数据库知识点④

2016年5月24日

09:18

**1.函数依赖的定义**

(a)平凡的函数依赖与非平凡的函数依赖

(b)函数依赖是语义范畴的概念

©函数依赖关系的存在与时间无关

(d)函数依赖可以保证关系分解的无损连接性

**2.函数依赖的基本性质 SCD(SNO,SN,AGE,DEPT,MN,CNO,,SCORE)**

(a)投影性:一组属性函数决定它的子集

举例：(SNO,CNO)->SNO 和(SNO,CNO)->CNO

(b)扩张性: 若x->y且w->z 则有(x,w)->(y,z)

举例：SNO->(SN,AGE) DEPT->MN 则有(SNO,DEPT)->(SN,AGE,MN)

©合并性：若x->y 且x->z则有x->(y,z)

举例：SNO->(SN,AGE) SNO->(DEPT,MN) 则有SNO->(SN,AGE,DEPT,MN)

(d)分解性：若x->(y,z)则有x->y, x->z

**3.完全函数依赖与部分函数依赖**

1.如果X→Y，并且对于X的任何一个真子集X′,都有X′ Y，则称Y对X完全函数依赖（Full Functional Dependency），记作 X -(f)->Y

2.如果对X的某个真子集X′，有X′→Y，则称Y对部分函数依赖（Partial Functional Dependency) 记作X -(p)-> Y。

举例:

在关系模式SCD中

因为SNO -\->SCORE，且CNO -\-> SCORE，所以有：（SNO，CNO)-(f)->SCORE 。

而SNO→AGE，所以（SNO，CNO)-(p)->AGE

3.只有当决定因素是组合属性时，讨论部分函数依赖才有意义

4.当决定因素是单属性时，只能是完全函数依赖。

举例：在关系模式S（SNO，SN，AGE，DEPT），决定因素为单属性SNO，有SNO→（SN，AGE，DEPT），不存在部分函数依赖。

**4.传递函数依赖**

1.定义：

设有关系模式R（U），U是属性全集，X，Y，Z是U的子集，

若X→Y，但Y -\->X，而Y→Z（Y 不属于X，Z 不属于Y），则称Z对X传递函数依赖（Transitive Functional Dependency），记作：X-(t)-> Z。

如果Y→X，则X <->Y，这时称Z对X**直接函数依赖，而不是传递函数依赖**。

举例：

例如，在关系模式SCD中，SNO→DEPTN，但DEPTN -\->SNO，而DEPTN→MN，则有SNO-(t)->MN。系主任是对学号的传递依赖

当学生不存在重名的情况下，有SNO→SN，SN→SNO，SNO <->SN，SN→DEPTN，这时DEPTN对SNO是直接函依赖，而不是传递函数依赖。

**5.总结函数依赖**

综上所述，函数依赖分为**完全函数依赖、部分函数依赖**和**传递函数依赖**三类，它们是规范化理论的依据和规范化程度的准则，下面我们将以介绍的这些概念为基础，进行数据库的规范设计。

**6.范式**

规范化的基本思想是**消除关系模式中的数据冗余，消除数据依赖中的不合适的部分，解决数据插入、删除时发生异常现象**。

这就要求关系数据库设计出来的关系模式要满足一定的条件。关系数据库的规范化过程中：不同程度的规范化要求设立的不同标准称为范式（Normal Form）。**规范化的程度不同，产生不同的范式**。满足最基本规范化要求的关系模式叫第一范式，在第一范式中**进一步满足**一些要求为第二范式，以此类推就产生了第三范式等概念。

每种范式都规定了一些限制约束条件。

**7.范式发展过程**

范式的概念最早由E.F.Codd提出。

从1971年起，Codd相继提出了关系的三级规范化形式，即第一范式（1NF）、第二范式（2NF）、第三范式（3NF）。

1974年，Codd和Boyce以共同提出了一个新的范式的概念，即Boyce-Codd范式，简称BC范式。

1976年Fagin提出了第四范式，

后来又有人定义了第五范式。

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,5NF,一级比一级有更严格的要求。

计算机生成了可选文字:


**8.第一范式（First Normal Form）**

第一范式是最基本的规范形式，即关系中每个属性都是不可再分的简单项。

定义:

如果关系模式R，其所有的属性均为简单属性，即**每个属性域都是不可再分的**，则称R属于第一范式，简称1NF，记作R1NF。

然而，一个关系模式仅仅属于第一范式是不适用的。给出的关系模式SCD属于第一范式，但其具有大量的数据冗余，具有插入异常、删除异常、更新异常等弊端。

为什么会存在这种问题呢？

让我们分析一下SCD中的函数依赖关系，它的关系键是（SNO，CNO）的属性组合，所以有：

(SNO，CNO) -(f)->SCORE

SNO→SN，(SNO,CNO)-(p)-> SN

SNO→AGE，(SNO,CNO)-(p)-> AGE

SNO→DEPT，(SNO,CNO)-(p)-> DEPT

SNO -(t)-> MN，(SNO,CNO)-(p)-> MN

我们可以用函数信赖图表示以上函数依赖关系，如下图所示。

计算机生成了可选文字:
SNO 
SNO 
CNO 
SCD 中 的 函 数 依 赖 关 系 
S DN 

**9.第二范式**

定义 如果关系模式R1NF，且每个非主属性都**完全函数依赖**于R的每个关系键，则称R属于第二范式（Second Normal Form），简称2NF，记作R2NF。

分析：在关系模式SCD中，SNO，CNO为主属性，AGE，DEPT，MN，SCORE均为非主属性，经上述分析，存在非主属性对关系键的部分函数依赖，所以SCD不是2NF。

拆分：而将SCD分解成三个关系模式S(SNO,SN,AGE,DEPT)，D(DEPT,MN)，SC(SNO,CNO,SCORE)，其中S的关系键为SNO，D的关系键为DEPT，都是单属性，不可能存在部分函数依赖。

结果：而对于SC，(SNO，CNO)-(f)->SCORE。所以SCD分解后，消除了非主属性对关系键的部分函数依赖，S，D，SC均属于2NF。

经以上分析，可以得到两个结论：

1．从1NF关系中消除**非主属性**对**关系键**的**部分函数依赖**，则可得到2NF关系。

2．如果R的关系键为单属性，或R的全体属性均为主属性，则R2NF。

**10.1NF分解为2NF规则**

2NF规范化是指把1NF关系模式通过投影分解转换成2NF关系模式的集合。

分解时遵循的基本原则就是“一事一表”，让一个关系只描述一个实体或者实体间的联系。如果多于一个实体或联系，则进行投影分解。

11.举例分析

计算机生成了可选文字:
下 面 以 关 系 模 式 SCD 为 例 ， 来 说 明 2NF 规 范 化 的 过 程 
例 将 SCD (SNO, SN, AGE ， DEPT ， MN, CNO ， SCORE) 规 范 到 2NF 
（ SNO, CNO) 了 
由 SNO—SN, SNO—AGE, SNO—DEPT ， 
一 一 ， SCORE 
可 以 判 断 ， 关 系 SCD 至 少 描 述 了 两 个 实 体 ， 
一 个 为 学 生 实 体 ， 属 性 有 SNO 、 SN 、 AGE 、 DEPT 、 皿 ， 
． 另 一 个 是 学 生 与 课 程 的 联 系 （ 选 课 ） ， 属 性 有 SNO 、 CNO 和 
SCORE. 
． 根 据 分 解 的 原 则 ， 我 们 可 以 将 SCD 分 解 成 如 下 两 个 关 系 ， 如 下 
图 所 示 。 

计算机生成了可选文字:
． SD (SNO, SN ， AGE, DEPT ， (N) ， 描 述 学 生 实 体 ； 
． SC (SNO, CNO, SCORE) ， 描 述 学 生 与 课 程 的 联 系 。 
SD 
计 算 机 
孙 
自 动 化 
SCORE 
刘 伟 
刘 伟 
关 系 SD 和 SC 

计算机生成了可选文字:
对 于 分 解 后 的 两 个 关 系 SD 和 SC ， 主 键 分 别 为 SNO 和 
键 完 全 函 数 依 赖 。 因 此 ， 
SDe2NF ， SCe2NF, 而 且 前 面 己 经 讨 论 ， SCD 的 这 种 分 
解 没 有 丢 失 任 何 信 息 ， 具 有 无 损 连 接 性 。 
． 分 解 后 ， SD 和 SC 的 函 数 依 赖 分 别 如 下 图 所 示 。 
SNO 
AGE 
DEPT 
SNO 
SCORE 
CNO 
SC 中 的 函 数 依 賴 关 系 
SD 中 的 函 数 依 赖 关 系 

改善之处：关系模式SD和SC在性能上比SCD有了显著提高。

(a).1NF的关系模式经过投影分解转换成2NF后，消除了一些数据冗余。

(b)分析图中SD和SC中的数据，可以看出，它们存储的冗余度比关系模式SCD有了较大辐度的降低。

(c)学生的姓名、年龄不需要重复存储多次。

(d)这样便可在一定程度上避免数据更新所造成的数据不一致性的问题。

(e)由于把学生的基本信息与选课信息分开存储，则学生基本信息因没选课而不能插入的问题得到了解决，插入异常现象得到了部分改善。

(f)同样，如果某个学生不再选修C1课程，只在选课关系SC中删去该该学生选修C1的记录即可，而SD中有关该学生的其它信息不会受到任何影响，也解决了部分删除异常问题。

数据库知识点⑤

2016年5月24日

10:43

**1.第三范式**

定义 如果关系模式R2NF，且每个非主属性都不传递依赖于R的每个关系键，则称R属于第三范式（Third Normal Form），简称3NF，记作R3NF。

基本性质：

1．如果R3NF，则R也是2NF。

2．如果R2NF，则R不一定是3NF。

例如，我们前面由关系模式SCD分解而得到的SD和SC都为2NF，其中，SC3NF，但在SD中存在着非主属性MN对主键SNO传递依赖，SD不是3NF。对于SD，应该进一步进行分解，使其转换成3NF。

**2.3NF规范化**

3NF规范化是指把2NF关系模式通过**投影分解**转换成3NF关系模式的集合。

和2NF的规范化时遵循的原则相同，即“一事一表”，让一个关系只描述一个实体或者实体间的联系。

举例：**将之前的SD表进行再分解，使其满足第三范式**

计算机生成了可选文字:
AGE 
DEPT 
SD 中 的 函 数 依 赖 关 系 
SNO 
SCORE 
CNO 
SC 中 的 函 数 依 賴 关 系 

**将SD(SN,AGE,DEPT,MN)分解为S(SN,AGE,DEPT)和D(MN),如下图所示**

计算机生成了可选文字:
SNO 
DEPT 
SLP 的 函 数 依 关 系 图 
DEPT 
DLP 的 函 数 依 关 系 图 

**3.第三范式3NF与第二范式2NF相比**

(a).数据冗余降低。系主任的名字存储的次数与该系的学生人数无关，只在关系D中存储一次。

(b).不存在插入异常。当一个新系没有学生时，该系的信息可以直接插入到关系D中，而与学生关系S无关。

(c).不存在删除异常。要删除某系的全部学生而仍然保留该系的有关信息时，可以只删除学生关系S中的相关学生记录，而不影响系关系D中的数据。

(d)不存在更新异常。更换系主任时，只需修改关系D中一个相应元组的MN属性值，从而不会出现数据的不一致现象。

SCD规范到3NF后，所存在的异常现象已经全部消失。

**4.第三范式3NF的不足**：

但是，3NF**只限制了非主属性对键的依赖关系**，而**没有限制主属性对键的依赖关系**。

如果发生了这种依赖，仍有可能存在数据冗余、插入异常、删除异常和修改异常。

这时，则需对3NF进一步规范化，消除主属性对键的依赖关系，为了解决这种问题，Boyce与Codd共同提出了一个新范式的定义，这就是Boyce-Codd范式，通常简称BCNF或BC范式。它弥补了3NF的不足。

**5.BCNF范式**

定义 如果关系模式R1NF，且所有的函数依赖X→Y（Y ，X）,决定因素X**都包含了R的一个候选键**，则称**R属于BC范式（Boyce-Codd Normal Form）**，记作RBCNF。

基本性质

(a).满足BCNF的关系将消**除任何属性（主属性或非主属性）对键的部分函数依赖和传递函数依赖**。也就是说，如果RBCNF，则R也是3NF。

(b)**如果R3NF，则R不一定是BCNF**。

举例:

设关系模式SNC（SNO，SN，CN0，SCORE），其中SNO代表学号，SN代表学生姓名并假设没有重名，CNO代表课程号，SCORE代表成绩。可以判定，SNC有两个候选键（SNO，CNO）和（SN，CNO），其函数依赖如下：

SNO <->SN

（SNO，CNO）→SCORE

（SN，CNO）→SCORE。

唯一的非主属性SCORE对键**不存在部分函数依赖，也不存在传递函数依赖**。**所以SNC3NF**。

但是，因为SNO <->SN，即决定因素SNO或SN不包含候选键，从另一个角度说，**存在着主属性对键的部分函数依赖**： （SNO，CNO)-(p)-> SN，（SN，CNO)-(p)->SNO，**所以SNC不是BCNF**。

分析：

正是存在着这种主属性对键的部分函数依赖关系，造成了关系SNC中存在着较大的数据冗余，学生姓名的存储次数等于该生所选的课程数。从而会引起修改异常。

比如，当要更改某个学生的姓名时，则必须搜索出现该姓名的每个学生记录，并对其姓名逐一修改，这样容易造成数据的不一致问题。

解决这一问题的办法仍然是通过投影分解进一步提高SNC的范式等级，**将SNC规范到BCNF**。

**6.BCNF范式的规范化**

BCNF规范化是指把3NF关系模式通过投影分解转换成BCNF关系模式的集合。

还是上面的例子：

将SNC(SNO,SN,CNO，SCORE)规范到BCNF。

分析SNC数据冗余的原因，是因为在这一个关系中存在两个实体，一个为学生实体，属性有SNO、SN；另一个是选课实体，属性有SNO、CNO和SCORE。

**根据分解的原则，我们可以将SNC分解成如下两个关系：**

**S1(SNO,SN)，描述学生实体；**

**S2(SNO,CNO,SCORE)，描述学生与课程的联系。**

对于S1，有两个候选键SNO和SN，

对于S2，主键为（SNO，CNO）。

在这两个关系中，**无论主属性还是非主属性都不存在对键的部分依赖和传递依赖**，S1BCNF，S2BCNF。

**7.让我们看一下从1NF到BCNF的转化过程**

**下图是1NF的函数依赖关系图（存在部分依赖以及传递依赖）**

计算机生成了可选文字:
SNO 
SNO 
CNO 
D 中 的 咱 数 依 赖 关 糸 
SDN 

**下图是2NF函数依赖关系图（消除了部分依赖但是存在传递依赖）**

计算机生成了可选文字:
AGE 
DEPT 
SD 中 的 函 数 依 赖 关 系 
SNO 
SCORE 
CNO 
SC 中 的 函 数 依 賴 关 系 

**下图是3NF函数依赖关系图（消除了部分依赖和传递依赖，但是主属性对键存在着部分依赖关系）**

计算机生成了可选文字:
SNO 
DEPT 
SLP 的 函 数 依 关 系 图 
DEPT 
DLP 的 函 数 依 关 系 图 

**下图是BCNF函数依赖关系图（消除了部分依赖和传递依赖，以及主属性对键的部分依赖关系）**

计算机生成了可选文字:
SNO 
Sl 中 的 函 数 依 顿 关 系 
SNO 
SCORE 
CNO 
S2 中 的 函 数 依 赖 关 系 

**8.关系模式的规范化**

定义：

一个低一级范式的关系模式，通过模式分解转化为若干个高一级范式的关系模式的集合，这种分解过程叫作关系模式的规范化（Normalization）

目的：

规范化的目的就是使结构合理，消除存储异常，使数据冗余尽量小，便于插入、删除和更新。

原则：

规范化的基本原则就是遵从概念单一化“一事一表”的原则，即一个关系只描述一个实体或者实体间的联系。

规范化的步骤:

(a).对1NF关系进行投影，消除原关系中**非主属性对键的部分函数依赖**，将1NF关系转换成若干个2NF关系。

(b).对2NF关系进行投影，消除原关系中**非主属性对键的传递函数依赖**，将2NF关系转换成若干个3NF关系。

(c).对3NF关系进行投影，消除原关系中**主属性对键的部分函数依赖和传递函数依赖**，也就是说使决定因素都包含一个候选键。得到一组BCNF关系。

计算机生成了可选文字:
氵 肖 除 决 廡 性 
不 是 候 选 键 的 
菲 平 凡 的 函 数 
依 赖 
2NF 
3NF 
消 除 非 主 属 性 对 键 的 邹 分 函 数 侬 赖 
消 除 菲 主 属 性 对 键 的 传 递 函 数 依 赖 
消 除 主 属 性 对 键 的 部 分 和 传 递 函 数 依 赖 
规 范 化 过 程 

规范化的要求：

关系模式的规范化过程是通过对关系模式的投影分解来实现的，但是投影分解方法不是唯一的，不同的投影分解会得到不同的结果。

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**的方法才是有意义的。

判断对关系模式的一个分解是否与原关系模式等价可以有三种不同的标准：

1．分解要具有无损连接性。

2．分解要具有函数依赖保持性。

3．分解既要具有无损连接性，又要具有函数依赖保持性。

**9.非规范化设计**

使用非规范化设计原因：

(a).许多数据库应用强调性能优先

(b).规范化设计有时会导致数据库运行效率的下降

(c).在特殊条件和要求下，适当地降低甚至抛弃关系模式的范式，不再要求一个表只描述一个实体或者实体间的一种联系。其主要目的在于提高数据库的运行效率。

考虑进行非规范化处理：

(a).大量频繁的查询过程所涉及的表都需要进行连接；

(b).主要的应用程序在执行时要将表连接起来进行查询；

(c).对数据的计算需要临时表或进行复杂的查询。

非规范化处理的主要技术:

包括增加冗余或派生列，对表进行合并、分割或增加重复表。

**10.增加冗余列（用空间换取时间）**

增加冗余列是指在多个表中具有相同的列，它常用来在查询时避免连接操作。

举例：

如果经常检索一门课的任课教师姓名，则需要做class和teacher表的连接查询：

　　select classname,teachername

　　from class,teacher

　　where class.teacherno=teacher.teacherno

这样的话就可以在class表中增加一列teacher－name就不需要连接操作了。

增加冗余列可以在查询时避免连接操作，但它需要更多的磁盘空间，同时增加表维护的工作量。

**11.增加派生列**

增加派生列指增加的列来自其它表中的数据，由它们计算生成。

它的作用是在查询时减少连接操作，避免使用集函数。派生列也具有与冗余列同样的缺点。

**12.重新组表**

重新组表指如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。

举例：

用户经常需要同时查看课程号，课程名称，任课教师号，任课教师姓名，则可把表class(classno,classname,teacherno)和表teacher(teacherno,teachername)合并成一个表class(classno,classname,teacherno,teachername)。

这种方法可以提高性能，但需要更多的磁盘空间，同时也损失了数据在概念上的独立性。

**13.对表进行分割**

表分割有两种方式：

水平分割

垂直分割

水平分割：根据一列或多列数据的值把数据行放到两个独立的表中。

水平分割的使用：

(a).表很大，分割后可以降低在查询时需要读的数据和索引的页数，同时也降低了索引的层数，提高查询速度。

(b).表中的数据本来就有独立性，例如表中分别记录各个地区的数据或不同时期的数据，特别是有些数据常用，而另外一些数据不常用。

需要把数据存放到多个介质上。

举例：

法规表law就可以分成两个表active－law和inactive－law。active－law表中的内容是正生效的法规，是经常使用的，而inactive－law表则使已经作废的法规，不常被查询。

水平分割会给应用增加复杂度，它通常在查询时需要多个表名，查询所有数据需要union操作。在许多数据库应用中，这种复杂性会超过它带来的优点。

垂直分割：把主键和一些列放到一个表，然后把主键和另外的列放到另一个表中

垂直分割的使用：

如果一个表中某些列常用，而另外一些列不常用，则可以采用垂直分割加快查询速度。其缺点是需要管理冗余列，查询所有数据需要join操作。

**14.非规范化设计总结**

主要优点：

(a).减少了查询操作所需的连接

(b).减少了外部键和索引的数量

(c).可以预先进行统计计算，提高了查询时的响应速度

主要问题：

(a).增加了数据冗余

(b).影响数据库的完整性

(c).降低了数据更新的速度

(d)增加了存储表所占用的物理空间