第3章 关系数据库规划和设计

【本章导读】

数据库的开发和使用是一门综合性技术，做好规划和设计特别重要。本章主要讲述关系数据库的规范化理论、关系数据库的设计、关系数据库的标准语言 SQL、关系数据库的保护以及数据库的最新技术。

【本章要点】

- 关系数据库理论。
- 关系规范化的方法和步骤。
- 数据库设计的内容、任务、步骤和方法。
- SQL 的功能。
- 数据库的安全性、完整性、并发控制和恢复的原理及方法。
- 数据库技术的发展以及与其他技术的结合。

3.1 关系数据库理论

关系数据库是由一组关系组成的，那么针对一个具体问题，应该如何构建一个适合于它的数据模式，即应该构造几个关系、每个关系由哪些属性组成等，这是关系数据库的规范化问题。

3.1.1 函数依赖

1. 函数依赖

定义 3.1 设 $R(U)$ 是一个关系模式，U 是 R 的属性集合，X 和 Y 是 U 的子集。对于 $R(U)$ 的任意一个可能的关系 r，如果 r 中不存在两个元组，它们在 X 上的属性值相同，而在 Y 上的属性值不同，则称 X 函数确定 Y 或 Y 函数依赖于 X，记作 $X \rightarrow Y$。

对于函数依赖(Functional Dependency，FD)，需要说明以下 6 点。

(1) 函数依赖不是指关系模式 R 的某个或某些关系实例满足的约束条件，而是指 R 的所有关系实例均要满足的约束条件。

(2) 函数依赖是语义范畴的概念，只能根据数据的语义来确定函数依赖。例如，“姓名→所在系”这个函数依赖只有在没有同名人的条件下成立。如果存在名字相同的人，则“所在系”就不再函数依赖于“姓名”了。

(3) $X \to Y$，但 $Y \subsetneq X$，则称 $X \to Y$ 是非平凡函数依赖。$X \to Y$，但 $Y \subseteq X$，则称 $X \to Y$ 是平凡函数依赖。若不特别声明，总是讨论非平凡函数依赖。

(4) 若 $X \to Y$，则 X 称为这个函数依赖的决定属性集。

(5) 若 $X \to Y$，并且 $Y \to X$，则记为 $X \longleftrightarrow Y$。

(6) 若 Y 不函数依赖于 X，则记为 $X \not\to Y$。

2. 完全函数依赖与部分函数依赖

定义 3.2　在关系模式 $R(U)$ 中，如果 $X \to Y$，并且对于 X 的任何一个真子集 X'，都有 $X' \not\to Y$，则称 Y 完全函数依赖于 X，记作 $X \xrightarrow{f} Y$。若 $X \to Y$，但 Y 不完全函数依赖于 X，称 Y 部分函数依赖于 X，记作 $X \xrightarrow{p} Y$。

例如，在选课关系 SC(学号 $S\#$，课程号 $C\#$，成绩 G)中，学生的成绩由学号和课程号共同决定，代表该学生的一次选课，所以函数依赖为：$(S\#, C\#) \to G$，但是单独由学号不能决定一门课的成绩，单独由课程号也不能决定某个学生的成绩，即 $S\# \not\to G$、$C\# \not\to G$，所以 $(S\#, C\#) \xrightarrow{f} G$。

3. 传递函数依赖

定义 3.3　在关系模式 $R(U)$ 中，如果 $X \to Y$ ($Y \not\to X$)，$Y \subsetneq X$，$Y \to Z$，则称 Z 传递函数依赖于 X，记作 $X \xrightarrow{\text{传递}} Z$。

例如，在关系 SD(学号 $S\#$，所在系 SDEPT，系主任姓名 MNAME)中，学号决定学生所在系，即 $S\# \to$SDEPT，学生所在系决定系主任姓名 SDEPT$\to$MNAME，则 $S\# \xrightarrow{\text{传递}}$NAME。

4. 码

定义 3.4　设 K 为关系模式 $R<U, F>$ 中的属性或属性组合。若 $K \xrightarrow{f} U$，则称 K 为 R 的一个候选码(Candidate Key)。若关系模式有多个候选码，则选定其中的一个作为主码(Primary Key)。

例如，学生关系 STUDENT(学号 SNO，姓名 SNAME，性别 SSEX，年龄 SAGE，所在系 SDEPT)的主码是学号 SNO，因为学号 SNO 能决定姓名、性别、年龄、所在系这几个属性，即 SNO$\xrightarrow{f} U$。选课关系 SC(学号 $S\#$，课程号 $C\#$，成绩 G)的主码是 $(S\#, C\#)$，因为 $(S\#, C\#) \xrightarrow{f} G$，则 $(S\#, C\#) \xrightarrow{f} (S\#, C\#, G)$，即 $(S\#, C\#) \xrightarrow{f} U$。

包含在任何一个候选码中的属性称为主属性(Prime Attribute)，不包含在任何候选码中的属性称为非主属性(Nonprime Attribute)或非码属性(Non-key Attribute)。最简单的情况下，候选码只包含一个属性。在最极端的情况下，关系模式的所有属性组都是这个关系模式的候选码，称为全码(All-key)。

例如，选课关系 SC(学号 $S\#$，课程号 $C\#$，成绩 G)的主码是 $(S\#, C\#)$，主属性是主码的各个属性，即 $S\#$、$C\#$，非主属性是 G。

3.1.2　范式

范式是符合某一种级别的关系模式的集合。关系数据库中的关系必须满足一定的要求。满足的要求不同，则范式不同。范式的概念最早是由 E. F. Codd 提出的，他从 1971 年

相继提出了三级规范化形式,即满足最低要求的第一范式(1NF),在 1NF 基础上又满足某些特性的第二范式(2NF),在 2NF 基础上再满足一些要求的第三范式(3NF)。1974 年,E. F. Codd 和 Boyce 共同提出了一个新的范式概念,即 Boyce-Codd 范式,简称 BC 范式(BCNF)。1976 年 Fagin 提出了第四范式(4NF),后来又有人定义了第五范式(5NF)。至此,在关系数据库规范中建立了一个范式系列:1NF、2NF、3NF、BCNF、4NF 和 5NF。

通常把某一关系模式 R 为第 n 范式简记为 $R \in n\text{NF}$。

1. 1NF

定义 3.5 如果一个关系模式的所有属性都是不可分的基本数据项,则 $R \in 1\text{NF}$。

任何一个关系模式都是 1NF,不满足第一范式的数据库模式不能称为关系数据库。

满足第一范式的关系模式不一定是一个好的关系模式。如关系模式 SLC(学号 SNO,学生所在系 SDEPT,学生住处 SLOC,课程号 CNO,成绩 GRADE),假设每个系的学生住在同一个楼上。SLC 满足第一范式,每个属性都不可分。

SLC 的码是(SNO,CNO),函数依赖有:

$(\text{SNO},\text{CNO}) \xrightarrow{f} \text{GRADE}$,

$\text{SNO} \rightarrow \text{SDEPT}$,$(\text{SNO},\text{CNO}) \xrightarrow{p} \text{SDEPT}$,

$\text{SNO} \rightarrow \text{SLOC}$,$(\text{SNO},\text{CNO}) \xrightarrow{p} \text{SLOC}$,

$\text{SDEPT} \rightarrow \text{SLOC}$。

SLC 关系模式存在的问题如下。

(1) 插入异常。如果有一位新生的信息要插入 SLC 中,由于该生还没有选课,课程号 CNO 为空,所以无法插入。

(2) 删除异常。假定 SLC 中已有某个学生的选课记录,现在该学生要休学,这时需要将该生的选课记录都删除,由于课程号 CNO 是主属性,不能为空,所以删掉选课信息也就将整个元组都删除了,即把学生的信息也删掉了。从而删除了不应该删除的信息,造成了删除异常。

(3) 更新异常。假设某学生要转系,这样学生所在系 SDEPT、学生住处 SLOC 都要发生变化,如果该生选修 N 门课,对应关系中 N 个元组,学生信息发生变化导致 N 个元组都要发生变化,导致修改复杂,即更新异常。

(4) 冗余度大。假设某学生选了 10 门课,那么学生的 SDEPT 和 SLOC 就要重复存储 10 次,导致数据冗余。

由以上分析可以看出,第一范式存在许多问题,需要通过模式分解向高一级范式转换。

2. 2NF

定义 3.6 若关系模式 $R \in 1\text{NF}$,并且每一个非主属性都完全函数依赖于 R 的码,则 $R \in 2\text{NF}$。

2NF 不允许关系模式的属性之间有这样的函数依赖:$X \rightarrow Y$,其中,X 是码的真子集,Y 是非主属性。显然,码只包含一个属性的关系模式,如果属于 1NF,那么它一定属于 2NF。

关系模式 SLC 的码是(SNO,CNO),主属性是 SNO 和 CNO,非主属性是 SDEPT、SLOC、GRADE。分析 SLC 的函数依赖,可以看到:

$(\text{SNO},\text{CNO}) \xrightarrow{p} \text{SDEPT}$

$(SNO,CNO)\xrightarrow{p}SLOC$

存在非主属性对码的部分函数依赖，不满足第二范式的要求，所以 SLC 只能达到 1NF。

解决的办法是把关系模式 SLC 分解成为两个关系模式：SC(SNO,CNO,GRADE)和 SL(SNO,SDEPT,SLOC)。

SC 的码是(SNO,CNO)，函数依赖是$(SNO,CNO)\xrightarrow{f}GRADE$。

SL 的码是 SNO，函数依赖是 SNO→SDEPT、SNO→SLOC、SDEPT→SLOC。

由此可见，分解后的两个关系模式不存在非主属性对码的部分函数依赖，都达到了 2NF，即 SC∈2NF，SL∈2NF。

3. 3NF

定义 3.7　如果关系模式 $R<U,F>$ 中不存在候选码 X、属性组 Y 以及非主属性 $Z(Z\subsetneq Y)$，使得 $X\rightarrow Y(Y\nrightarrow X)$，$Y\rightarrow Z$ 成立，则 $R\in 3NF$。

由定义 3.7 可以证明，若 $R\in 3NF$，则 R 的每一个非主属性既不部分函数依赖于候选码，也不传递函数依赖于候选码。显然，如果 $R\in 3NF$，则 R 也是 2NF。

研究分解后的关系模式 SL(SNO,SDEPT,SLOC)，SL 的码是 SNO，主属性是 SNO，非主属性是 SDEPT 和 SLOC，存在非主属性 SLOC 对主属性 SNO 的传递函数依赖，所以达不到 3NF 的要求，可以将 SL 分解为：

SD(SNO,SDEPT)

DL(SDEPT,SLOC)

分解后的 SD 和 DL 不再存在传递函数依赖，所以 SD∈3NF，DL∈3NF。

4. BCNF

BCNF(Boyce Codd Normal Form，鲍依斯-科得范式)是由 Boyce 和 Codd 共同提出的，比上述 3NF 又进了一步，通常认为 BCNF 是修正的第三范式，有时也称为扩充的第三范式。

定义 3.8　设关系模式 $R<U,F>\in 1NF$。若 $X\rightarrow Y$ 且 $Y\subsetneq X$ 时 X 必含有码，则 $R<U,F>\in BCNF$。

也就是说，关系模式 $R<U,F>$ 中，若每一个决定因素都包含码，则 $R<U,F>\in BCNF$。

由 BCNF 的定义可以得到结论，一个满足 BCNF 的关系模式有：

(1) 所有非主属性对每一个码都是完全函数依赖。

(2) 所有主属性对每一个不包含它的码是完全函数依赖。

(3) 没有任何属性完全函数依赖于非码的任何一组属性。

2NF、3NF 分别消除了非主属性对码的部分函数依赖和传递函数依赖，而 BCNF 在 3NF 的基础上消除了主属性对码的部分函数依赖，因此如果 $R\in BCNF$，则 $R\in 3NF$，反之则不成立。

假设有系别管理关系 DEPT(系别 D，教研室 R，教师数 C，系主任 M)，其中，C 表示教研室中的教师数。一个系里只有一个主任，可以有多个教研室。这个数据库表中存在如下函数依赖。

$(D,R)\rightarrow(M,C)$，$(M,R)\rightarrow(D,C)$

所以，(D,R)和(M,R)都是 DEPT 的候选关键字，表中的唯一非关键字为 C，它是符合

3NF 的。但是,由于存在如下函数依赖:

$D \rightarrow M, M \rightarrow D$

即存在关键字段决定关键字段的情况,所以其不符合 BCNF 范式。它会出现如下异常情况:

(1) 插入异常。当没有提供教研室的信息时,无法插入此系及主任的信息。

(2) 更新异常。如果系更换了主任,则表中此系的所有行的主任列都要修改,造成修改操作的复杂化。

(3) 删除异常。如果要删除某系所有教研室的信息,则系别和主任的信息也将被删除,造成删除异常。

解决的办法是把 DEPT 关系表分解为下面两个关系表:系别管理 DEPT(D,M)和教研室管理 REAC(D,R,C)。这样的数据库表是符合 BCNF 范式的,消除了删除异常、插入异常和更新异常。

5. 多值依赖

研究下面的关系模式 Teaching(课程 C,教员 T,参考书 B),一门课程可以由多个教员讲授,使用相同的一套参考书。一个教员可以讲授多门课程,一本参考书可以供多门课程使用。如表 3-1 所示的示例数据表示了 C、T、B 之间的关系。

表 3-1 Teaching 表

课程 *C*	教员 *T*	参考书 *B*
物理	李勇	普通物理学
物理	李勇	光学物理
物理	王军	普通物理学
物理	王军	光学物理
数学	李勇	高等数学
数学	李勇	微分方程
数学	张平	高等数学
数学	张平	微分方程
…	…	…

关系模式 Teaching 具有唯一候选码(C,T,B),即全码。因而 Teaching$\in$BCNF。

但是当某一课程(如物理)增加一名讲课教员,如李兰时,由于存在多本参考书,所以必须插入多个元组:(物理,李兰,普通物理学)、(物理,李兰,光学原理)、(物理,李兰,物理习题集)。同样,要去掉一门课,也需要删除多个元组。

在这个例子中,存在着称为多值依赖的数据依赖。

定义 3.9 设 $R(U)$是属性集 U 上的一个关系模式。X、Y、Z 是 U 的子集,并且 $Z=U-X-Y$。关系模式 R 中多值依赖 $X \rightarrow\rightarrow Y$ 成立,当且仅当对于 R 的任一关系 r,给定的一对(x,z)值,有一组 y 的值,这组值仅取决于 x 值而与 z 值无关。

在上述关系模式 Teaching 中,T 多值依赖于 C,即 $C \rightarrow\rightarrow T$。即对于一个(物理,光学物理)有一组 T 值(李勇,王军),这组值仅决定于课程 C 上的值,即物理。这就是多值依赖。

若 $X \rightarrow\rightarrow Y$,而 $Z=\varnothing$,即 Z 为空,则称 $X \rightarrow\rightarrow Y$ 为平凡的多值依赖;反之即为非平凡的多值依赖。

多值依赖的主要性质如下。

(1) 多值依赖具有对称性。即若 $X \rightarrow\rightarrow Y$，则 $X \rightarrow\rightarrow Z$，其中 $Z=U-X-Y$。

(2) 多值依赖的传递性。即若 $X \rightarrow\rightarrow Y$，$Y \rightarrow\rightarrow Z$，则 $X \rightarrow\rightarrow Z-Y$。

(3) 函数依赖可以看作多值依赖的特殊情况。即若 $X \rightarrow Y$，则 $X \rightarrow\rightarrow Y$。这是因为当 $X \rightarrow Y$ 时，对 X 的每一个值 x，Y 有一个确定的值 y 与之对应，所以 $X \rightarrow\rightarrow Y$。

(4) 若 $X \rightarrow\rightarrow Y$，$X \rightarrow\rightarrow Z$，则 $X \rightarrow\rightarrow YZ$。

(5) 若 $X \rightarrow\rightarrow Y$，$X \rightarrow\rightarrow Z$，则 $X \rightarrow\rightarrow Y \cap Z$。

(6) 若 $X \rightarrow\rightarrow Y$，$X \rightarrow\rightarrow Z$，则 $X \rightarrow\rightarrow Y-Z$，$X \rightarrow\rightarrow Z-Y$。

6. 4NF

多值依赖对关系模式会产生怎样的影响呢？研究下面的关系模式 $R=\{$仓库(W)，保管员(S)，商品(C)$\}$，假定每个仓库有若干保管员、若干商品，每个保管员保管所在仓库的所有商品，每种商品被所在仓库的所有保管员保管。

可以看出，对于此关系模式上的实例 r，满足 $W \rightarrow\rightarrow S$ 及 $W \rightarrow\rightarrow C$，即某仓库 W_i 中有 n 个保管员及 m 种商品，则 W 属性值为 W_i 的元组在数据库表中必有 $m \times n$ 个，这样 r 中的数据冗余仍是十分可观的。为了进一步减少冗余，引入划分更为细致、限制条件更加严格的范式，即 4NF。

定义 3.10　关系模式 $R<U,F>\in$ 1NF，如果对于 R 的每个非平凡多值依赖 $X \rightarrow\rightarrow Y(Y \subsetneq X)$，$X$ 都含有码，则 $R \in$ 4NF。

4NF 就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的多值依赖。因为根据定义，对于每一个非平凡的多值依赖 $X \rightarrow\rightarrow Y$，$X$ 都含有候选码，于是就有 $X \rightarrow Y$，所以 4NF 所允许的非平凡的多值依赖实际上是函数依赖。

Teaching(C，T，B)不满足 4NF 的要求，因为存在非平凡的多值依赖 $C \rightarrow\rightarrow T$，且 C 不是候选码。把 Teaching 分解为 CT(C，T)、CB(C，B)两个关系模式，则它们均是 4NF。

由此可见，4NF 是在 BCNF 的基础上消除了非平凡且非函数依赖的多值依赖。如果一个关系模式是 4NF，则它必为 BCNF。

函数依赖和多值依赖是两种最重要的数据依赖。如果只考虑函数依赖，则属于 BCNF 的关系模式规范化程度已经最高了。如果考虑多值依赖，则属于 4NF 的关系模式规范化程度是最高的。而 5NF(投影、连接范式)是基于连接依赖的关系模式规范化范式，在本书中将不做介绍。

7. 关系模式的规范化

规范化的基本思想是逐步消除数据依赖中不合适的部分，使模式中的各关系模式达到某种程度的“分离”，即“一事一地”的模式设计原则。

通过对关系模式进行规范化，可以逐步消除数据依赖中不合适的部分，使关系模式达到更高的规范化程度。关系模式的规范化过程是通过对关系模式的分解来实现的，即把低一级的关系模式分解为若干个高一级的关系模式。关系模式的规范化过程可以用图 3-1 来概括。

关系模式的规范化过程是通过模式分解来实现的，而这种分解并不是唯一的。下面将进一步讨论分解后的关系模式与原关系模式的等价问题及分解的算法。

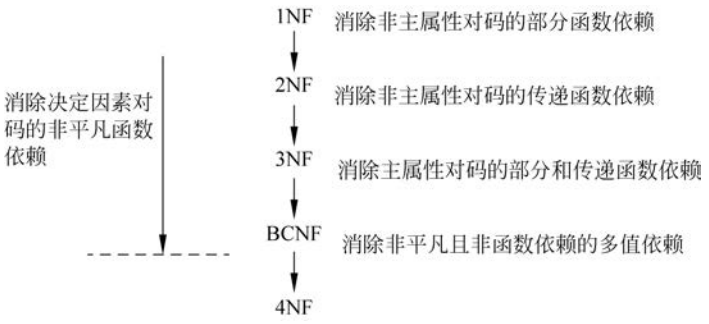

图 3-1 关系模式的规范化过程

3.1.3 数据依赖的公理系统

数据依赖的公理系统是模式分解算法的理论基础，下面首先讨论函数依赖的一个有效而完备的公理系统——Armstrong 公理系统。

1. Armstrong 公理系统

定义 3.11 对于满足一组函数依赖 F 的关系模式 R，其任何一个关系 r，若函数依赖 $X\rightarrow Y$ 都成立，则称 F 逻辑蕴含 $X\rightarrow Y$。

为了求得给定关系模式的码，首先要从一组函数依赖中求得其蕴含的函数依赖。而对于已知的函数依赖集 F，要判断 $X\rightarrow Y$ 是否为 F 所蕴含，就需要使用 Armstrong 公理系统。Armstrong 公理系统是 1974 年首先由 Armstrong 提出来的，这组公理系统是一套推理规则，是模式分解算法的理论基础。

Armstrong 公理系统对关系模式 $R<U,F>$ 来说有以下推理规则。

- A1. 自反律(平凡函数依赖)：若 $Y\subseteq X\subseteq U$，则 $X\rightarrow Y$ 为 F 所蕴含。
- A2. 增广律：若 $X\rightarrow Y$ 为 F 所蕴含，且 $Z\subseteq U$，则 $XZ\rightarrow YZ$ 为 F 所蕴含。
- A3. 传递律：若 $X\rightarrow Y$ 及 $Y\rightarrow Z$ 为 F 所蕴含，则 $X\rightarrow Z$ 为 F 所蕴含。

根据 A1、A2、A3 这 3 条推理规则可以得到下面 3 条很有用的导出规则。

- 合并规则：由 $X\rightarrow Y$，$X\rightarrow Z$，有 $X\rightarrow YZ$(由 A2、A3 导出)。
- 伪传递规则：由 $X\rightarrow Y$，$WY\rightarrow Z$，有 $XW\rightarrow Z$(由 A2、A3 导出)。
- 分解规则：由 $X\rightarrow Y$ 及 $Z\subseteq Y$，有 $X\rightarrow Z$(由 A1、A3 导出)。

根据合并规则和分解规则，得到下面的引理。

引理 3.1 $X\rightarrow A_1A_2\cdots A_k$ 成立的充分必要条件是 $X\rightarrow A_i$ 成立($i=1,2,\cdots,k$)。

定义 3.12 设 F 为属性集 U 上的一组函数依赖，$X\subseteq U$，$X_F^+=\{A\mid X\rightarrow A$ 能由 F 根据 Armstrong 公理导出$\}$，X_F^+ 称为属性集 X 关于函数依赖集 F 的闭包。

引理 3.2 设 F 为属性集 U 上的一组函数依赖，X、$Y\subseteq U$，$X\rightarrow Y$ 能由 F 根据 Armstrong 公理导出的充分必要条件是 $Y\subseteq X_F^+$。

于是，判定 $X\rightarrow Y$ 是否能由 F 根据 Armstrong 公理导出的问题，就转换为求出 X_F^+，判定 Y 是否为 X_F^+ 的子集的问题。这个问题可以使用算法 3.1 解决。

2. 求解函数依赖集的闭包

算法 3.1 求属性集 $X(X\subseteq U)$关于 U 上的函数依赖集 F 的闭包 X_F^+。

输入：X，F。

输出：X_F^+。

步骤：

(1) 令 $X^{(0)}=X$，$i=0$。

(2) 求 B，这里 $B=\{A\mid(\exists V)(\exists W)(V\rightarrow W\in F\wedge V\subseteq X^{(i)}\wedge A\in W)\}$。

(3) $X^{(i+1)}=B\cup X^{(i)}$；

(4) 判断 $X^{(i+1)}=X^{(i)}$；

(5) 若相等或 $X^{(i+1)}=U$，则 $X^{(i+1)}$ 就是 X_F^+，算法终止。

(6) 若否，则 $i=i+1$，返回第(2)步。

【例 3-1】 已知关系模式 $R<U,F>$，其中，$U=\{A,B,C,D,E\}$，$F=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,EC\rightarrow B,AC\rightarrow B\}$，求 $(AB)_F^+$。

解：设 $X^{(0)}=AB$

(1) 计算 $X^{(1)}$：逐一扫描 F 集合中各个函数依赖，找左部为 A、B 或 AB 的函数依赖，得到两个：$AB\rightarrow C$，$B\rightarrow D$。于是，$X^{(1)}=AB\cup CD=ABCD$。

(2) 因为 $X^{(0)}\neq X^{(1)}$，所以再找出左部为 $ABCD$ 的子集的那些函数依赖，又得到 $AB\rightarrow C$，$B\rightarrow D$，$C\rightarrow E$，$AC\rightarrow B$。于是，$X^{(2)}=X^{(1)}\cup BCDE=ABCDE$。

(3) 因为 $X^{(2)}=U$，算法终止。

得到结果：$(AB)_F^+=ABCDE$。

根据以上的定理及算法，可以根据关系模式的函数依赖集判断关系模式的码，从而判断关系模式的规范化程度，即判断关系模式达到第几范式。

【例 3-2】 关系模式 $R<U,F>$，$U=\{A,B,C,D,E\}$，函数依赖集 $F=\{AB\rightarrow CE,E\rightarrow AB,C\rightarrow D\}$，$R$ 最高属于第几范式？

(1) 求函数依赖集中决定因素是否为候选码，即求 AB_F^+、E_F^+、C_F^+。得到：$AB_F^+=U$、$E_F^+=U$，求 A_F^+ 和 B_F^+，判断是否为 U，$A_F^+\neq U$，$B_F^+\neq U$，所以 AB 和 E 是候选码。

(2) 由候选码判断主属性为 A、B、E，非主属性为 C 和 D。

(3) 判断非主属性对候选码有没有部分函数依赖。

候选码 E 只有一个属性，不可分，所以不必判断。

判断候选码 AB，决定因素中没有 A 或 B，所以不存在非主属性对候选码的部分函数依赖，达到 2NF。

(4) 判断非主属性对候选码有没有传递函数依赖。

在函数依赖集中有 $AB\rightarrow CE$、$C\rightarrow D$，所以 $AB\rightarrow C$、$C\rightarrow D$，存在非主属性 D 对候选码 AB 的传递函数依赖，只能达到 2NF。

得到结论：关系模式 R 只能达到第二范式。

3. 最小依赖集

从蕴含的概念出发，又引出了最小依赖集的概念。每一个函数依赖集 F 均等价于一个极小函数依赖集 F_m，此 F_m 称为 F 的最小依赖集。求得最小函数依赖集是模式分解的基础。下面就给出求 F 的最小依赖集的算法。

算法 3.2　分 3 步对 F 进行“极小化处理”，找出 F 的一个最小依赖集。

(1) 逐一检查 F 中各函数依赖 FD_i：$X\rightarrow Y$。

若 $Y=A_1A_2\cdots A_k$，$k>2$，则用$\{X\rightarrow A_j \mid j=1,2,\cdots,k\}$来取代 $X\rightarrow Y$。

(2) 逐一检查 F 中各函数依赖 FD_i：$X\rightarrow A$，令 $G=F-\{X\rightarrow A\}$，若 $A\in X_G^+$，则从 F 中去掉此函数依赖。

(3) 逐一取出 F 中各函数依赖 FD_i：$X\rightarrow A$，设 $X=B_1B_2\cdots B_m$，逐一考察 $B_i(i=1,2,\cdots,m)$，若 $A\in(X-B_i)_F^+$，则以 $X-B_i$ 取代 X。

最后剩下的 F 就一定是极小依赖集。

【例 3-3】 $F=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,EC\rightarrow B,AC\rightarrow B\}$，求其极小函数依赖集。

解：

第一步：先分解右端，F 不变。

$F=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,EC\rightarrow B,AC\rightarrow B\}$

第二步：

(1) 去掉 $AB\rightarrow C$，得到 $G=\{B\rightarrow D,C\rightarrow E,EC\rightarrow B,AC\rightarrow B\}$，

求 AB_G^+，AB_G^+ 中不包含 C，不可以去掉 $AB\rightarrow C$。F 不变。

(2) 去掉 $B\rightarrow D$，得到 $G=\{AB\rightarrow C,C\rightarrow E,EC\rightarrow B,AC\rightarrow B\}$，

求 B_G^+，B_G^+ 中不包含 D，不可以去掉 $B\rightarrow D$。F 不变。

(3) 去掉 $C\rightarrow E$，得到 $G=\{AB\rightarrow C,B\rightarrow D,EC\rightarrow B,AC\rightarrow B\}$，

求 C_G^+，C_G^+ 中不包含 E，不可以去掉 $C\rightarrow E$。F 不变。

(4) 去掉 $EC\rightarrow B$，得到 $G=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,AC\rightarrow B\}$，

求 EC_G^+，EC_G^+ 中不包含 B，不可以去掉 $EC\rightarrow B$。F 不变。

(5) 去掉 $AC\rightarrow B$，得到 $G=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,EC\rightarrow B\}$，

求 AC_G^+，AC_G^+ 中包含 B，可以去掉 $AC\rightarrow B$。

$F=G=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,EC\rightarrow B\}$。

第三步：

(1) 判断 $AB\rightarrow C$：

求 $A_F^+=A$，不包含 C，不能去掉 B。

求 $B_F^+=BD$，不包含 C，不能去掉 A。

(2) 判断 $EC\rightarrow B$：

求 $E_F^+=E$，不包含 B，不能去掉 C。

求 $C_F^+=BCDE$，包含 B，可以去掉 E。

F 变为 $F=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E,C\rightarrow B\}$

最终得到的就是 F 的最小函数依赖集。

注意：F 的最小函数依赖集不一定是唯一的，它与对各函数依赖 FD_i 及 $X\rightarrow A$ 中 X 各属性的处理顺序有关。

3.1.4 关系模式的规范化

一个关系只要其分量都是不可分的数据项，它就是规范化的关系。但规范化程度过低的关系不一定能够很好地描述现实世界，可能会存在插入异常、删除异常、修改复杂、数据冗余等问题。解决这些问题的方法就是对其进行规范化，转换成高级范式。一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式集合，这个过程就叫关

系模式的规范化。

1. 规范化的原则

关系模式的规范化是通过对关系模式的分解来实现的，但是把低一级的关系模式分解为若干个高一级的关系模式的方法并不是唯一的。在这些分解方法中，只有能够保证分解后的关系模式与原关系模式等价的方法才有意义。

下面先来看一下判断关系模式的一个分解是否与原关系模式等价的两种不同的标准。

(1) 分解具有无损连接性。

设关系模式 $R<U,F>$ 被分解为若干个关系模式 $R_1(U_1,F_1)$，$R_2(U_2,F_2)$，…，$R_n(U_n,F_n)$（其中 $U=U_1 \cup U_2 \cup \cdots \cup U_n$，且不存在 $U_i \subseteq U_j$，R_i 为 R 在 U_i 上的投影），若 R 与 R_1，R_2，…，R_n 自然连接的结果相等，则称关系模式 R 的这个分解具有无损连接性。

(2) 分解保持函数依赖。

设关系模式 $R<U,F>$ 被分解为若干个关系模式 $R_1(U_1,F_1)$，$R_2(U_2,F_2)$，…，$R_n(U_n,F_n)$（其中 $U=U_1 \cup U_2 \cup \cdots \cup U_n$，且不存在 $U_i \subseteq U_j$，F_i 为 F 在 U_i 上的投影），若 F 所逻辑蕴含的函数依赖一定也由分解得到的某个关系模式中的函数依赖 F_i 所逻辑蕴含，则称关系模式 R 的这个分解是保持函数依赖的。

如果一个分解具有无损连接性，则它能够保证不丢失信息。如果一个分解保持了函数依赖，则它可以减轻或解决各种异常情况。

表 3-2 给出了关系模式 SL(S#,SDEPT,SLOC)的一个关系。下面就对此关系做不同的分解。

表 3-2　SL

S#(学号)	SDEPT(所在系)	SLOC(住处)
95001	计算机系	1 号楼
95002	信息系	2 号楼
95003	数学系	3 号楼
95004	信息系	2 号楼
95005	物理系	2 号楼

① 分解为下面 3 个关系模式。

SN(S#),SD(SDEPT),SO(SLOC)

分解后的关系如表 3-3 所示。

表 3-3　分解模式 1

SN	SD	SO
S#(学号)	SDEPT(所在系)	SLOC(住处)
95001	计算机系	1 号楼
95002	信息系	2 号楼
95003	数学系	3 号楼
95004	物理系	
95005		

分解后的关系规范化程度很高,但 SN、SD、SO 无法连接,丢失了很多信息。

② 分解为下面 2 个关系模式。

NL(S#,SLOC),DL(SDEPT,SLOC)

分解后的关系如表 3-4 所示。

表 3-4 分解模式 2

NL

S#(学号)	SLOC(住处)
95001	1 号楼
95002	2 号楼
95003	3 号楼
95004	2 号楼
95005	2 号楼

DL

SDEPT(所在系)	SLOC(住处)
计算机系	1 号楼
信息系	2 号楼
数学系	3 号楼
物理系	2 号楼

将 NL 和 DL 进行自然连接,结果集如表 3-5 所示。结果比原来的 SL 关系多了 3 个元组。

表 3-5 NL 和 DL 的自然连接结果集

S#(学号)	SDEPT(所在系)	SLOC(住处)
95001	计算机系	1 号楼
95002	信息系	2 号楼
95002	物理系	2 号楼
95003	数学系	3 号楼
95004	信息系	2 号楼
95004	物理系	2 号楼
95005	信息系	2 号楼
95005	物理系	2 号楼

③ 分解为下面 2 个关系模式。

ND(S#,SDEPT),NL(S#,SLOC)

分解后的关系如表 3-6 所示。将 ND 和 NL 进行自然连接,结果与 SL 完全一样,无信息丢失或添加。但它没有保持原关系中的函数依赖:SDEPT→SLOC。

表 3-6 分解模式 3

NL

S#(学号)	SOEPT(所在系)
95001	计算机系
95002	信息系
95003	数学系
95004	信息系
95005	物理系

DL

S#(学号)	SLOC(住处)
95001	1 号楼
95002	2 号楼
95003	3 号楼
95004	2 号楼
95005	2 号楼

④ 分解为下面 2 个关系模式。

ND(S#,SDEPT),DL(SDEPT,SLOC)

分解后的关系如表 3-7 所示。

表 3-7　分解模式 4

NL

S#（学号）	SLOC（住处）
95001	计算机系
95002	信息系
95003	数学系
95004	信息系
95005	物理系

DL

SDEPT（所在系）	SLOC（住处）
计算机系	1 号楼
信息系	2 号楼
数学系	3 号楼
物理系	2 号楼

这种分解方式既没有信息丢失，又保持了原关系中的函数依赖，是理想的分解。

分解具有无损连接性和分解保持函数依赖是两个互相独立的标准。具有无损连接性的分解不一定能够保持函数依赖。同样，保持函数依赖的分解也不一定具有无损连接性。例如，上面的第一种分解方法既不具有无损连接性，也未保持函数依赖。第二种分解方法保持了函数依赖，但不具有无损连接性。第三种分解方法具有无损连接性，但未保持函数依赖。它们都不是原关系模式的一个等价分解。第四种分解方法既具有无损连接性，又保持了函数依赖，它是原关系模式的一个等价分解。

2. 规范化的方法

算法 3.3　达到 3NF 保持函数依赖的分解方法。

设关系模式 $R<U,F>$，达到 3NF 保持函数依赖的分解步骤如下。

（1）求 F 的最小函数依赖集，令 $F=F_{\min}$。

（2）如果 U 中某些属性不出现在 F 中，将这些属性组成一个关系模式，从 R 中分离出去。

（3）对 F 中每一个 $X_i \rightarrow A_i$，都构成一个关系模式 $R_i=X_iA_i$。如果 F 中有 $X \rightarrow A_1,\cdots,X \rightarrow A_n$（左部决定因素相同），则以 $X,A_1,A_2,\cdots,A_n$ 构成一个关系模式输出。

【例 3-4】　关系模式 R 的最小函数依赖集 $F=\{B \rightarrow G, CE \rightarrow B, C \rightarrow A, CE \rightarrow G, B \rightarrow D, C \rightarrow D\}$，将该关系模式分解为 3NF 且保持函数依赖。

解：

对 $B \rightarrow G, B \rightarrow D$，得到 $R_1: U_1(BDG), F_1=\{B \rightarrow G, B \rightarrow D\}$

对 $CE \rightarrow B, CE \rightarrow G$，得到 $R_2: U_2(BCEG), F_2=\{CE \rightarrow B, CE \rightarrow G\}$

对 $C \rightarrow A, C \rightarrow D$，得到 $R_3: U_3(ACD), F_3=\{C \rightarrow A, C \rightarrow D\}$

分解完毕。

算法 3.4　达到 3NF 保持函数依赖和无损连接性的分解。

分解步骤如下。

（1）按照达到 3NF 保持函数依赖的分解将 R 分解为 $R_1,R_2,\cdots,R_n$。

（2）选取 R 的主码，将主码与函数依赖相关的属性组成一个关系 R_{n+1}。

（3）如果 R_{n+1} 就是 $R_1,R_2,\cdots,R_n$ 中的一个，将它们合并，否则加入分解后的关系模式。

【例 3-5】　关系模式 R 的最小函数依赖集 $F=\{B \rightarrow G, CE \rightarrow B, C \rightarrow A, CE \rightarrow G, B \rightarrow D,$

$C\rightarrow D\}$,求达到 3NF 保持函数依赖和无损连接性的分解。

解:

$R_1: U_1(BDG), F_1=\{B\rightarrow G, B\rightarrow D\}$

$R_2: U_2(BCEG), F_2=\{CE\rightarrow B, CE\rightarrow G\}$

$R_3: U_3(ACD), F_3=\{C\rightarrow A, C\rightarrow D\}$

R 的码是 CE,与 CE 相关的函数依赖是 $CE\rightarrow B, CE\rightarrow G$。

形成 $R_4: U_4(BCEG)$,因为 R_4 和 R_2 相等,所以合并 R_4 和 R_2。R 分解为 R_1,R_2 和 R_3。分解完毕。

算法 3.5 达到 BCNF 保持无损连接性的分解。

设关系模式 $R<U,F>$,令 $\rho=\{R\}$,如果 ρ 中所有关系模式都达到 BCNF,则结束;否则在 r 中选择不是 BCNF 的关系模式 S,在 S 中必存在 $X\rightarrow A$,X 不包含 S 的码,也不包含 A。此时用 S_1 和 S_2 代替 S,$S_1(XA)$,S_2(S 中的属性集 A)。

【例 3-6】 关系模式 $R<U,F>$,$U=\{A,B,C,D,E\}$,最小函数依赖集 $F=\{AB\rightarrow C, B\rightarrow D, D\rightarrow E\}$,码是 AB。将 R 分解为 BCNF 且保持无损连接。

解:

(1) $D\rightarrow E$ 的决定因素不是 R 的码,将 R 分解为$\{R_1(DE), R_2(ABCD)\}$,$F_1=\{D\rightarrow E\}$,$F_2=\{AB\rightarrow C, B\rightarrow D\}$。

(2) 考虑 R_2 中,$B\rightarrow D$ 的决定因素不是 R_2 的码,将 R_2 分解为$\{R_2(BD), R_3(ABC)\}$。

(3) 最终 R 分解为$\{R_1(DE), R_2(BD), R_3(ABC)\}$。

3.2 数据库设计

数据库技术是信息资源开发、管理和服务最有效的手段,因此数据库的应用范围越来越广,从小型的事务处理系统到大型的信息系统大多利用了先进的数据库技术来保持系统数据的整体性、完整性和共享性。目前,数据库的建设规模、信息量大小和使用频度已成为衡量一个国家信息化程度的重要标志之一。这就使如何科学地设计与实现数据库及其应用系统成为日益引人注目的课题。

大型数据库设计是一项庞大的工程,其开发周期长、耗资多。它要求数据库设计人员既要具有坚实的数据库知识,又要充分了解实际应用对象。所以可以说数据库设计是一项涉及多学科的综合性技术。设计出一个性能较好的数据库系统并不是一件简单的工作。

3.2.1 数据库设计的任务与内容

数据库设计的任务是在 DBMS 的支持下,按照应用的要求,为某一部门或组织设计一个结构合理、使用方便、效率较高的数据库及其应用系统。

数据库设计应包含两方面的内容:一是结构设计,也就是设计数据库框架或数据库结构;二是行为设计,即设计应用程序、事务处理等。

设计数据库应用系统,首先应进行结构设计。一方面,数据库结构设计得是否合理,直接影响系统中各个处理过程的性能和质量。另一方面,结构特性又不能与行为特性分离。静态的结构特性的设计与动态的行为特性的设计分离,会导致数据与程序不易结合,增加数

据库设计的复杂性。

3.2.2　数据库设计方法

目前常用的各种数据库设计方法大部分属于规范设计法，即都是运用软件工程的思想与方法，根据数据库设计的特点，提出了各种设计准则与设计规范。这种工程化的规范设计方法也是在目前技术条件下设计数据库的最实用方法。

在规范设计中，数据库设计的核心与关键是逻辑数据库设计和物理数据库设计。逻辑数据库设计是根据用户要求和特定数据库管理系统的具体特点，以数据库设计理论为依据，设计数据库的全局逻辑结构和每个用户的局部逻辑结构。物理数据库设计是在逻辑结构确定之后，设计数据库的存储结构及其他实现细节。

规范设计在具体使用中又可以分为两类：手工设计和计算机辅助数据库设计。按规范设计法的工程原则与步骤手工设计数据库，其工作量较大，设计者的经验与知识在很大程度上决定了数据库设计的质量。计算机辅助数据库设计可以减轻数据库设计的工作强度，加快数据库设计速度，提高数据库设计质量。但目前计算机辅助数据库设计还只是在数据库设计的某些过程中模拟某一规范设计方法，并以人的知识或经验为主导，通过人机交互实现设计中的某些部分。

3.2.3　数据库设计的步骤

通过分析、比较与综合各种常用的数据库规范设计方法，可将据库设计分为 6 个阶段，如图 3-2 所示。

图 3-2　数据库设计的步骤

1. 需求分析

进行数据库设计首先必须准确了解和分析用户需求（包括数据与处理）。需求分析是整个设计过程的基础，是最困难、最耗费时间的一步。需求分析的结果是否准确地反映了用户的实际要求，将直接影响后面各个阶段的设计，并影响设计结果是否合理和实用。

2. 概念结构设计

准确抽象出现实世界的需求后，下一步应该考虑如何实现用户的这些需求。由于数据库逻辑结构依赖于具体的 DBMS，直接设计数据库的逻辑结构会增加设计人员对不同数据库管理系统的数据库模式的理解负担，因此在将现实世界的需求转换为机器世界的模型之前，要先以一种独立于具体数据库管理系统的逻辑描述方法来描述数据库的逻辑结构，即设计数据库的概念结构。概念结构设计是整个数据库设计的关键。

3. 逻辑结构设计

逻辑结构设计是将抽象的概念结构转换为所选用的 DBMS 支持的数据模型，并对其进行优化。

4. 数据库物理设计

数据库物理设计是为逻辑数据模型选取一个最适合应用环境的物理结构，包括存储结构和存取方法。

5. 数据库实施

在数据库实施阶段,设计人员运用 DBMS 提供的数据语言及其宿主语言,根据逻辑设计和物理设计的结果建立数据库,编制并调试应用程序,组织数据入库,并进行试运行。

6. 数据库运行和维护

数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中必须不断地对其进行评价、调整与修改。

设计一个完善的数据库应用系统,往往是这 6 个阶段不断反复的过程。

在数据库设计过程中必须注意以下几个问题。

(1) 数据库设计过程中要注意充分调动用户的积极性。用户的积极参与是数据库设计成功的关键因素之一。用户最了解自己的业务和需求,用户的积极配合能够缩短需求分析的进程,帮助设计人员尽快熟悉业务,更加准确地抽象出用户的需求,减少反复,也使设计出的系统与用户的最初设想更为符合。同时用户参与意见,双方共同对设计结果承担责任,也可以减少数据库设计的风险。

(2) 应用环境的改变、新技术的出现等都会导致应用需求的变化,因此设计人员在设计数据库时必须充分考虑到系统的可扩充性,使设计易于变动。一个设计优良的数据库系统应该具有一定的可伸缩性,应用环境的改变和新需求的出现一般不会推翻原设计,不会对现有的应用程序和数据造成大的影响,而只是在原设计基础上做一些扩充即可满足新的要求。

(3) 系统的可扩充性最终都是有一定限度的。当应用环境或应用需求发生巨大变化时,原设计方案可能终将无法再进行扩充,必须推倒重来,这时就会开始一个新的数据库设计的生命周期。但在设计新数据库应用的过程中,必须充分考虑到已有应用,尽量使用户能够平稳地从旧系统迁移到新系统。

3.3 关系数据库标准语言——SQL

SQL(Structured Query Language,结构化查询语言)是 1974 年由 Boyce 和 Chamberlin 提出的。1975—1979 年,IBM 公司 San Jose Research Laboratory 研制的关系数据库管理系统的原型系统 System R 实现了这种语言。由于它功能丰富,语言简洁,使用方法灵活,备受用户及计算机工业界欢迎,被众多计算机公司和软件公司所采用。经各公司的不断修改、扩充和完善,SQL 最终发展成为关系数据库的标准语言。1986 年 10 月美国国家标准局(ANSI)公布将 SQL 作为关系数据库语言的美国标准,1987 年国际标准化组织(ISO)也通过了这一标准。

自 SQL 成为国际标准语言以后,各个数据库厂家纷纷推出各自支持的 SQL 软件或与 SQL 兼容的接口软件。这就有可能使将来大多数数据库均用 SQL 作为共同的数据存取语言和标准接口,使不同数据库系统之间的交互操作有了共同的基础。

SQL 成为国际标准,对数据库以外的领域也产生了很大影响,有不少软件产品将 SQL 的数据查询功能与图形功能、软件工程工具、软件开发工具、人工智能程序结合起来。SQL 已成为关系数据库领域中一个主流语言。

SQL 集数据查询、数据操纵、数据定义和数据控制功能于一体。它是一个综合的、通用的、功能极强,同时又简洁易学的语言。其主要特点包括以下几个方面。

1. 综合统一

非关系模型(层次模型、网状模型)的数据语言一般分为模式数据定义语言(Data Definition Language,模式 DDL)、外模式数据定义语言(外模式 DDL),子模式数据定义语言(子模式 DDL)及数据操纵语言(Data Manipulation Language,DML),它们分别完成模式、外模式、内模式的定义和数据存取、处理等功能。而 SQL 则集数据定义语言(DDL)、数据操纵语言(DML)、数据控制语言(Data Control Language,DCL)的功能于一体,语言风格统一,可以独立完成数据库生命周期中的全部活动。包括定义关系模式、录入数据以建立数据库、查询、更新、维护、数据库重构、数据库安全性控制等一系列操作的要求,这就为数据库应用系统开发提供了良好的环境。

2. 高度非过程化

非关系数据模型的数据操纵语言是面向过程的语言。要完成某项请求,必须指定存取路径。而用 SQL 进行数据操作,用户只需提出"做什么",而不必指明"怎么做"。因此用户无须了解存取路径,存取路径的选择以及 SQL 语句的操作过程由系统自动完成。这不但大大减轻了用户负担,而且有利于提高数据独立性。

3. 面向集合的操作方式

非关系数据模型采用的是面向记录的操作方式,操作对象是一条记录。例如,查询所有平均分在 60 分以上的学生姓名,用户必须一条一条地把满足条件的学生记录找出来(通常要说明具体处理过程,即按照哪条路径、如何循环等)。而 SQL 采用集合操作方式,不仅操作对象、查找结果可以是元组的集合,而且一次插入、删除、更新操作的对象也可以是元组的集合。

4. 用同一种语法结构提供两种使用方式

SQL 既是自含式语言,又是嵌入式语言。作为自含式语言,它能够独立地用于联机交互的使用方式,用户可以在终端键盘上直接输入 SQL 命令对数据库进行操作。作为嵌入式语言,SQL 语句能够嵌入高级语言(如 C、COBOL、FORTRAN、PL/1)程序中,供程序员设计程序时使用。而在两种不同的使用方式下,SQL 的语法结构基本上是一致的。这种以统一的语法结构提供两种不同的使用方式的做法,为用户提供了极大的灵活性与方便性。

5. 语言简洁,易学易用

SQL 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、数据控制的核心功能只用了 9 个命令动词:CREATE、DROP、ALTER、SELECT、INSERT、UPDATE、DELETE、GRANT 和 REVOKE,如表 3-8 所示。而且 SQL 语法简单,接近英语口语,因此容易学习和使用。

表 3-8　SQL 的命令动词

SQL 功能	动　　词
数据查询	SELECT
数据定义	CREATE,DROP,ALTER
数据操纵	INSERT,UPDATE,DELETE
数据控制	GRANT,REVOKE

3.4 数据库保护

数据库系统中的数据是由 DBMS 统一管理和控制的,为了适应数据共享的环境,DBMS 必须提供数据的安全性、完整性、并发控制和数据库恢复等数据保护能力,以保证数据库中数据的安全可靠和正确有效。

3.4.1 安全性

1. 数据库的安全性

数据库的安全性是指保护数据库,防止因用户非法使用数据库造成数据泄漏、更改或破坏。

数据库的一大特点是数据可以共享,但数据共享必然带来数据库的安全性问题。数据库中放置了组织、企业和个人的大量数据,其中许多数据可能是非常关键的、机密的或者涉及个人隐私的。如果 DBMS 不能严格地保证数据库中数据的安全性,就会严重制约数据库的应用。

因此,数据库系统中的数据共享不能是无条件的共享,而必须是在 DBMS 统一严格的控制之下,只允许有合法使用权限的用户访问允许他存取的数据。数据库系统的安全保护措施是否有效是数据库系统主要的性能指标之一。

2. 安全性控制的一般方法

在数据库中用于安全性控制的方法主要有用户标识和鉴定、存取控制、定义视图、审计和数据加密等。

(1) 用户标识和鉴定。

用户标识和鉴定是系统提供的最外层安全保护措施。其方法是由系统提供一定的方式让用户标识自己的名字或身份。系统内部记录着所有合法用户的标识,每次用户要求进入系统时,由系统将用户提供的身份标识与系统内部记录的合法用户标识进行核对,通过鉴定后才提供机器使用权。用户标识和鉴定的方法有很多种,而且在一个系统中往往是多种方法并举,以获得更强的安全性。

标识和鉴定一个用户最常用的方法是用一个用户名或用户标识号来标明用户身份,系统鉴别此用户是否是合法用户,若是则可进入下一步的核实;若不是,则不能使用计算机。

为了进一步核实用户,在用户输入了合法用户名或用户标识号后,系统常常要求用户输入口令(Password),然后系统核对口令以鉴别用户身份。为保密起见,用户在终端上输入的口令是不显示在屏幕上的。

通过用户名和口令来鉴定用户的方法简单易行,但用户名与口令容易被人窃取,因此还可以用更复杂的方法。例如,每个用户都预先约定好一个计算过程或者函数,鉴别用户身份时,系统提供一个随机数,用户根据自己预先约定的计算过程或者函数进行计算,系统根据用户计算结果是否正确进一步鉴定用户身份。用户可以约定比较简单的计算过程或函数,以使计算起来方便;也可以约定比较复杂的计算过程或函数,以使安全性更好。

在一些安全性要求比较高的数据库中,还可以采取生物特征(如指纹、虹膜和掌纹等)识别和智能卡鉴别的用户标识和鉴定方式。

（2）存取控制。

在数据库系统中，为了保证用户只能访问他有权存取的数据，必须预先对每个用户定义存取权限。对于通过鉴定获得上机权的用户（即合法用户），系统根据他的存取权限定义对他的各种操作请求进行控制，确保他只执行合法操作。

存取权限是由两个要素组成的：数据对象和操作类型。定义一个用户的存取权限就是要定义这个用户可以在哪些数据对象上进行哪些类型的操作。在数据库系统中，定义存取权限称为授权。这些授权定义经过编译后存放在数据字典中。对于获得上机权后又进一步发出存取数据库操作的用户，DBMS 查找数据字典，根据其存取权限对操作的合法性进行检查，若用户的操作请求超出了定义的权限，系统将拒绝执行此操作。一组与数据库操作相关的权限称为角色。可以通过为具有相同权限的用户创建一个角色来管理数据库的权限，从而简化授权过程。

（3）定义视图。

进行存取权限的控制，不仅可以通过授权与收回权限来实现，还可以通过定义视图来提供一定的安全保护功能。

视图是从基本表或其他视图中导出的表，它本身不独立存储在数据库中，也就是说，数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的表中，因此视图是一个虚表。

在关系系统中，为不同的用户定义不同的视图，通过视图机制把要保密的数据对无权存取这些数据的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

（4）审计。

审计追踪使用的是一个专用文件或数据库，系统自动将用户对数据库的所有操作记录在上面，利用审计追踪的信息，就能重现导致数据库现有状况的一系列事件，从而找到非法存取数据的人。

审计通常是很费时间和空间的，所以 DBMS 往往都将其作为可选特征，允许 DBA 根据应用对安全性的要求，灵活地打开或关闭审计功能。审计功能一般主要用于安全性要求较高的部门。

（5）数据加密。

对于高度敏感性数据，如财务数据、军事数据、国家机密等，除以上安全性措施外，还可以采用数据加密技术，以密码形式存储和传输数据。这样当企图通过不正当渠道获取数据时，例如，利用系统安全措施的漏洞非法访问数据，或者在通信线路上窃取数据，只能看到一些无法辨认的二进制代码。用户正常检索数据时，首先要提供密码钥匙，由系统进行译码后，才能得到可识别的数据。

所有提供加密机制的系统必然也提供相应的解密程序。这些解密程序本身也必须具有一定的安全性保护措施，否则数据加密的优点也就遗失殆尽了。

由于数据加密与解密是比较费时的操作，而且数据加密与解密程序会占用大量系统资源，因此数据加密功能通常也作为可选特征，允许用户自由选择，只对高度机密的数据加密。

3.4.2 完整性

1. 数据库的完整性

数据库的完整性是指数据的正确性和相容性。例如,学生的性别只能是男或女,学生的姓名是字母或汉字,学号必须唯一等。数据库是否具备完整性关系到数据库系统能否真实地反映现实世界,因此维护数据库的完整性是非常重要的。

2. 安全性与完整性的区别

数据的完整性与安全性是数据库保护的两个不同方面。安全性是防止用户非法使用数据库,包括恶意破坏数据和越权存取数据。完整性则是防止合法用户使用数据库时向数据库中加入不符合语义的数据。也就是说,安全性措施的防范对象是非法用户和非法操作,完整性措施的防范对象是不符合语义的数据。

3. 数据库的完整性约束条件和完整性控制机制

为维护数据库的完整性,DBMS 必须提供一种机制来检查数据库中的数据,看其是否满足语义规定的条件。这些加在数据库数据之上的语义约束条件称为数据库的完整性约束条件,它们作为模式的一部分存入数据库中。而 DBMS 中检查数据是否满足完整性条件的机制称为完整性控制机制。

(1) 数据库的完整性约束条件。

完整性约束条件作用的对象可以有列级、元组级和关系级 3 种精度。完整性约束条件涉及的这 3 种对象,其状态可以是静态的,也可以是动态的。其中,对静态对象的约束是反映数据库状态合理性的约束,这是最重要的一类完整性约束。对动态对象的约束是反映数据库状态变迁的约束。

综合上述两个方面,可以将完整性约束条件分为以下 6 类。

① 静态列级约束。静态列级约束是对一个列的取值域的说明,这是最常见、最简单同时也是最容易实现的一类完整性约束,包括以下几个方面。

- 对数据类型的约束,包括数据的类型、长度、单位、精度等。
- 对数据格式的约束。
- 对取值范围或取值集合的约束。
- 对空值的约束。空值表示未定义或未知的值,它与零值和空格不同。有的列允许空值,有的列则不允许。
- 其他约束,如关于列的排序说明、组合列等。

② 静态元组约束。静态元组约束是规定组成一个元组的各个列之间的约束关系。例如,工资关系中的实发工资=应发—应扣,订货关系中发货量不得超过订货量等。

静态元组约束只局限在单个元组上,因此比较容易实现。

③ 静态关系约束。静态关系约束是一个关系的各个元组之间或者若干关系之间存在的各种联系或约束。常见的静态关系约束有以下 4 种。

- 实体完整性约束。
- 参照完整性约束。
- 函数依赖约束。
- 统计约束:指一个关系的某个字段的值与该关系的多个元组的统计值之间的关系。

④ 动态列级约束。动态列级约束是修改列定义或列值时应满足的约束条件。

⑤ 动态元组约束。动态元组约束是指修改某个元组的值时需要参照其旧值，并且新、旧值之间需要满足某种约束条件。

⑥ 动态关系约束。动态关系约束是加在关系变化前后状态上的限制条件。动态关系约束实现起来开销较大。

(2) 完整性控制机制。

DBMS 的完整性控制机制具有以下 3 个方面的功能。

① 定义功能，即提供定义完整性约束条件的机制。

② 检查功能，即检查用户发出的操作请求是否违背了完整性约束条件。

③ 操作功能，如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性。

3.4.3　并发控制

数据库是一个共享资源，可以供多个用户使用。这些用户程序可以一个一个地串行执行，每个时刻只有一个用户程序运行，执行对数据库的存取，其他用户程序必须等到这个用户程序结束以后方能对数据库存取。如果一个用户程序涉及大量数据的输入/输出交换，则数据库系统的大部分时间将处于闲置状态。为了充分利用数据库资源，发挥数据库共享资源的特点，应该允许多个用户并行地存取数据库。但这样就会产生多个用户程序并发存取同一数据的情况，若对并发操作不加控制就可能会存取不正确的数据，破坏数据库的一致性。因此数据库管理系统必须提供并发控制机制。并发控制机制的好坏是衡量一个数据库管理系统性能的重要标志之一。

1. 事务

事务是数据库的逻辑工作单位，它是用户定义的一组操作序列，这些操作要么都做，要么都不做。在关系数据库中，一个事务可以是一组 SQL 语句、一条 SQL 语句或整个程序。通常情况下，一个应用程序包括多个事务。

事务以 BEGIN TRANSACTION 开始，以 COMMIT 或 ROLLBACK 结束。

原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)是事务的 4 个特性，通常称为事务的 ACID 特性。一个事务必须满足这 4 个特性。

(1) 原子性。事务是数据库的逻辑工作单位，事务中包括的诸操作要么都做，要么都不做。这就是事务的原子性。

(2) 一致性。事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。

(3) 隔离性。对并发执行而言，一个事务的执行不能被其他事务干扰。一个事务内部的操作及使用的数据对其他并发事务是隔离的，同时并发执行的各个事务之间也不能互相干扰。

(4) 持续性。一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。

2. 并发操作带来的数据不一致性

对并发操作如果不进行合适的控制，可能会导致数据库中数据的不一致性。并发操作带来的数据不一致性问题包括 3 类：丢失修改、不可重复读和读“脏”数据。

(1) 丢失修改(Lost Update)。丢失修改是指事务 1 与事务 2 从数据库中读入同一数据并修改,事务 2 的提交结果破坏了事务 1 提交的结果,导致事务 1 的修改被丢失。一个最常见的例子是飞机订票系统中的订票操作。该系统中的一个活动序列如下。

① 甲售票员读出某航班的机票余额 A,设 $A=16$。

② 乙售票员读出同一航班的机票余额 A,也为 16。

③ 甲售票点卖出一张机票,修改机票余额 $A=A-1$,$A=15$,把 A 写回数据库。

④ 乙售票点也卖出一张机票,修改机票余额 $A=A-1$,$A=15$,把 A 写回数据库。

结果明明卖出两张机票,数据库中机票余额却只减少了 1。

(2) 不可重复读(Non-repeatable Read)。不可重复读是指事务 1 读取数据后,事务 2 执行更新操作,使事务 1 无法再现前一次的读取结果。

根据操作的不同,通常有三类不可重复读的类型。当有事务 1 读取某一数据后:

① 事务 2 对其做了修改,当事务 1 再次读该数据时,得到与前一次不同的值。

② 事务 2 删除了其中部分记录,当事务 1 再次读取数据时,发现某些记录神秘地消失了。

③ 事务 2 插入了一些记录,当事务 1 再次按相同条件读取数据时,发现多了一些记录。

其中,后两种不可重复读有时也称为幻影现象(Phantom Row)。

(3) 读"脏"数据(Dirty Read)。读"脏"数据是指事务 1 修改某一数据,并将其写回磁盘,事务 2 读取同一数据后,事务 1 由于某种原因被撤销,这时事务 1 已修改过的数据恢复原值,事务 2 读到的数据就与数据库中的数据不一致,是不正确的数据,称为"脏"数据。

3. 并发控制

并发控制就是要用正确的方式调度并发操作,避免造成数据的不一致性,使一个用户事务的执行不受其他事务的干扰。

4. 并发控制的主要方法

并发控制的主要方法是采用封锁机制。封锁就是事务 T 在对某个数据对象(如表、记录等)操作之前,先向系统发出请求,对其加锁。加锁后事务 T 就对该数据对象有了一定的控制,在事务 T 释放它的锁之前,其他的事务不能更新此数据对象。

例如,在前面的飞机订票系统中,甲事务要修改数据 A,则在读出 A 前先封锁 A。这时其他事务就不能读取和修改数据 A 了,直到甲修改并写回 A 后,解除了对 A 的封锁为止。这样,就不会丢失甲的修改操作了。

DBMS 通常提供了多种类型的封锁。一个事务对某个数据对象加锁后究竟拥有什么样的控制是由封锁的类型决定的。基本封锁类型包括排他锁(eXclusive lock,X 锁)和共享锁(Share lock,S 锁)。

排他锁又称为写锁。若事务 T 对数据对象 A 加上 X 锁,则只允许 T 读取和修改 A,其他任何事务都不能再对 A 加任何类型的锁,直到 T 释放 A 上的锁。共享锁又称为读锁。若事务 T 对数据对象 A 加上 S 锁,则其他事务只能再对 A 加 S 锁,而不能加 X 锁,直到 T 释放 A 上的 S 锁。

3.4.4 数据库恢复

当前计算机硬、软件技术已经发展到相当高的水平,但硬件的故障、系统软件和应用软

件的错误、操作员的失误以及恶意的破坏等仍然是不可避免的。为了保证各种故障发生后，数据库中的数据都能从错误状态恢复到某种逻辑一致的状态，数据库管理系统中恢复子系统是必不可少的。数据库系统所采用的恢复技术是否行之有效，不仅对系统的可靠程度起着决定性作用，而且对系统的运行效率也有很大影响，是衡量系统性能优劣的重要指标。

1. 恢复的原理

数据库恢复主要是利用存储在系统其他地方的冗余数据来重建或修复数据库中已经被破坏或已经不正确的那部分数据。恢复的基本原理虽然简单，但实现的技术却相当复杂。一般一个大型数据库产品，恢复子系统的代码要占全部代码的 10%以上。

2. 恢复的实现技术

恢复就是利用存储在系统其他地方的冗余数据来重建或修复数据库中被破坏的或不正确的数据。因此数据库的恢复包括以下两步。

(1) 建立冗余数据。建立冗余数据最常用的技术是数据转储和登录日志文件。通常在一个数据库系统中，这两种方法是一起使用的。

① 数据转储。数据转储是指 DBA 将整个数据库复制到磁带或另一个磁盘上保存起来的过程。这些备用的数据文本称为后备副本或后援副本。一旦系统发生介质故障，数据库遭到破坏，可以将后备副本重新装入，把数据库恢复起来。

② 登录日志文件。日志文件是用来记录事务对数据库的更新操作的文件。不同的数据库系统采用的日志文件格式并不完全一样。

(2) 利用冗余数据实施数据库恢复。当系统运行过程中发生故障时，利用数据库后备副本和日志文件就可以将数据库恢复到故障前的某个一致性状态。对于不同故障，其恢复技术也不同。

① 事务故障的恢复。事务故障是指事务在运行过程中由于某种原因，如输入数据的错误、运算溢出、违反了某些完整性限制、某些应用程序的错误以及并行事务发生死锁等，使事务未运行至正常终止点就夭折了。这时恢复子系统应撤销此事务已对数据库进行的修改，具体做法如下。

- 反向扫描文件日志，查找该事务的更新操作。
- 对该事务的更新操作执行逆操作。
- 继续反向扫描日志文件，查找该事务的其他更新操作，并做同样处理。
- 如此处理下去，直至读到此事务的开始标记，事务故障恢复就完成了。

事务故障的恢复是由系统自动完成的，不需要用户干预。

② 系统故障的恢复。系统故障是指系统在运行过程中由于某种原因，如操作系统或 DBMS 代码错误、操作员操作失误、特定类型的硬件错误(如 CPU 故障)、突然停电等造成系统停止运行，致使所有正在运行的事务都以非正常方式终止。此恢复操作就是要撤销故障发生时未完成的事务，重做已完成的事务。具体做法如下。

- 正向扫描日志文件(即从头扫描日志文件)，找出在故障发生前已经提交的事务，将其事务标识记入重做队列；同时还要找出故障发生时尚未完成的事务，将其事务标识记入撤销队列。
- 对撤销队列中的各个事务进行撤销理。
- 对重做队列中的各个事务进行重做处理。

进行重做处理的方法是：正向扫描日志文件，对每个重做事务重新执行登记的操作。系统故障的恢复也是由系统自动完成的，不需要用户干预。

③ 介质故障的恢复。发生介质故障后，磁盘上的物理数据和日志文件被破坏，这是最严重的一种故障。

恢复介质故障的方法是重装数据库，然后重做已完成的事务。具体做法如下。

- 装入最新的后备数据库副本，使数据库恢复到最近一次转储时的一致性状态。
- 装入有关的日志文件副本，重做已完成的事务。

这样就可以将数据库恢复至故障前某一时刻的一致状态了。

介质故障的恢复需要 DBA 介入，但 DBA 只需要重装最近转储的数据库副本和有关的各日志文件副本，然后执行系统提供的恢复命令即可，具体的恢复操作仍由 DBMS 完成。

3.5 数据库新技术

3.5.1 数据库技术与其他技术的结合

数据库技术与其他学科的内容相结合，是新一代数据库技术的一个显著特征。在结合中涌现出各种新型的数据库，例如：

- 数据库技术与分布处理技术相结合，出现了分布式数据库。
- 数据库技术与并行处理技术相结合，出现了并行数据库。
- 数据库技术与人工智能相结合，出现了演绎数据库、知识库和主动数据库。
- 数据库技术与多媒体处理技术相结合，出现了多媒体数据库。
- 数据库技术与模糊技术相结合，出现了模糊数据库。
- 数据库技术与移动通信技术相结合，出现了移动数据库系统。
- 数据库技术与 Web 技术相结合，出现了 Web 数据库等。

3.5.2 大数据

当前，人们从不同的角度诠释大数据的内涵。一般意义上大数据是指无法在可容忍的时间内用现有的 IT 技术和软硬件工具对其进行感知、获取、管理、处理和服务的数据集合。大数据通常被认为是 PB(10^3TB)或 EB(1EB=10^6TB)或更高数量级的数据。其规模或复杂程度超出了传统数据库和软件技术所能管理和处理的数据集范围。

1. 大数据的特征

大数据不仅是量“大”，它具有许多重要的特征。专家们归纳为若干个 V，即巨量(Volume)、多样(Variety)、快变(Velocity)、价值(Value)和真实性(Veracity)。大数据的这些特征给我们带来了巨大的挑战。

2. 大数据的关键技术

目前，大数据所涉及的关键技术主要包括数据的采集和迁移、数据的存储和管理、数据的处理和分析、数据安全和隐私保护。

数据采集技术将分布在异构数据源或异构采集设备上的数据通过清洗、转换和集成技术，存储到分布式文件系统中，成为数据分析、挖掘和应用的基础。

数据迁移技术将数据从关系型数据库迁移到分布式文件系统或 NoSQL 数据库中。NoSQL 数据库是一种非结构化的新型分布式数据库，它采用键值对的方式存储数据，支持超大规模数据存储，可灵活地定义不同类型的数据库模式。

数据处理和分析技术利用分布式并行编程模型和计算框架，如 Hadoop 和 MapReduce 计算框架和 Spark 的混合计算框架等，结合模式识别、人工智能、机器学习、数据挖掘等算法，实现对大数据的离线分析和大数据流的在线分析。

数据安全和隐私保护是指在确保大数据被良性利用的同时，通过隐私保护策略和数据安全等手段，构建大数据环境下的数据隐私和安全保护。

3. 大数据的应用

目前，大数据技术的应用已经非常广泛，涉及的领域包括传统零售业、金融业、医疗业和政府机构等。

在传统零售行业中，用户购物的大数据可用于分析具有潜在购买关系的商品，经销商将分析得到的关联商品以搭配的形式进行销售，从而提高相关商品的销售概率。这类应用的经典案例是“啤酒和尿布”的搭配，两种产品看似是无关的，但是从购买记录中发现，购买啤酒的用户通常会购买尿布，如果将两者就近摆放，则会综合提高两种商品的销售数量。

在金融业中，每日股票交易的数据量具有大数据的特点，很多金融公司纷纷成立金融大数据研发机构，通过大数据技术分析市场的宏观动向并预测某些公司的运行情况。同时，银行可以根据区域用户日常交易情况，将常用的业务放置在区域内 ATM 机器上，方便用户更快捷地使用所需的金融服务。

在医疗行业中，各类患者的诊断信息、检查信息和处方信息可用于预测、辨别和辅助各种医疗活动，代表性的案例如“癌症的预测”。研究发现，很多症状能够用于早期的癌症预测，但由于传统医疗数据量较小，导致预测结果精度不高。随着大数据技术与医疗大数据的深度结合，越来越多有意义的癌症指征被发现并用于早期的癌症预测中。

在政府机构中，其掌握的各类大数据对政府的决策具有重要的辅助作用。传统的出租车 GPS 信息，只用于掌握出租车的运行情况，目前这一数据可用于预测各主要街道的拥堵情况，从而对未来的市政建设提供决策依据。再有，药店销售的感冒药数量不仅可用于行业的基本监督，还可用于预测当前区域的流感发病情况等。

小结

关系数据库是由一组关系组成的，那么针对一个具体问题，应该构造几个关系，每个关系由哪些属性组成等，这是关系数据库的规范化问题。通过对关系模式进行规范化，可以逐步消除数据依赖中不合适的部分，使关系模式达到更高的规范化程度。

数据库设计的任务是在 DBMS 的支持下，按照应用的要求，为某一部门或组织设计一个结构合理、使用方便、效率较高的数据库及其应用系统。数据库设计应包含两方面的内容：一是结构设计，二是行为设计。

SQL 集数据查询、数据操纵、数据定义和数据控制功能于一体。它功能丰富，语言简洁，使用方法灵活，备受用户及计算机工业界欢迎，已发展成为关系数据库的标准语言。

数据库系统中的数据是由 DBMS 统一管理和控制的。DBMS 必须提供数据的安全性、

完整性、并发控制和数据库恢复等数据保护能力，以保证数据库中数据的安全可靠和正确有效。

大数据是指无法在可容忍的时间内用现有的 IT 技术和软硬件工具对其进行感知、获取、管理、处理和服务的数据集合。目前，其应用领域包括传统零售业、金融业、医疗业和政府机构等。

习题

1. 解释下列术语：函数依赖，平凡函数依赖，非平凡函数依赖，完全函数依赖，部分函数依赖，传递函数依赖，候选码，主码，1NF，2NF，3NF，多值依赖，BCNF，4NF。

2. 为什么要对关系模式进行规范化？

3. $R<U,F>$中，$U=$｛SNO，SDEPT，MNAME，CNAME，GRADE｝

（SNO 学号；SDEPT 所在系；MNAME 系主任名；CNAME 课程名；GRADE 分数）。

有关语义如下：一个系只有一个系主任，一个学生可以选择多门课程，一门课程可以被多个学生所选择。写出 U 上的极小函数依赖，把该关系规范化为 3NF。

4. 设有关系模式 R（运动员编号，比赛项目，成绩，比赛类别，比赛主管），如果规定：每个运动员每参加一个比赛项目，只有一个成绩；每个比赛项目只属于一个比赛类别；每个比赛类别只有一个比赛主管。写出该关系的主码和 R 上的极小函数依赖，把该关系规范化为 3NF。

5. 什么是数据库的安全性？数据库安全性控制的常用方法有哪些？

6. 什么是数据库的完整性？它与安全性有什么区别？

7. 并发操作可能会产生哪几类数据不一致？

8. 什么是数据库的恢复？恢复的实现技术有哪些？

9. 试述数据库设计的步骤。