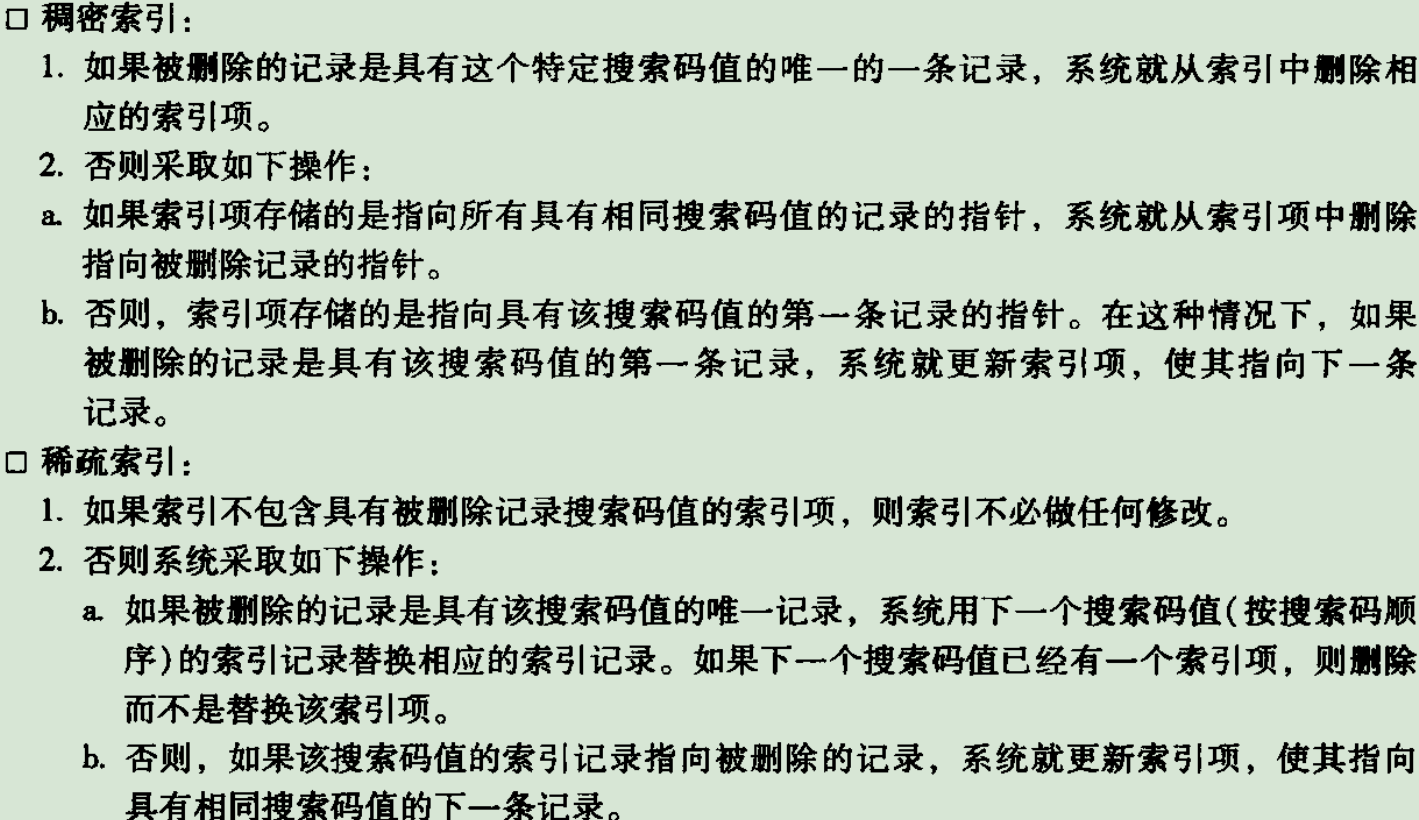
外部索引——主要索引的稀疏索引

内部索引——主要索引文件

如果外部索引太大，不能适应主存，那么就可以创建另一级索引，等等。

所有级别的索引必须在文件的插入或删除中更新。

15.



15.索引顺序文件的缺点：性能随着文件的增长而降低，因为许多溢出块被创建。需要对整个文件进行周期性重组。

16.B+树索引文件的优点：在插入和删除时自动地重组小的、局部的、变化的。不需要对整个文件进行重组以保持性能。

17.B+树的缺点：额外的插入和删除开销，空间开销。

B+树的优点大于缺点

B+树被广泛使用。

1. B+tree 每个叶节点最多有n-1个值（n为一个节点的指针数），最少有（n-1）/2个值。非叶子节点必须有n/2到n个子节点。根节点子节点树为2到到m个 m为树的阶
2. B+tree 的查找：在B+树，不管查找成功与否，每次查找都是走了一条从根到叶子结点的路径，若非终端结点上的关键码等于给定值，并不终止，而是继续向下直到叶子结点。
3. 对B+TREE 的插入与删除：<https://wenku.baidu.com/view/8aaf9d9bf8c75fbfc67db22c.html>

Ch12

Basic Steps in Query Processing查询处理的基本步骤：1. Parsing and translation句法分析和翻译：解析和翻译，将查询转换为内部表单。然后将其转换为关系代数。解析器检查语法，验证关系

2. Optimization优化，在所有同等评价方案中，选择成本最低的评价方案。

3. Evaluation评价，查询执行引擎接受一个查询评估计划，执行该计划，并返回查询的答案

Ch14

1. Transactions事务：事务是访问并可能更新数据项的一个程序执行单元 有以下ACID性质
2. Automicity requirement事务的原子性：事务内的操作要么全执行，要么全不执行，如果开始执行的事务因为某些原因失败，事务对数据库造成的任何修改都要撤销。
3. Consistency requirement事务的一致性：没有其他事务并发执行的情况下，保持数据库一致性
4. Durability requirement 耐久性要求：一个事务完成后，对数据库的修改是永久的，即使系统故障。
5. Isolation requirement 隔离性要求：每个事务感觉不到其他事务在并发执行
6. Transaction State事务的状态：

Active活动——初始状态；事务在执行时保持在该状态中。

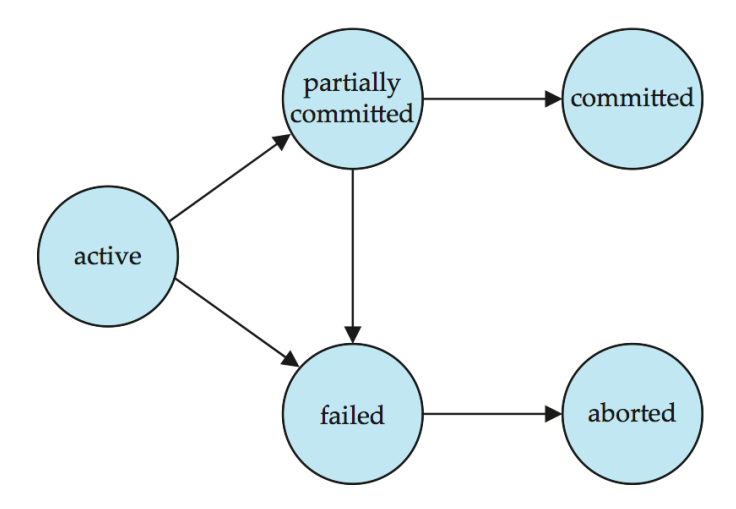
Partially committed部分执行——在最后语句被执行后。

Failed失败——在发现正常执行不能继续进行之后。

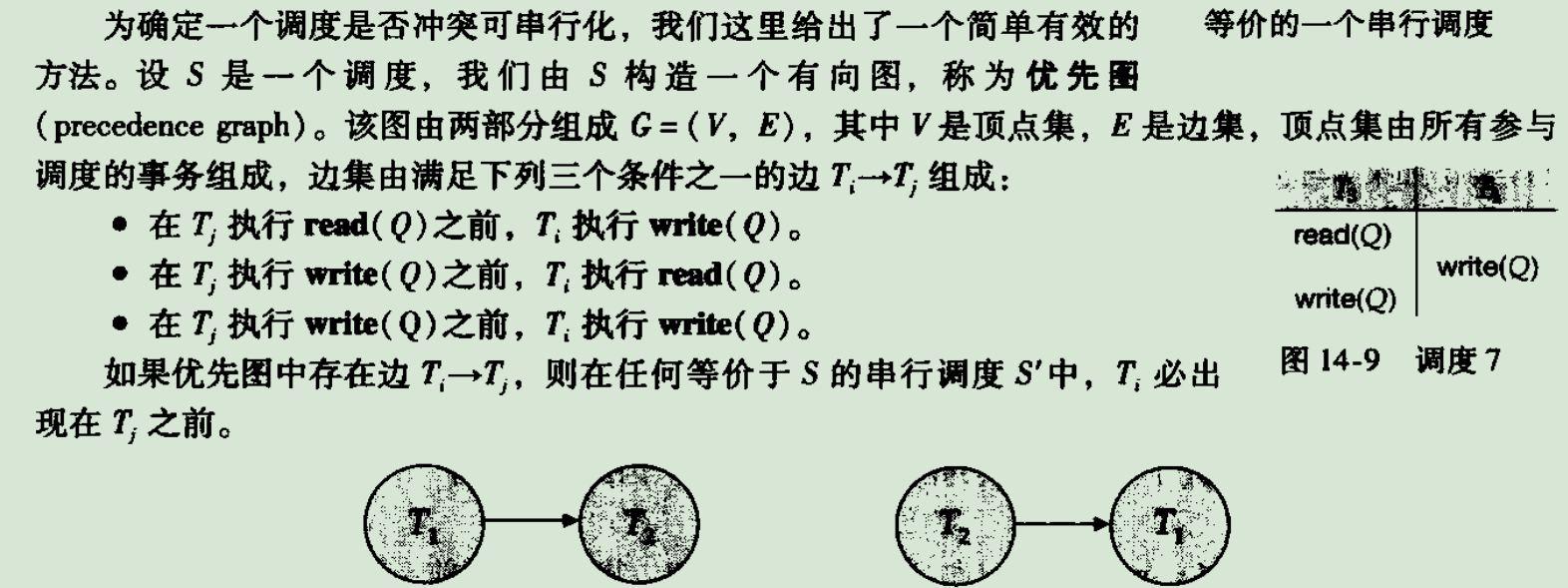
Aborted中止-在事务回滚之后，数据库在事务开始之前还原到其状态。中止后的两个选项：

重启事务（只有在没有内部逻辑错误的情况下才可以这样做）杀了事务

Committed 提交的-成功完成后。



1. Concurrent Executions并行执行：允许在系统中同时运行多个事务。优点是：增加处理器和磁盘利用率，从而导致更好的事务吞吐量；减少等待时间
2. Serializability可串行化：



1. conflict equivalent冲突等价：Schedule S 是冲突可串行化如果冲突等同于一个串行调度

如果schedule S可以转化为一个表的S´由一系列非冲突的指令互换，我们说的是冲突S的S´等价。

1. 要判断冲突可串行化，需要构造优先图Precedence graph 并调用一个环检测算法。无环则冲突可串行化。
2. 优先图测试不能直接用于视图可串行化
3. Recoverable Schedules可恢复调度：对于事务TI TJ，如果TJ读取了TI所写入的数据，则TI先于TJ前commit，否则为不可恢复调度
4. Cascading Rollbacks：级联回滚：因单个事务故障，导致依赖该事务的其他事务回滚。
5. Cascadeless schedules：无级联调度：对于事务TI TJ，如果TJ读取了TI所写入的数据，则TI先于TJ读之前commit 无级联调度也是可恢复调度
6. Concurrency Control并发控制：
7. 数据库必须提供一种机制，确保所有可能的时间表都是冲突或视图可串行化，且是可恢复的和无级联的
8. SQL标准的隔离性级别：

Serializability 可串行化--默认

Repeatable read可重复读取--只能读取已提交的记录，同一记录的重复读取必须返回相同的值。

Read committed只能读取已提交的记录，但记录的连续读取可能返回不同的（但提交的）值。

Read uncommitted 读未提交-甚至未提交的记录可能被读取。

专英：

ternary relationship：三元关联

Tuple Relational Calculus：元组关系演算

Identifying relationship：标识性联系

Aggregation：聚集

dependency-preserving decomposition：依赖-保持分解

Multitable Clustering File Organization：多表聚族文件组织

Cookie：网络跟踪器

CGI：公共网关接口

Application Security（应用程序安全性）：SQL Injection （SQL注入）Cross Site Scripting（跨站点脚本）Password Leakage（密码泄露）Application Authentication（应用程序认证）Application-Level Authorization（应用级授权）Audit Trails（审计追踪）

Read-write head：读写头

Tracks ：磁道，盘表面为磁道

Spindle：主轴

Sector：扇区 磁道被分为多个扇区 扇区是可以读取或写入的最小数据单位。

Cylinder：柱面

Platter：盘面

Presentation or user interface：展示或用户界面层

business-logic layer：业务逻辑层

data access layer：数据访问层

referential integrity：参照完整性

Server-Side Scripting：服务器端脚本

Client Side Scripting：客户端脚本

Discriminator：分辨符/partial key

Transaction：事务

Closure：闭包

sequential scan：顺序扫描

Ch1.

1. DBMS与文件系统的不同：

DBMS协调 physical and logical access to data,文件管理只有physical access to data;

DBMS reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it but 文件管理may not be readable by another program；

DBMS allow flexible access to data BUT 文件管理allow pre-determined access to data；

DBMS可以协调多个用户在同一时间对同一数据访问，文件管理允许多个program同时访问不同data。

1. Physical Data Independence – the ability to modify the physical schema without changing the logical schema
2. five responsibilities of a DBMS：interaction with the file manager.integrity enforcement. security enforcement.backup and recovery. concurrency control.
3. five main functions of a database administrator：

To backup data

• In some cases, to create the schema definition

• To define the storage structure and access methods

• To modify the schema and/or physical organization when necessary

• To grant authorization for data access

• To specify integrity constraints

Ch2

1.2条数据库使用null的原因：the actual data may be unknown or not exists.

1. procedural VS nonprocedural :

Nonprocedural simplify the specification of queries 简化查询规范

Procedural more powerful in terms of what computations they can perform 计算方面强大

Ch3

SQL练习

Ch4

1. grants authorizations to manager roles but not user because:如果authorization由user 提供，那么当作废这个user的authorization时，他之前做过grant authorization都会被作废，但如果是role就不会都作废
2. 看题

Ch5

5.12

JDBC ODBC

5.15

5.16

1. embedded SQL和SQL function：

SQL function是一种扩展的SQL来处理复杂数据（如图像）。或者执行复杂和非标准的机制

Embedded SQL当需要展示结果或与用户交互时，这个很有用

1. 一个trigger会引发其他的trigger,可能会导致多个嵌套触发，数据库会设置一个deep为什么？ 因为可能会导致无穷递归 non-terminating recursion.而且不能静态识别和禁止 statically identify and prohibit such triggers

Ch6习题

Ch7

7.16

1. 强实体与弱实体的区别：强实体有primary key 弱实体集没有主键，除非包含它所依赖的强实体集的属性
2. 需要弱实体的理由：

避免数据重复data duplication

弱实体能反应逻辑结构，一个弱实体依赖其他实体

强实体被删时，弱实体自动被删

弱实体可以与他们的强实体一起被物理存储

7.20 7.22 7.25

1. Explain the distinction between condition-defined and user-defined constraints：condition-defined constraints受到概化结构中上层实体attribute的约束，如student中有type “student\_type”，如果student\_type = graduate则是graduate\_student entity 否则是undergraduate\_student

user-defined constraints不受上述约束，由用户操作

其中Condition-defined constraints可以被数据库系统自动检测，即自动更新等操作

1. Explain the distinction between disjoint and overlapping constraints：

Disjoint :实体集只能属于低层实体集中的一类，如学生要么是未毕业的实体要么是毕业的实体

Overlapping：与上述相反

1. Explain the distinction between total and partial constraints.

Total 每个高层实体必须belong to一个低层实体

Partial 每个高层实体不需要必须belong to一个低层实体

Ch8

1. trivial functional dependencies: they are satisfied by all relations
2. 习题
3. List the three design goals for relational databases, and explain why each is desirable： lossless-join decompositions, dependency preserving decompositions, and minimization of repetition of information
4. why might we choose a non- BCNF design？BCNF is not always dependency preserving

why 4NF is a normal form more desirable than BCNF？ it reduces the repetition of information

Ch9

1. CRUD：CRUD是指在做计算处理时的增加(Create)、读取查询(Retrieve)、更新(Update)和删除(Delete)几个单词的首字母简写。主要被用在描述软件系统中数据库或者[持久层](http://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "http://baike.baidu.com/_blank)的基本操作功能
2. REST：REST 定义了一组体系架构原则，您可以根据这些原则设计以系统资源为中心的 Web 服务，包括使用不同语言编写的客户端如何通过 HTTP 处理和传输资源状态。
3. XSS attack:In an XSS attack, a malicious user enters code written in a client-side scripting language such as JavaScript or Flash instead of entering a valid name or comment. When a different user views the entered text,the browser would execute the script,which can carry out actions such as sending private cookie information back to the malicious user,or execute an action on a different Web site, such as a bank Web site,that the user may be logged into.
4. multi-factor authentication:使用多种独立信息来确定用户身份，比如除了密码还要用手机发送验证码
5. two advantages of encrypting data stored in the database：

a. Unauthorized users who gain access to the OS files in which the DBMS stores the data cannot read the data.

b. If the application encrypts the data before it reaches the database, it is possible to ensure privacy for the user’s data such that even privileged users like database administrators cannot access other users’ data.

1. challenge-response system：挑战/响应方式的身份认证机制就是每次认证时认证服务器端都给客户端发送一个不同的“挑战”码，客户端程序收到这个“挑战”码，根据客户端和服务器之间共享的密钥信息，以及服务器端发送的“挑战”码做出相应的“应答”。服务器根据应答的结果确定是否接受客户端的身份声明。从本质上讲，这种机制实际上也是一次性口令的一种。

比传统密码系统更安全因为在身份验证期间，密码不会在网络上传输

1. Store each relation in one file：Advantages of storing a relation as a file include using the file system provided by the OS , thus simplifying the DBMS , but incurs the disadvantage of restricting the ability of the DBMS to increase performance by using more sophisticated storage structures.
2. Store multiple relations (perhaps even the entire database) in one file：these complex structures can be implemented through the DBMS , but this increases the size and complexity of the DBMS
3. In the sequential file organization, why is an overflow block used even if there is, at the moment, only one overflow record? a block is the smallest space which can be read from a disk
4. Current generation disks have more sectors per track on outer tracks, and fewer sectors per track on inner tracks (since they are shorter in length). What is the effect of such a change on each of the three main indicators of disk speed? the disk’s data-

transfer rate will be greater on the outer tracks than the inner tracks。

Ch11

1. 什么时候dense index比sparse index好 the file is not sorted on the indexed field (such as when the index is a secondary index) or when the index file is small compared to the size of memory.
2. the difference between a clustering index and a secondary index:clustering index是严格有顺序的 secondary index无序存放。

Ch14

14.14 14.15

1. 为什么数据库支持并行执行Improved throughput and resource utilization，Reduced waiting time