6. Database Design*

6.1 Data Dependency and Normalization of Relational Schema

- Some dependent relations exist between attributes.
- Function dependency (FD): the most basic kind of data dependencies. The value of one or a group attributes can decide the value of other attributes.

  FD is the most important in general database design.
- Multi-valued Dependency (MVD): the value of some attribute can decide a group of values of some other attributes.
- Join Dependency (JD): the constraint of lossless join decomposition.

第10章 数据依赖和关系模式的规范化

10.1 关系模式设计中的一些数据语义问题

数据的语义不但表现为完整性约束，对数据库的状态或状态的转换施加一定的限制，而且对关系模式的设计也提出一定的要求。

属性间往往存在一定的依赖关系，而最基本的依赖关系是函数依赖。所谓函数依赖是指一个或一组属性的值可以决定其他属性的值。

一般地讲，设X，Y是关系的两个不同的属性组，如果Y函数依赖于X，或X函数决定Y，则其依赖关系可表示为X→Y。

设有一关系R，具有下列属性：学号（S#）、课程号（C#）、成绩（G）、任课教师姓名（TN）、教师所在系名（D）。这些数据具有下列语义：

（１）学号是一个学生的标识，课程号是一门课程的标识，这些标识与其代表的学生和课程分别一一对应；

（２）一位学生所修的每门课程都有一个成绩；

（３）每门课程（注意：不是每种课程！同一种课，如数学课，可以开好多门，每门课有一个课程号）只有一位任课教师，但一位教师可以教多门课；

（４）教师中没有重名，每位教师只属于一个系。

根据上述语义，可以确认下面函数依赖的集合：

F＝｛｛S＃，C＃｝→G，C＃→TN，TN→D｝

从图１０－１可以看出，属性集｛S＃，C＃｝可以决定其他所有属性的值，而｛S＃，C＃｝的任何子集则不能，故｛S＃，C＃｝是这个关系的主键。这样的关系用来查询是很方便的。

计算机系通知教师准备给学生补考，可以“查询图10-1

函数依赖计算机系所开课程的不及格学生的学号、不及格课程号以及任课教师的姓名”.查询仅涉及R一个关系，是一元查询，不须做连接运算，可以用下面的SQL语句来表达：

```
SELECT S#，C#，TN FROM R
   WHERE D＝CS'AND G＝F';
```

但是这样的关系也有问题，首先数据冗余太多，如一门课程的教师名须对选这门课的所有学生重复一次；一个系名须对选该系所开课程的所有学生重复一次。除冗余外，在进行增、删、改操作时，还会发生所谓更新异常现象：

（１）由于冗余，在修改时往往会导致数据的不一致。例如，改变一门课程的任课教师，或一门课改由另一个系开出，则需要修改多个元组。如果部分修改，部分不修改，则会导致数据的不一致。这叫修改异常。

（２）由于主属性不能为空值，如某系有位教师不教课，则这位教师的姓名及所属的系名就不能插入；同样，如果一位教师所开的课暂时无人选，或是列入计划而目前不开，则也无法插入。这叫插入异常。

（３）如果所有学生都退选一门课，则有关这门课的其他数据（任课教师名及所属系名）也将被删除。如果一位教师因病暂时停开他所开的课，则有关这位教师的其他信息（所属系、可开课程）都将被删去。这叫删除异常。

在上例的关系中，包含了三方面的信息：学生各门课程的成绩，各门课程开课的教师以及各个教师所属的系。

上例中,所有这三方面的数据都集中在一个关系中，此关系的主键为属性集｛S#，C#｝。

（C#，TN）和（TN，D）本来可以作为独立的关系而存在，而今却不得不依附于其他关系。这就是说，必须对应一个主键｛S#，C#｝的值，才能插入或存在一个（C#，TN）或（TN，D）的值。这是关系结构带来的限制，不是现实世界的真实反映。

解决这个问题的途径是把关系分解，也就是进行所谓关系规范化。例如，把上例的关系分解为下列三个关系：

SCG（S#，C#，G）

CTN（C#，TN）

TND（TN，D）

这样的分解使关系的语义单纯化，使之符合“一地一事”的原则。但是分解以后，对某些查询必须进行开销很大的连接操作，影响数据库的性能。

关系的规范化主要是对关系进行必要的分解，但如何分解，分解后是否有损于原来的信息，回答这些问题需要理论的指导，下面将讨论这些问题。

- R表示一个关系的模式，U={A1,A2,…,An}是R的所有属性的集合，F是R中函数依赖的集合，r是R所取的值，即R实有元组的集合。
- 定义10-1 设有一关系模式R(A1,A2,…,An),X和Y为其属性的子集。设t1,t2是关系R中的任意两个元组，如果t1[X]=t2[X],则t1[Y]=t2[Y]。这时我们称Y函数依赖于X，或X函数决定Y，X称为决定子（determinant）。

一个函数依赖要能成立，不但要求关系的当前值都能满足函数依赖条件，而且还要求关系的任一可能值都能满足函数依赖条件。确认一个函数依赖，需要弄清数据的语义，而语义是现实世界的反映，不是主观的臆断。

如果Y为X的子集，显然X→Y成立，这称为平凡函数依赖（trivial functional dependency）。平凡函数依赖必然成立，它不反映新的语义。我们平常所指的函数依赖一般都指非平凡函数依赖（nontrivial functional dependency）。

如果Y不函数依赖于X，则记做X $\not\rightarrow$ Y。

如果X→Y且Y→X，则X与Y一一对应，可记做

X$\longleftrightarrow$Y。

定义10-2 设X,Y是某关系的不同属性集,如X→Y，且不存在X′为X的子集，使**X′**→Y，则称Y完全函数依赖(full functional dependency)于X，记做X $\xrightarrow{f}$ Y；否则称Y部分函数依赖**p**(partial functional dependency)于X，记做X →Y。

在10.1节的例子中，{S#,C#}是主键，故{S#,C#}→TN，但C#→TN，故{S#，p f C#}→TN，而{S#，C#}→G。

定义10-3　设X,Y,Z是某关系的不同的属性集，如果X→Y,Y　X,Y→Z,则称Z传递函数依赖（transitive functional dependency）于X。

- 定义10-4 设F是R的函数依赖集合，X→Y是R的一个函数依赖。如果一个关系模式满足F，则必然满足X→Y，就称F逻辑蕴涵X→Y，或表示为F|=X→Y。
- 定义10-5 函数依赖集合 F 所逻辑蕴涵的函数依赖的全体称为F的闭包（closure），记为F^+，即F^+＝{X→Y | F|=X→Y}。

为了从已知的函数依赖推导出其他函数依赖，Armstrong提出了一套推理规则，我们常称为Armstrong公理（ Armstrong saxions）。其推理规则可归结为如下三条:

A1：自反律（reflexivity）

如果Y⊆X ⊆U，则X→Y成立。这是一个平凡函数依赖。

A2：扩展律（augumentation）

如果X→Y成立，且Z ⊆U，则XZ→YZ成立。式中，XZ和YZ是X∪Z和Y∪Z的简写，以后将沿用此表示法。

A3：传递律（transitivity）

如果X→Y，Y→Z成立，则X→Z成立。

引理10-1　Armstrong公理是正确的（sound），即如果F成立，则由F根据Armstrong公理所推导的函数依赖总是成立的。

引理10-2 下列三条推理规则是正确的。

(1)合并规则（the union rule）

{X→Y，X→Z} |=X→YZ。

(2)伪传递规则（the pseudo transitivity rule）：

{X→Y，WY→Z} |=XW→Z。

(3)分解规则（the decomposition rule）：如果X→Y且Z为Y的子集，则X→Z成立。

定义10-6　设X为U的子集，则属性集X关于函数依赖集F的闭包X^+定义为$X^+=\{A \mid A \in U$且$X \rightarrow A$可由Armstrong公理导出$\}$。

引理10-3　$X \rightarrow Y$能由Armstrong公理导出的充分必要条件是$Y \subseteq X^+$。

定理10-1　Armstrong公理是正确的、完备的(complete)

算法10-1　计算属性集X关于F的闭包X＋。

输入：属性集X为U的子集，函数依赖集F。

输出：X＋。

方法：按下列方法计算属性集序列$X^{(0)}, X^{(1)}, \cdots, X^{(i)} \cdots$

(1)初始化$X^{(0)}=X$，$i=0$。

(2)求属性集$B=\{A \mid (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W)\}$。

(3)$X^{(i+1)}=B \cup X^{(i)}$。

(4)判别$X^{(i+1)}=X^{(i)}$。

(5)若不等，$i=i+1$，返回(2)；若相等，$X^{+}=X^{(i)}$，结束。

【例10-1】　设F＝｛AB→C，D→EG，C→A，BE→C，BC→D，CG→BD，ACD→B，CE→AG｝，试用算法10-1计算$(BD)^+$。

解：令$X^{(0)}$＝BD。

找左部为BD的子集的函数依赖，只有D→EG。$X^{(1)}$＝BD∪EG＝BDEG,找左部为BDEG的子集的函数依赖，有D→EG，BE→C。$X^{(2)}$＝ BDEG ∪EGC＝ BCDEG

找左部为BCDEG的子集的函数依赖，有D→EG，C→A，BE→C，BC→D，CG→BD，CE→AG。$X^{(3)}$＝ BCDEG ∪ ABCDEG= ABCDEG

因$X^{(3)}=U$，一望可知不必再进行计算了，$(BD)^{+}=ABCDEG$。可是在算法10-1中，还要迭代一次才能结束。

定义10-7　设F，G为两个函数依赖集，如果$F^+=G^+$，则称F和G是等价的，也可称F覆盖（cover）G，或G覆盖F；也可说F和G互相覆盖。

引理10-4　F＝G的充分必要条件是$F \subseteq G^+$且$G \subseteq F^+$。

引理10-5 任一函数依赖集F总可以为一右部恒为单属性的函数依赖集所覆盖.

定义10-8　函数依赖集F如果满足下列条件，则称为极小函数依赖集或最小覆盖。

(1)F中每个函数依赖的右部为单属性。

(2)F中不存在这样的函数依赖X→A，使得F－｛X→A｝与F等价。

(3)在F中也不存在这样的X→A，使得F－｛X→A｝∪｛Z→A｝与F等价，式中，Z为X的子集。

定理10-2　任一函数依赖集F都与一最小函数依赖集F′等价。F′称为F的最小覆盖。

10.3 多值依赖

除了函数依赖外，关系的属性间还有其他一些依赖关系，多值依赖（MultiValued Dependency,MVD）是其中之一。

在多值依赖（表示为X→→Y，读做X多值决定Y，或Y多值依赖于X）中，对于给定的X值，其对应的是Y的一组数值（其个数可以从零到多个），而且这种对应关系，对于给定的X值所对应的（U－X－Y）的每个值都能成立。

- 定义10-9　设X，Y是关系模式R的属性集，如果对于R的任何值r，都有如下的性质，则称R满足X→→Y。

  与函数依赖一样，多值依赖也有一组公理：
- A 4：互补律

  如果X→→Y，则X→→（U－X－Y）。

  如果需要，可用X→→Y｜（U－X－Y）表示多值依赖，以强调其互补关系。
- A 5：扩展律（多值依赖）

  如果X→→Y，而V　W，则WX→→VY。

- A6：传递律（多值依赖）

  如果 $X \rightarrow\rightarrow Y$，且 $Y \rightarrow\rightarrow Z$，则 $X \rightarrow\rightarrow (Z - Y)$。

  下面两个公理与函数依赖和多值依赖都有关。
- A7：如果 $X \rightarrow Y$，$\subseteq$ 则 $X \rightarrow\rightarrow Y$，即函数依赖是多值依赖的特例。
- A8：如果 $X \rightarrow\rightarrow Y$，$Z \quad Y$，而且对某个与 Y 不相交的 W，有 $W \rightarrow Z$，则 $X \rightarrow Z$。

由前述公理，还可以推导出下列4个多值依赖推理规则。

（1）多值依赖合并规则

如果X→→Y，X→→Z，则X→→YZ。

（2）多值依赖伪传递规则

如果X→→Y，WY→→Z，则WX→→（Z－WY）。

（3）混合伪传递规则

如果X→→Y，XY→→Z，则X→（Z－Y）。

（4）多值依赖分解规则

如果X→→Y，X→→Z，则X→→（Y∩Z），X→→（Y－Z）及X→→（Z－Y）均成立。

10.4 连接依赖

前面所讲的函数依赖和多值依赖都是数据语义对数据所施加的某种限制。这些依赖关系可以统称为数据依赖（Data Dependency,DD）。数据依赖不仅包含函数依赖和多值依赖，还有其他一些依赖。

函数依赖实际表现为对属性值的约束，例如，王平是计算机系的教师，若有函数依赖TN→D（参看图10-1），则在TN为王平的元组中，其对应的D必为计算机系，不能为其他值。

多值依赖实际上表示为对元组值的约束，如在图10-3中，由于T→→S｜C，如果出现元组Li，No1 physics>及Li，No2 chemistry>，则必有元组Li，No2 physics>及Li，No1 chemistry>。

推广这个概念，还可以发现其他依赖关系，连接依赖（Join Dependency,JD）就是其中之一。

定义10-10　设X_1，X_2，…，X_n是关系R的属性集U的子集，且$\cup X_i = U$。若对R的任一值，$R = \infty R[X_i]$均成立，则称R具有连接依赖，记为$\infty(X_1, X_2, \ldots, X_n)$。

注：　∞表示连接操作

10.5 关系模式的分解及其问题

在10.1节中讨论过由函数依赖所引起的更新异常问题。同样，多值依赖和连接依赖也会引起类似的问题。解决这些问题的途径，就是按照前面曾提到过的“一地一事”原则，对关系模式进行分解，使其语义单纯化。

定义10-11 设有一关系模式R(U)，若用一关系模式的集合 $\{R_1(U_1), R_2(U_2), \ldots, R_k(U_k)\}$ 来取代，其中，$U=\cup U_i$，则称此关系模式的集合为R的一个分解，用 $\rho=\{R_1, R_2, \ldots, R_k\}$ 表示。

定义10-12　F在属性集U_i（为U的真子集）上的投影定义为$\prod_{Ui}(F)$＝｛X→Y｜X→Y∈F^+∧XY为U_i的子集｝

定义10-13　设ρ＝｛R_1，R_2,...,R_n｝是R的一个分解，r是R的任一个值，如果满足条件$r=\prod_{U1}(r)\infty\prod_{U2}(r)\cdots\prod_{Uk}(r)$则称$\rho$是无损连接分解或简称无损分解。

定义10-14　设ρ=｛R_1，R_2,...,R_n｝是R的一个分解，如$\cup\prod_{Ui}(F)|=F$，则称分解ρ保持函数依赖，或简称保持依赖。

关系模式经分解后与原来的关系等价。所谓“等价”不是“等同”，“等价”是指两者对数据的使用者来说应是等价的，即对分解前后的数据，做同样内容的查询，应产生同样的结果。这是对模式分解的最基本的要求，否则，不能进行分解。

如果硬要进行分解的话，那就意味着分解前后的模式代表着两个不同的现实世界。因此，无损分解应是关系模式分解时所必须满足的条件。不是任意的分解都是无损的。

关系分解还带来保持依赖的问题。不满足保持依赖条件，并不意味着某些函数依赖真正丢失了，而是某些函数依赖的有关属性分散在不同的关系中，不能被F的所有投影所蕴涵。不同关系的属性间的函数依赖关系并不是不可能存在的，问题在于这种函数依赖所代表的语义约束不如在一个关系中容易检查，一般须在检查前对有关的关系做一次连接运算。此外，不保持依赖还会引起一些更新异常，这可以通过下面的例子说明。

【例10-3】　取图10-1的一部分组成关系模式R（C#，TN，D），其函数依赖关系见图10-6，即F＝｛C#→TN，TN→D｝。C#→D为F所逻辑蕴涵，如图10-6中虚线所示。如果R分解为两个关系R1（C#，TN），R2（C#，D），则由于C#是R的主键，故R1和R2中都含有C#。如前所述，这样的分解是无损分解。但函数依赖TN→D不能被｛C#→TN，C#→D｝所逻辑蕴涵，故分解ρ＝｛R1，R2｝不保持依赖。TN和D分属于R1和R2。 R2中某门课的系名D应是R1中教该门课的教师所在系。如果R1中的教师所在的系变了，则需要修改R2中相应的属性D的值。更有甚者，如果一位教师不教课，则无法表示这位教师所在系的信息。

综上所述，关系模式的分解主要有两种准则：

（１）只满足无损分解要求；

（２）既满足无损分解要求，又满足保持依赖要求。

准则（２）要比准则（１）理想，但分解要受到更多的限制。当然，只满足保持依赖，而不满足无损分解要求的分解是存在的。

下面进一步讨论无损分解和保持依赖的性质及其判别方法。

首先介绍一些记号。设 $\rho=\{R_1, R_2, \ldots, R_n\}$ 是R的一个分解，r为R的一个值，$\Pi_{R_i}(r)$为r在$\mathbf{R_i}$的属性（即U_i）上的投影，t为r中的一个元组，$t[\mathbf{R_i}]=t[U_i]$是t在$\mathbf{R_i}$的属性上的投影。定义m_ρ为$m_\rho(r)=\infty\Pi_{R_i}(r)$

引理10-6　设 $\rho=\{R_1, R_2, \ldots, R_n\}$ 是R的一个分解，r为R的一个值，$\mathbf{r_i}=\Pi_{Ri}(r)$，则有：

（1）r是$\boldsymbol{m}_\rho(r)$的子集；

（2）如果$s=\boldsymbol{m}_\rho(r)$，则$\Pi_{Ri}(s)=\mathbf{r_i}$；

（3）$\boldsymbol{m}_\rho \boldsymbol{m}_\rho(r)=\boldsymbol{m}_\rho(r)$.

【例10-4】　设有两个关系模式R_1（AB）和R_2（BC），r_1，r_2为其值。若r_1＝｛a_1b_1｝,r_2＝｛b_1c_1，b_2c_2｝，则s＝ r_1∞ r_2＝｛$a_1b_1c_1$｝，$\Pi_{R2}(s)$=｛b_1c_1｝为r_2的子集。r_2中的元组b_2c_2由于在r_1中找不到匹配对象而不出现在s中。这些通过连接而被淘汰的元组称为不连接（dangling）元组。在图10-6所示的关系R(C#,TN,D)中，如果一位教师不教课，则其所属的系也不能在此关系中表示。如果分解为R_1(C#,TN)和R_3(TN,D)两个关系，就可以解决此问题。实际上，对应不教课教师的元组是以R_3中的不连接元组出现的。从中也可看出，为什么通过分解可以消除更新异常。

下面介绍无损分解的测试算法。

算法10-2　检验一个分解是否无损分解。

输入：关系模式R($A_1, A_2, \cdots, A_n$）；R上的函数依赖集F；R上的分解ρ＝｛$R_1, R_2, \cdots, R_k$｝。

输出：ρ是否无损分解。

方法：建立k×n的矩阵M（见图10-7），其中每列对应于R的一个属性，每行对应于ρ中的一个关系模式。

M矩阵各元素的值由下面的规则确定：

$$M[i,j]=\begin{cases} a_j & A_j \in U_i \\ b_{ij} & A_j \notin U_i \end{cases}$$

对F中的每一函数依赖X→Y反复进行下面的检查和处理，直至M无可改变为止。检查X中的属性所对应的列，找出X相等的那些行。如果找到X相等的两个行（或多行），就把对应行中Y的属性所对应的符号改成一致，即如果其中之一为a_j，则其他的也改成a_j；如果这两个符号为b_{ij}和b_{lj}，则将它们统一成b_{ij}和b_{lj}。

如此进行到M无可改变时，如果发现某一行变成了a_1，a_2, …, a_n，则ρ是无损分解，否则，ρ不是无损分解。

【例10-5】　设有关系模式R（ABCDE），ρ＝{R_1（AD），R_2（AB），R_3（BE），R_4（CDE），R_5（AE）}是R的一个分解。在R上有下列函数依赖：A→C，B→C，C→D，DE→C，CE→A。试判断ρ是否为无损分解。

解：先构造初始矩阵M，如图10-8(a)所示。然后按下列次序反复检查和修改M。

A→C

b_{13}，b_{23}，b_{53}→b_{13}

B→C

b_{13}，b_{33}→b_{13}

C→D

a_4，b_{24}，b_{34}，b_{54}→a_4

DE→C

a_3，b_{13}→a_3（该列其余行的b_{13}都须改成a_3，以保持它们的一致）

CE→A

b_{31}，b_{41}, a_1→a_1

再进行下去，M将无任何改动，检查和修改到此结束。最后的M示于图10-8(c)。从中可以看出，第三行都是a，故ρ是无损分解。

定理10-3　算法10-2可以正确判断一个分解是否无损分解。

定理10-4　设ρ＝{$R_1(U_1)$，$R_2(U_2)$}是R(U)的一个分解，则ρ为无损分解的充分必要条件为（$U_1 \cap U_2$）→（$U_1 - U_2$）或（$U_1 \cap U_2$）→（$U_2 - U_1$）。

【例10-6】　在图10-6所示的关系中，有

R（C#，TN，D）

F＝｛C#→TN，TN→D｝

R_1（C#，TN）

R_2（TN，D）试证ρ＝$\{R_1, R_2\}$是R的无损分解。

解：U_1＝｛C#，TN｝，U_2＝｛TN,D｝，$U_1 \cap U_2$＝｛TN｝，$U_1 - U_2$＝｛C#｝,$U_2 - U_1$＝｛D｝。

因TN→D，故（$U_1 \cap U_2$）→（$U_2 - U_1$）。

上述两条件之一成立，故ρ是无损分解。

下面介绍保持依赖的检验方法。

算法10-3　检验一个分解是否保持依赖。

输入：分解 $\rho = \{R_1, R_2, \cdots, R_k\}$ 和函数依赖集F。

输出：ρ 是否保持F。

方法：令$\mathbf{G}=\cup \prod_{Ui}(\mathbf{F})$。

为了检验G是否覆盖F，可对F中的每一函数依赖 $X \rightarrow Y$ 进行下列检查。

计算 X 关于 G 的闭包 X^+，并且检查 Y 是否包含在 X^+中。为了计算 X^+ ，不必求出G，可以分别地、反复地计算$\prod_{Ui}(\mathbf{F})$对X所增加的属性。

这可以用下面的算法：

Z ：＝X

WHILE Z有改变 DO

FOR i＝1 to k do

Z:=Z ∪（（Z ∩U_i）$^+$∩U_i）

Z ∩U_i是Z中与R_i有关的属性。（Z ∩U_i）$^+$是Z ∩U_i关于F的闭包。（Z ∩U_i）$^+$∩U_i是Π_{Ui}(F)对X^+所增加的属性。经反复计算，直至Z不变为止。最终的Z就是X关于G的闭包X^+。如果Y是X^+子集，则X→Y ∈G^+。

如果对F中的所有函数依赖经检查都属于G^+ ，则ρ保持依赖，否则ρ不保持依赖。

【例10-7】设有关系模式R(ABCD),F＝{A→B,B→C,C→D,D→A},试判别分解ρ={R_1(AB),R_2(BC),R_3(CD)}是否保持依赖。

解：$\Pi_{U1}(F)$＝{A→B}，$\Pi_{U2}(F)$＝{B→C}，$\Pi_{U3}(F)$＝{C→D}，F中的前三个函数依赖已明显地在G中，只要检验D→A是否为G所蕴涵。

令Z＝{D}作为Z的初始值。

第一趟：

i ＝1

$Z \cap U_1$＝{D}∩{A,B}＝空集，Z不变

i ＝2

$Z \cap U_2$＝{D}∩{B,C}＝空集，Z不变

i ＝3

$Z \cap U_3$＝{D}∩{C,D}＝{D}

$Z = \{D\} \cup (\{D\}^+ \cap \{C,D\}) = \{D\} \cup (\{A,B,C,D\} \cap \{C,D\}) = \{C,D\}$

第二趟：

$i = 1$

$Z \cap U_1 = \{C,D\} \cap \{A,B\} =$空集，$Z$不变

$i = 2$

$Z \cap U_2 = \{C,D\} \cap \{B,C\} = \{C\}$

$Z = \{C,D\} \cup (\{\mathbf{C}\}^+ \cap \{B,C\}) = \{C,D\} \cup (\{A,B,C,D\} \cap \{B,C\}) = \{B,C,D\}$

$i = 3$

$Z \cap U_3 = \{B,C,D\} \cap \{C,D\} = \{C,D\}$

$Z = \{B, C, D\} \cup (\{C,D\}^+ \cap \{C, D\}) =$
$\{B, C, D\} \cup (\{A, B, C, D\} \cap \{C, D\}) = \{B, C, D\}$， Z不变

第三趟：

i = 1

$Z \cap U_1 = \{B, C, D\} \cap \{A, B\} = \{B\}$

$Z = \{B, C, D\} \cup (\{B\}^+ \cap \{A, B\}) = \{B, C, D\} \cup (\{A, B, C, D\} \cap \{A, B\}))$
$= \{A, B, C, D\}$

Z已等于全部属性的集合，不可能再变，故$\{D\}^+ = \{A, B, C, D\}$。因为{A}是{A, B, C, D}的子集，故D→A属于G^+，即ρ可保持依赖。请注意，D，A两个属性虽然不同时出现在任何分解关系中，但G |= D→A，故仍保持依赖。

6.1 Data Dependency and Normalization of Relational Schema

- Some dependent relations exist between attributes.
- Function dependency (FD): the most basic kind of data dependencies. The value of one or a group attributes can decide the value of other attributes.

  FD is the most important in general database design.
- Multi-valued Dependency (MVD): the value of some attribute can decide a group of values of some other attributes.
- Join Dependency (JD): the constraint of lossless join decomposition.

1NF

every attribute of a relation must be atomic.

name	dept	address		
		prov	city	street

Non 1NF

name	dept	prov	city	street

1NF

2NF

- R∈1NF and no partially function dependency exists between attributes.

  S(S#, SNAME, AGE, ADDR, C#, GRADE)

  --- non 2NF

Problems of non 2NF:

- Insert abnormity: can not insert the students' information who have not selected course.
- Delete abnormity: if a student unselect all courses, his basic information is also lost.
- Hard to update: because of redundancy, it is hard to keep consistency when update.

Resolving：

According to the rule of **"one fact in one place"** to decompose the relation into 2 new relations:

S(S#, SNAME, AGE, ADDR)

SC(S#, C#, GRADE)

3NF

- R∈2NF and no transfer function dependency exists between attributes.

  EMP(EMP#, SAL_LEVEL, SALARY)

  --- non 3NF

Problems of non 3NF

- Insert abnormity: before the employees's sal_level are decided, the correspondence between sal_level and salary can not input.
- Delete abnormity: if some sal_level has only one man, the correspondence between sal_level and salary of this level will be lost when the man is deleted.
- Hard to update: because of redundancy, it is hard to keep consistency when update.

Resolving：

According to the rule of **"one fact in one place"** to decompose the relation into 2 new relations:

EMP(EMP#,SAL_LEVEL)

SAL(SAL_LEVEL,SALARY)

3NF

- R∈2NF and no transfer function dependency exists between attributes.

  EMP(EMP#, SAL_LEVEL, SALARY)

  --- non 3NF

Problems of non 3NF

- Insert abnormity: before the employees's sal_level are decided, the correspondence between sal_level and salary can not input.
- Delete abnormity: if some sal_level has only one man, the correspondence between sal_level and salary of this level will be lost when the man is deleted.
- Hard to update: because of redundancy, it is hard to keep consistency when update.

Resolving：

According to the rule of **"one fact in one place"** to decompose the relation into 2 new relations:

EMP(EMP#,SAL_LEVEL)

SAL(SAL_LEVEL,SALARY)

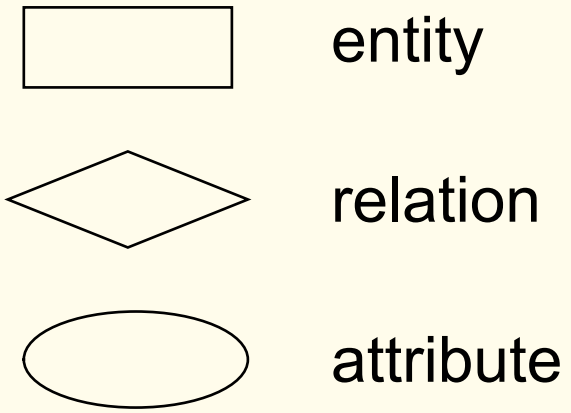
Material Card

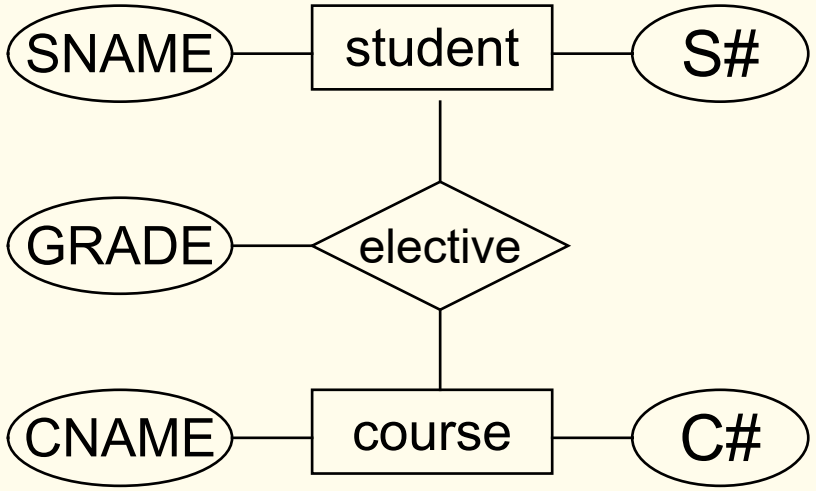
How to define relations to express the information on this card?

Equipment name： Type： Code：							
Unit price： Store place： room rack layer position							
date	voucher No.	coming/going place	take in	take out	balance	sum	remark

5.2 ER Model and ER Diagram

- Concept model: entity—relation, be independent of practical DBMS.
- Legend:





5.3 Database Design Method

- **Procedure oriented method**

  This method takes business procedures as center, the database schema is designed basically in accordance directly with the vouchers, receipts, reports, etc. in business. Because of no detailed analysis on data and inner relationships between data, although it is fast at the beginning of the project, it is hard to ensure software quality and the system will be hard to fit future changes in requirement and environment. So this method is not suitable for the development of a large, complex system.

- **Data oriented method**

  This method design the database schema based on the detailed analysis on data and inner relationships between data which are involved in business procedures. It takes data as center, not procedures. It can not only fulfill the current requirements, but also some potential requirements. It is liable to fit future changes in requirement and environment. It is recommended in the development of large, complex systems.

5.3 Database Design Method

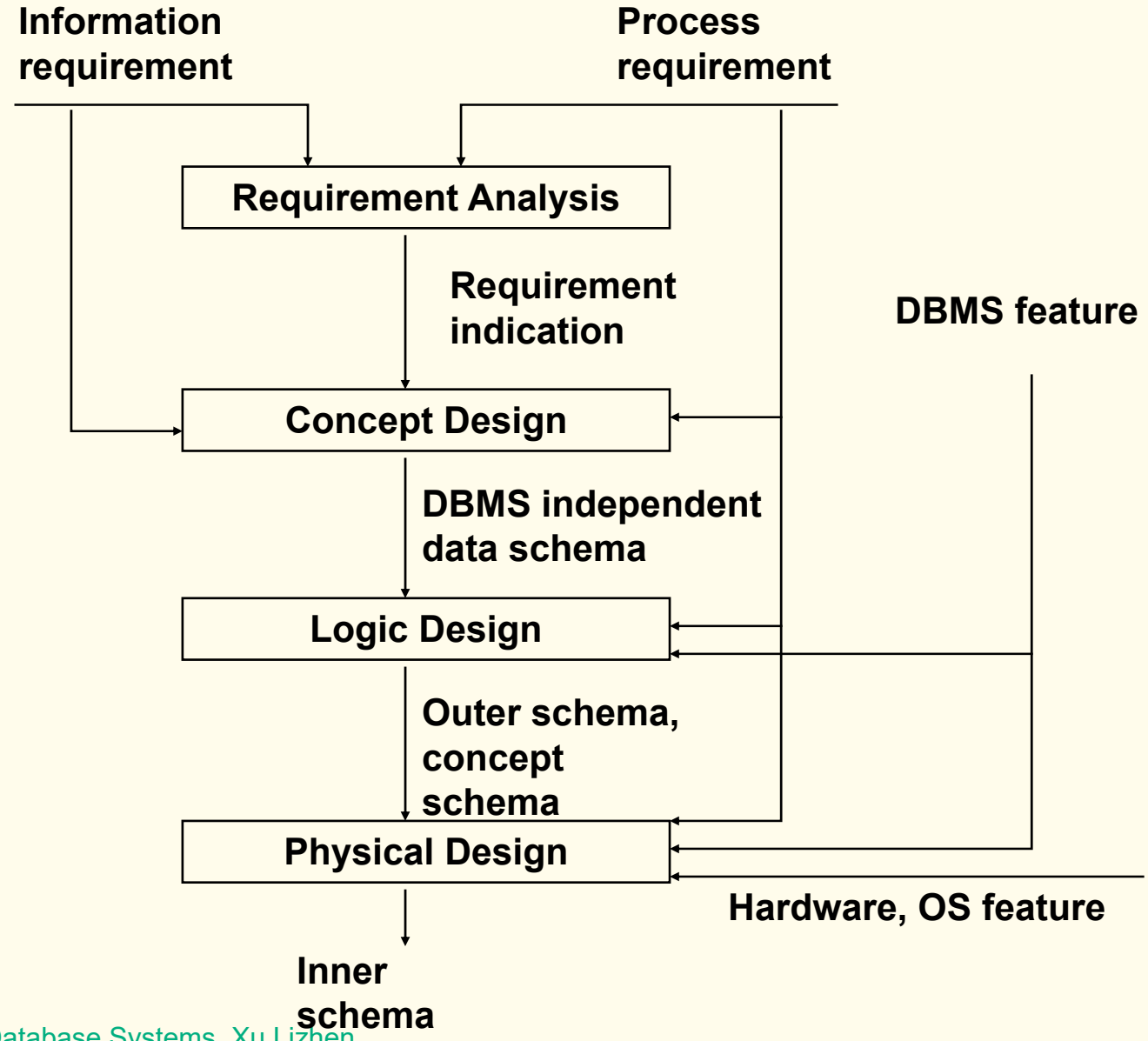
- **Procedure oriented method**

  This method takes business procedures as center, the database schema is designed basically in accordance directly with the vouchers, receipts, reports, etc. in business. Because of no detailed analysis on data and inner relationships between data, although it is fast at the beginning of the project, it is hard to ensure software quality and the system will be hard to fit future changes in requirement and environment. So this method is not suitable for the development of a large, complex system.

- **Data oriented method**

  This method design the database schema based on the detailed analysis on data and inner relationships between data which are involved in business procedures. It takes data as center, not procedures. It can not only fulfill the current requirements, but also some potential requirements. It is liable to fit future changes in requirement and environment. It is recommended in the development of large, complex systems.

Database Design Flow



Requirement Analysis

A very important part of system requirement analysis. In requirement analysis phase, the data dictionary and DFD (or UML) diagrams are the most important to database design.

- **Dictionary and DFD**
  - Name conflicts
    - Homonym(the same name with different meanings)
    - Synonym(the same meaning in different names)
  - Concept conflicts
  - Domain conflicts
- **About coding**
  - Standardization of information
  - Identifying entities
  - Compressing information
- **Through requirement analysis, all information must be with unique source and unique responsibility.**

Requirement Analysis

A very important part of system requirement analysis. In requirement analysis phase, the data dictionary and DFD (or UML) diagrams are the most important to database design.

- **Dictionary and DFD**
  - Name conflicts
    - Homonym(the same name with different meanings)
    - Synonym(the same meaning in different names)
  - Concept conflicts
  - Domain conflicts
- **About coding**
  - Standardization of information
  - Identifying entities
  - Compressing information
- **Through requirement analysis, all information must be with unique source and unique responsibility.**

Concept Design

Based on data dictionary and DFD, analyze and classify the data in data dictionary, and refer to the processing requirement reflected in DFD, identify entities, attributes, and relationships between entities. Then we can get concept schema of the database.

- Identify Entities
- Define the relationships between entities
- Draw ER diagram and discuss it with user
  - It is proposed to use ER design tools such as ERWin, Rose, etc.

Logic Design

According to the entities and relationships in ER diagram, define tables and views in target DBMS. Basic standard is 3NF.

- Translate entities and relationships in ER diagram to tables
- Naming rule of table and attribute
- Define the type and domain of every attribute
- Suitable denormalization
- Necessary view
- Consider the tables in legacy system
- Interface tables

Physical Design

For relational database, the main task in this phase is to consider creating necessary indexes according to the processing requirements, including single attribute indexes, multi attributes indexes, cluster indexes, etc. Generally, the attribute often as query conditions should have index.

Other problems:

- Partition design
- Stored procedure
- Trigger
- Integrity constraints

Remarks

- 仅仅在结构上达到3NF (BCNF) 是不够的。
- “一事一地”包括每项信息的唯一，要提取出问题的本质，识别出本质上同一概念的信息项。
- 对于表达类似信息，模式相似只是取值不同的表，应尽量合并。如学习经历、进修经历；奖励信息、惩处信息等。
- 考虑到效率、用途等因素，该分开的表还应分开。如本科生基本信息和研究生基本信息。
- 结合DBMS内部实现技术，合理设计索引和文件结构，为查询优化准备好存取路径。
- 在结构规范化、减少数据冗余和提高数据库访问性能之间仔细权衡，适当折中。

数据库设计实例分析

1.1.1学习经历

教师的学习简历

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10			
RXNY	入学年月	Text	20		是	因为入学年月中不包含日，故使用字符型 ，希望格式为 YYYY-MM
XXXS	学习形式	Text	60		是	因学习的形式很多，自己填写。例如：正规高等院校、 夜大学、函授等
XXFS	学习方式	Text	60		是	自己填写，内容如全脱产、半脱产、业务学习。
XZ	学制	Text	30		是	因考虑部分短期培训，如二个月，三个星期，因此使用字符型。
BYNY	毕业年月	Text	20		是	因为毕业年月中不包含日，故使用字符型 ，希望格式为 YYYY-MM
BYYXXHDW	毕、肄业学校或单位	Text	150		是	学习所在的学校、研究所或别的单位
SXZY	所学专业	Text	60		是	专业可能不在教委的专业信息中，所以由教职工自己填写。
XLDM	学历代码	Text	2		是	T_ZXBZ_WHCD 使用文化程度的字典表
XWDM	学位代码	Text	3		是	T_ZXBZ_XW
XWSYRQ	学位授予日期	Transitorily	-1		是	指获得此学位授予的时间。
XWSYGJDM	学位授予国家代码	Text	3		是	T_ZXBZ_GJDQ
XWSYDW	学位授予单位	Text	150		是	授予此次学习获得学位的单位
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.2攻读硕士博士

教职工攻读硕士博士履历信息

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		
SBRQ	申报日期	Transitorily	-1	是		申请日期
ZHXL	最后学历	Text	2		是	T_ZXBZ_WHCD，申请前的最高学历

字段 ID	字段中文名	类型	长度	主键	可否空	备注
SYSJ	最后学历授予时间	Transitorily	-1		是	申请前的最高学历授予时间
SYDW	最后学历授予单位	Text	150		是	申请前的最高学历授予单位
BKXW	报考学位	Text	3		是	T_ZXBZ_XW，报考的学位
BKXX	报考学校	Text	150		是	
BKZY	报考专业	Text	90		是	
BKFS	报考方式	Text	30		是	参照学习方式，内容如全脱产、半脱产、业务学习。
YWXXFWNF	愿为学校服务年份	Text	30		是	
HDXWRQ	获得学位日期	Transitorily	-1		是	获得硕士、博士的学位的时间
HDXW	获得学位	Text	3		是	T_ZXBZ_XW
HDXL	获得学历	Text	2		是	T_ZXBZ_WHCD
BXJE	报销金额	Numeric	-1		是	此信息供本人查询
BXRQ	报销日期	Transitorily	-1		是	报销日期
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.3国内进修学习

教职工在国内进修学习（非学历学位）

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		
XXFS	学习方式	Text	30		是	参考 DM-XXFS<学习方式代码>
XXQSNY	学习起始年月	Text	10	是		
XXZZNY	学习终止年月	Text	10		是	
XXNR	学习内容	Text	150		是	
JXBMC	进修班名称	Text	150		是	
ZXS	总学时	Numeric	-1		是	
ZBDW	主办单位	Text	150		是	
ZBDWXZ	主办单位性质	Text	10		是	
ZXDW	在学单位	Text	150		是	
JXJG	进修结果	Text	30		是	
CLRQ	处理日期	Transitorily	-1		是	

字段 ID	字段中文名	类型	长度	主键	可否空	备注
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.4T_JZG_XXJL(学习经历)

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		关联 T_JZG_JBXX 的 ZGH
RXNY	入学年月	Text	20	是		因为入学年月中不包含日，故使用字符型 ，希望格式为 YYYY-MM-DD
CJXXZL	参加学习种类	Text	60		是	如攻读硕博\国内进修学习\等.
XZ	学制	Text	30		是	因考虑部分短期培训，如二个月，三个星期，因此使用字符型。
BYNY	毕业年月	Text	20		是	因为毕业年月中不包含日，故使用字符型 ，希望格式为 YYYY-MM-DD
BYYXXHDW	毕、肄业学校或单位	Text	150		是	学习所在的学校、研究所或别的单位
SXZY	所学专业	Text	60		是	专业可能不在教委的专业信息中，所以由教职工自己填写。
XLDM	学历代码	Text	2		是	T_ZXBZ_WHCD 的 WHCDDM 使用文化程度的字典表
XWDM	学位代码	Text	3		是	T_ZXBZ_XW 的 XWDM
XWSYRQ	学位授予日期	Transitorily	-1		是	指获得此学位授予的时间。格式为 YYYY-MM-DD
XWSYGJDM	学位授予国家代码	Text	3		是	T_ZXBZ_GJDQ 的 GJDQDM
XWSYDW	学位授予单位	Text	150		是	授予此次学习获得学位的单位
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.5 T_JZG_SHJZ(社会兼职)

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		关联 T_JZG_JBXX 的 ZGH
QSRQ	起始日期	Transitorily	-1	是		兼职开始时间，具体时间可以 1 日进行录入
SHJZ	社会兼职	Text	90	是		教职工的社会兼职内容、如民主党派、工商等
JZZW	兼职职务	Text	60		是	如主委、秘书长、主席等
ZZRQ	终止日期	Transitorily	-1		是	兼职终止时间，具体时间可以 1 日进行录入
CZYY	辞职原因	Text	100		是	辞去兼职的原因
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.6 T_JZG_XSTTJZ(学术团体兼职)

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		关联 T_JZG_JBXX 的 ZGH
QSRQ	起始日期	Transitorily	-1	是		兼职开始时间，具体时间可以 1 日进行录入
XSTTMC	学术团体名称	Text	150	是		兼职的学术团体名称
TTJB	团体级别	Text	100		是	学术团体的级别
TTJZMC	团体兼职名称	Text	150		是	如理事、主席等
ZZRQ	终止日期	Transitorily	-1		是	兼职终止时间，具体时间可以 1 日进行录入
CZYY	辞职原因	Text	150		是	辞去兼职的原因
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.7教职工通讯信息

教职工本人的相关通讯信息

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		
JTZZ	家庭住址	Text	150		是	教职工的户口所在的地址,此类信息比较敏感,可以不填.
XZZ	现住址	Text	150		是	教职工真实居住的地址;此信息比较敏感,可以不填.
HKSZD	户口所在地	Text	150		是	户口所在的派出所
HKXZM	户口性质码	Text	1		是	对应 T_ZXBZ_HKXZ（此类信息目前已不再使用，但部分档案存在或需要；门户应用不要涉及）
TXDZ	通信地址	Text	150		是	教职工的通信地址;
YZBM	邮政编码	Text	6		是	教职工的通信地址对应的邮政编码
SJ	手机	Text	30		是	
ZZDH	住宅电话	Text	20		是	
BGLXDH	办公联系电话	Text	60		是	老师的联系电话.
DZXX	电子信箱	Text	60		是	
ZYDZ	主页地址	Text	120		是	教职工的个人主页地址
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	add_delete_update

1.1.8家庭基本信息

此类信息也是比较敏感,主要是便于联系其家庭人员.信息可以与教职工通讯信息不一样.

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZGH	职工号	Text	10	是		
JTZZ	家庭住址	Text	150		是	
JTYZBM	家庭邮政编码	Text	6		是	
JTDH	家庭电话	Text	30		是	
JTLXR	家庭联系人	Text	30		是	
JTDZXX	家庭电子信箱	Text	60		是	
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.8.1　课程基本信息

字段 ID	字段中文名	类型	长度	主键	可否空	备注
KCDM	课程代码	Text	20	是		
KCMC	课程名称	Text	90		是	
KCYWMC	课程英文名称	Text	100		是	
KCJJ	课程简介	Text	240		是	
KCYQ	课程要求	Text	240		是	对不同类学生也有不同
RKJS	任课教师	Text	60		是	此时其实无法确定
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.1.1 课程基本信息

字段 ID	字段中文名	类型	长度	主键	可否空	备注
KCDM	课程代码	Text	20	是		
KCMC	课程名称	Text	90		是	
KCYWMC	课程英文名称	Text	100		是	
KCJJ	课程简介	Text	240		是	
KCYQ	课程要求	Text	240		是	对不同类学生也有不同
RKJS	任课教师	Text	60		是	此时其实无法确定
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

1.1.1.2 专业所设课程情况

字段 ID	字段中文名	类型	长度	主键	可否空	备注
ZYDM	专业代码	Text	6	是		T_ZXBZ_BKSZY
KSXND	开设学年度	Text	10	是		
KKXQDM	开课学期代码	Text	10	是		
KCDM	课程代码	Text	20	是		
KCLBDM	课程类别代码	Text	10		是	参见 T_ZXBZ_KCLB
XF	学分	Numeric	-1		是	此时其实无法确定
XS	学时	Numeric	-1		是	此时其实无法确定
CLRQ	处理日期	Transitorily	-1		是	
SEQNUM	序列号	Numeric	-1		是	

1.1.1.3 课程安排（选课）

字段 ID	字段中文名	类型	长度	主键	可否空	备注
XKKCH	选课课程号	Text	60		是	
KCDM	课程代码	Text	20		是	T_JX_KCJBXX
KCM	课程名	Text	90		是	重复了
ZYFXMC	专业方向名称	Text	150		是	重复了

字段 ID	字段中文名	类型	长度	主键	可否空	备注
KCLB	课程类别	Text	120		是	重复了
XKXH	选课序号	Text	30		是	
KKXN	开课学年	Text	10		是	重复了
KKXQ	开课学期	Text	20		是	重复了
XF	学分	Numeric	-1		是	
SKZS	上课周数	Text	30		是	
SKAP	上课安排	Text	200		是	课程的上课信息,主要是周和节次
KS	课时	Text	30		是	和上表中的学时是否一回事？
KCXZ	课程性质	Text	60		是	和课程类别代码是否一回事？
JSDM	教室代码	Text	20		是	T_ZXBZ_JSXX
KKYX	开课院系	Text	10		是	T_ZXBZ_DW
JSGH	教师工号	Text	20		是	
JSXM	教师姓名	Text	60		是	考虑兼职教师,外聘教师
CLRQ	处理日期	Transitorily	-1		是	
XLH	序列号	Numeric	-1		是	
CZLX	操作类型	Text	10		是	

不论具体用户在现实中是怎么做的，术语是怎么叫的，关键是通过需求分析理清关于课程，最原子的概念是什么？如果同一门课，可以由不同院系、不同教师开设，而且同一教师在同一教室上的课，听课学生的要求可不一样，学分也不一样，则关于课程可分为三个表，表达三层概念，不仿称之为“课程”——“开课课程”——“选课课程”，教学要求和学分由选课课程才能确定。作为数据库设计者应将这些概念定义清楚，而不应受具体用户所用称谓的束缚和影响。

如果某高校管理较规范，不允许不同要求的学生混在一个教室上课，那么选课课程这一层概念就可取消，因为开课课程号就可唯一决定选课课程号；甚至如果某高校规定不同院系、不同教师开的课视作不同的课，有单独的课程号，那么开课课程这层概念也可取消，因为课程号就可唯一确定开课课程号和选课课程号。

因此，只有通过全面细致的分析，提取出问题的本质，才能设计出结构合理、稳定的数据库，适应不同高校不同情况的变化，以产品的形态满足市场需求。