课程名称: 数据库系统

关系模式设计优化

单 位：重庆大学计算机学院

评判标准

- 设计具有主观性，不同的设计师设计出来的作品都有不同的特点，如何客观评判这些作品呢？
- 可以通过判断作品是否达到不同等级规范标准，来相对客观地评价作品的等级。

主要目标

- BCNF和3NF定义和分解算法
- 理解BCNF和3NF的区别。

思考问题

- 关系数据库设计的规范标准是什么？
- 不同等级之间有什么区别？

1. 范式

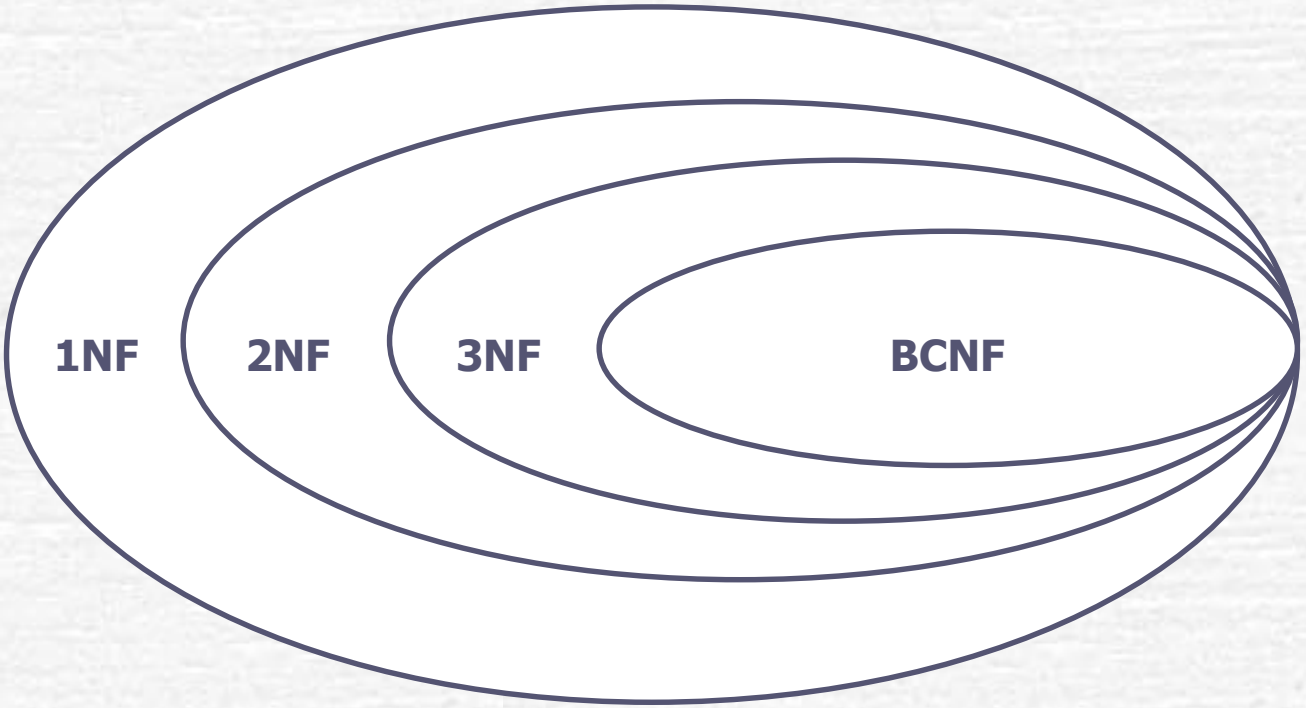
- 第一范式（1NF）
  - 属性原子性
- 第二范式（2NF）
  - 满足1NF
  - 非主属性**完全依赖**于候选键，不依赖于候选键的子集
- 第三范式（3NF）
  - 满足2NF
  - 非主属性**直接依赖**于候选键，不存在**传递依赖**（非主属性间的依赖）

选课表（学号, 课程号, 姓名, 成绩）
- 候选键：(学号, 课程号)
- 主属性：学号，课程号
- 非主属性：姓名，成绩

学生表（学号，姓名，年龄，学院名称，学院电话）
- 主键/候选键：学号
- 因为存在依赖：学号->学院名称，学院名称->学院电话
- 符合第二范式，但不符合第三范式

1. 范式

- Boyce-Codd 范式(BCNF)
  - 任何符合BCNF的关系也符合 3NF



2. Boyce-Codd 范式(BCNF)

- 对于 R 有函数依赖集 FDs *F*，*如果R符合BCNF 当且仅当每一个非平凡的* FD
  X → A in *F*，*X 是R的超键*
  (i.e., X → R in *F+*).
  - 对于 FD X → A 是平凡的，当且仅当 A ⊆ X.
- 也就是说，R 符合BCNF 当且仅当非平凡的FDs 箭头左侧是键.
- BCNF:
  - R中没有数据能够使用FDs预测.

Why:

- X是超键，不会有两个元组的X值相同

X	Y	A
x	y1	a
x	y2	?

Suppose we know that this instance satisfies X → A. This situation cannot arise if the relation is in BCNF.

2. Boyce-Codd 范式(BCNF)

- 如果R只有两个属性，那么它符合BCNF
- 如果F只包括R中的属性：
  - R 符合BCNF *当且仅当每一个F中的函数依赖 X → Y (not F+!), X 是 R的超键, i.e., X → R is* in F+ (not F!).

证明：为什么**F**就行了？采用**RAT**公理！

BCNF的检测

- 列出所有的非平凡函数依赖
- 确认每一个函数依赖箭头左边的属性集是R的超键。
- 注意：我们需要首先找出R的超键!

BCNF的检测

- Courses(course_num, dept_name, course_name, classroom, enrollment, student_name, address) 符合BCNF?
- FDs 包括:
  - course_num, dept_name → course_name
  - course_num, dept_name → classroom
  - course_num, dept_name → enrollment
- 那么(course_num, dept_name)+?
  - {course_num, dept_name, course_name, classroom, enrollment}
- Therefore, the key is

{course_num, dept_name, student_name}

- 不符合BCNF

3. BCNF范式分解

- 当一个关系不符合BCNF：那么分解它.
- 分解的定义：假定关系R 包含属性*A1 ... An.*
  R的分解会分解为两个或者多个关系:
  - 每个新的关系的属性为R属性的子集（不会有属性不属于 R)，并且
  - R 中的每一个属性至少会在一个新的关系中.
- Intuitively, decomposing R means we will store instances of the relation schemas produced by the decomposition, instead of instances of R.

3. BCNF范式分解

- 考虑关系R和函数依赖集 FDs F. 如果F中的函数依赖 X → A 违背 BCNF, 那么：分解R 为R - A 和XA.
- 重复这个思想，我们会得到一个符合BCNF的关系集合.
- 保持无损连接分解。

3. BCNF分解示例

如何将关系模式 **Instr_dept**分解为**BCNF**？

Instr_dept

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

ID→name,salary,dept_name
dept_name→buiding,budget

候选码是什么？

前面已知：ID为候选码，非BCNF！如何分解？

因dept_name→buiding,budget不满足BCNF。
(且函数依赖的左右方交为空)原模式R分解为：
R1=(dept_name,buiding,budget) 即department
R2=R-(buiding,budget)
= (ID,name,salary,dept_name) 即instructor

Instructor R2

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

F2=?{ID→ID,name,dept_name,salary}
是否BCNF？ID为候选键，是BCNF！

Department **R1**

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

F1=?{dept_name→buiding,budget}
是否BCNF？dept_name为候选键，是BCNF！

3. BCNF分解示例(续)

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
  Key = $\{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not superkey)
- Decomposition
  - $R_1 = (B, C)$, $R_2 = (A, B)$
  - $F1 = \{B \rightarrow C\}$, $F2 = \{A \rightarrow B\}$

候选键？
是BCNF？

R1,R1为BCNF？ R1键为B为BCNF！ R2键为A为BCNF！

5）上述介绍的到BCNF的一般分解方法有无不足？

只是强调保证无损分解，但对保持依赖只字未提！

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $L \rightarrow K\}$
  Two candidate keys = JK and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve $JK \rightarrow L$
  This implies that testing for $JK \rightarrow L$ requires a join (between its two sub-relations)

候选键？
是BCNF？

未必总能保持函数依赖！

由于L→K不满足BNCF
将R分解为：
R1=(L,K), F1={L→K}
R2=(J,L), F2=Φ

R1,R2为BCNF？ R1键为L，为BCNF！ R2键为JL，为BCNF！

3. BCNF范式分解

- 通常情况，有多个函数依赖违背BCNF. 那么，我们处理它们的顺序的不同会导致分解的结果不同。
- 会分解为不同的都满足BCNF的关系集。

BCNF 和 依赖保持

- 分解得到的关系，符合BCNF，但可能不满足保持函数依赖.
- 例子：关系CSZ（city, street_name, zip_code）函数依赖 FDs:

  (city, street_name) → zip_code

  zip_code → city
- 要保持*CS → Z*，*则不能分解*，但是CSZ 不符合BCNF.

3. BCNF范式分解

- 如果一个关系符合BCNF，它一般不会出现冗余. 因此，确保所有关系都符合BCNF是很好的一种解决冗余的方式.

- 如何一个关系不符合BCNF，我们可以把它分解为符合BCNF的一个关系集合.
  - 将一个关系分解为符合BCNF的关系集合:
    - 分解是无损分解
    - 可能会保持函数依赖.

4. 3NF

- 有一些情况
  - BCNF 不能保持函数依赖,
  - 在更新上有效的检查函数依赖是否违背是非常重要的。
- 解决方法：定义一个弱的关系，叫做第三范式(3NF)
  - 允许存在一些冗余
  - 函数依赖是否保持可以在单独的关系上检查，而不需要进行连接计算.
  - 3NF一般是保持无损连接分解和保持函数依赖。

4. 3NF

- R符合BCNF, 那么R符合3NF.

- 如果 R 符合 3NF, 可能会存在一定的冗余.

  它是分解和性能的一种折中。 used when BCNF not achievable (e.g., no "good" decomposition, or performance considerations).

  - *将R分解为满足3NF的关系集合，是保持了无损连接分解和保持函数依赖的。*

4. 3NF

- 关系R ，函数依赖集FDs *F 符合3NF，当且仅当F中每一个函数依赖* X → A (X ⊆ R and A ⊆ R)， *下符合列描述中的一个:*
  - A ⊆ X (平凡的FD)， or
  - X 是超键， or
  - A 是R的键的一部分

If one of these two is satisfied for ALL FDs, then R is in BCNF

- 第三范式（3NF）
  - 满足2NF
  - 非主属性直接依赖于候选键，不存在传递依赖（非主属性间的依赖）

学生表（学号，姓名，年龄，学院名称，学院电话）
- 主键/候选键：学号
- 因为存在依赖：学号->学院名称，学院名称->学院电话
- 符合第二范式，但不符合第三范式

5. 3NF分解

3NF的检查方法

3NF关系模式如何判定，如何分解？

如何判定一个关系模式为3NF？

同BCNF检查方法

对BCNF检查方法作补充修改即可得到！

- Optimization: Need to check only FDs in F, need not check all FDs in F^+. 仅需检查F中的每个函数依赖 α → β
- Use attribute closure to check for each dependency $\alpha \rightarrow \beta$, if α is a superkey.
- If α is not a superkey, we have to verify if each attribute in β is contained in a candidate key of R 要清楚R的所有候选键

补充:检查主属性

注：
- this test is rather相当地 more expensive, since it involve finding candidate keys
- testing for 3NF has been shown to be NP-hard
- Interestingly, decomposition into third normal form (described shortly) can be done in polynomial time

计算候选键:代价高

检查3NF:NP难度

分解到3NF:多项式时间

R=(A,B,C)
F={A→B
B→C}

候选键？是3NF？

为：A
不是3NF！

R=(J,K,L)
F={JK→L
L→K}

候选键？是3NF？

为:JK 和 JL
因K为主属性的一部分
是3NF！

5. 3NF分解

- 显然，分解为BCNF的无损连接分解的算法能够用来获取无损连接分解的3NF.

- 为了保持函数依赖，一个思想是:
  - 如果 X → Y 不保持，增加关系XY.
  - 问题是XY可能会违背 3NF!

- 细化：不考虑 FDs F，而是使用*F*的最小的函数依赖集（正则覆盖）.

正则覆盖（最小函数依赖集）

如果去除函数依赖中的一个属性不改变该函数依赖集的闭包，则称该属性是**无关的**(extraneous)。**无关属性**(extraneous attribute)的形式化定义如下：考虑函数依赖集 F 及 F 中的函数依赖 $\alpha \rightarrow \beta$。

- 如果 $A \in \alpha$ 并且 F 逻辑蕴含 $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$，则属性 A 在 α 中是无关的。
- 如果 $A \in \beta$ 并且函数依赖集 $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ 逻辑蕴含 F，则属性 A 在 β 中是无关的。

F 的**正则覆盖**(canonical cover) F_c 是一个依赖集，使得 F 逻辑蕴含 F_c 中的所有依赖，并且 F_c 逻辑蕴涵 F 中的所有依赖。此外，F_c 必须具有如下性质：

- F_c 中任何函数依赖都不含无关属性。
- F_c 中函数依赖的左半部都是唯一的。即，F_c 中不存在两个依赖 $\alpha_1 \rightarrow \beta_1$ 和 $\alpha_2 \rightarrow \beta_2$，满足 $\alpha_1 = \alpha_2$。

```
F_c = F
    repeat
        使用合并律将 F_c 中所有形如 α1→β1 和 α1→β2 的依赖
            替换为 α1→β1β2
        在 F_c 中寻找一个函数依赖 α→β，它在 α 或在 β 中
            具有一个无关属性
            /* 注意，使用 F_c 而非 F 检验无关属性 */
        如果找到一个无关属性，则将它从 F_c 中的 α→β 中删除
    until (F_c 不变)
```

令 F_c 为 F 的正则覆盖；　通常需要根据F计算Fc
$i := 0$;
for each F_c 中的函数依赖 $\alpha \rightarrow \beta$
　$i := i + 1$;
　$R_i := \alpha\beta$;　Fc的每个依赖做成一子模式Ri
if 模式 R_j, $j = 1, 2, \ldots, i$ 都不包含 R 的候选码
then
　$i := i + 1$;
　$R_i := R$ 的任意候选码；+候选键做成子模式
/*（可选）移除冗余关系*/
repeat
　if 模式 R_j 包含于另一个模式 R_k 中
　then
　/* 删除 R_j */
　$R_j := R_i$;
　$i := i - 1$;　Rj 包含Ri时，可去掉Ri
until 不再有可以删除的 R_j
return($R_1, R_2, \cdots, R_i$)　最终的一组Ri时为所求

图 8-12　到 3NF 的保持依赖且无损的分解

“双保持”

3NF分解算法 p. 197

2)任何模式都能分解为一组3NF,保持依赖和无损分解？

设有关系模式:R=(A, B, C, G, H, I)
函数依赖集F={$A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H$ }
R如何“双保持”分解为一组3NF？

第1步: 求F的正则覆盖 (前面已介绍, 用合并律)
Fc={$A \rightarrow BC$
$CG \rightarrow HI$
$B \rightarrow H$ }

第2步：求R的3NF分解。

$A \rightarrow BC$	R1=(A,B,C)	F1={$A \rightarrow BC$}
$CG \rightarrow HI$	R2=(CGHI)	F2={$CG \rightarrow HI$}
$B \rightarrow H$	R3=(BH)	F3={$B \rightarrow H$}

第3步: 确定是否添加R候选键做成的子模式
仅一个候选键AG
因为：(AG)+=AG BCH I
而：　A+=A BCH，　G+=G
因该候选键未包含在R1，R2，R3中
需要产生子模式R4=(AG)　　F4=Φ

注意：还需求出各Ri的函数依赖Fi(限定/投影)　故R保持依赖和无损分解为一组3NF：R1,R2,R3,R4

3. 3NF分解算法的其它例子

3) 如何将关系模式R"双保持"分解为一组3NF?

- Relation schema:

R *cust_banker_branch* = (*customer_id, employee_id, branch_name, type*)

- The functional dependencies F for this relation schema are:

F
1. *customer_id, employee_id* → *branch_name, type*
2. *employee_id* → *branch_name*
3. *customer_id, branch_name* → *employee_id*

- 1) We first compute a canonical cover F_C
  - *branch_name* is extraneous in the r.h.s. of the 1st dependency
  - No other attribute is extraneous, so we get F_C =

Fc
customer_id, employee_id → *type* ← F1
employee_id → *branch_name* ← F2
customer_id, branch_name → *employee_id* ← F3

(等价说明)

即需说明(*customer_id, employee_id*)$^+_{Fc}$包含*branch_name*

- 2) The **for** loop generates following 3NF schema:

R1=(*customer_id, employee_id, type*)
R2=(*employee_id, branch_name*)
R3=(*customer_id, branch_name, employee_id*)

- 3) **Observe** that (*customer_id, employee_id, type*) contains a candidate key of the original schema, so no further relation schema needs be added
- 4) At end of for loop, detect and delete schemas, such as (*employee_id, branch_name*), which are subsets of other schemas
  - result will not depend on the order in which FDs are considered
- The resultant simplified 3NF schema is:

R1=(*customer_id, employee_id, type*)
R3=(*customer_id, branch_name, employee_id*)

(注：删除了R2，因R3包含了R2)

故R保持依赖和无损分解为一组3NF：R1, R3

比较前面BCNF分解（碰巧：
这种分解既是到BCNF的无损分解
又是到3NF的保持依赖和无损分解

3NF分解算法的其它例子(续)

如何将关系模式Instr_dept分解为3NF?

Instr_dept

R

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

F { ID→name,salary,dept_name
dept_name→buiding,budget

前面已知：ID候选码，如何“双保持”分解为一组3NF？

Instructor R2

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

F2= {ID→ID,name,dept_name,salary}
3NF　　ID为候选键！

Department R1

dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

F1={dept_name→buiding,budget}
3NF　　dept_name为候选键！

1）易知F本身已是正则覆盖，Fc=F。
2）故原模式R分解为：
R1=(dept_name,buiding,budget)　即department
R2= (ID,name,salary,dept_name)　即instructor

3）因候选键ID已包含在R2中，不需要单独形成子模式

BCNF和3NF比较

- There is some **redundancy** in this schema
- Example of problems due to redundancy in 3NF
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L;\ L \rightarrow K\}$

Key? JK和JL

R是3NF? *Yes*

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
null	l_2	k_2

- **Repetition**重复 of information
  (e.g., the relationship l_1, k_1)
- need to use **null** values
  (e.g., to represent表示 the relationship
  l_2, k_2 where there is no corresponding value for J).

存在某种数据冗余现象！

讨论3．BCNF分解与3NF分解各自的特点？

1) 该关系模式R有何不足，需要进一步分解吗？

比较前面BCNF分解（见前面PPT8）
1）若分解无损分解为BCNF，
　　可能不能保持函数依赖；
2）若为保持函数依赖而不分解，
　　可能存在数据冗余现象。
可见，各有利弊，需因应用需求定！

注：数据库优化的一般过程：（如果丢失重要函数依赖）（如果某些多表连接查询非常重要）

分解到BCNF→返回到3NF→甚至返回到2NF(甚至1NF)

BCNF 和 3NF的比较

- 将一个关系分解为符合3NF的关系集合:
  - 分解是无损的
  - 分解是保持函数依赖的
- 将一个关系分解为符合BCNF的关系集合:
  - 分解是无损的
  - 可能不保持函数依赖.

设计目标

- 关系型数据库设计的目标:
  - BCNF.
  - 无损连接.
  - 保持函数依赖.
- 如果我们不能得到这些，那么
  - 缺失函数依赖的保持
  - 使用3NF产生冗余

Example

- StorehouseManage(仓库ID, 存储物品ID, 管理员ID, 数量)。

F:(仓库ID, 存储物品ID) →(管理员ID, 数量)
(管理员ID, 存储物品ID) → (仓库ID, 数量)
(仓库ID) → (管理员ID)
(管理员ID) → (仓库ID)

- 是否符合BCNF?
- 是否符合3NF?

Example

- 不符合BCNF
- 分解：
  - 仓库管理：StorehouseManage(仓库ID, 管理员ID)
    仓库：Storehouse(仓库ID, 存储物品ID, 数量)
- 符合3NF

example

- R(**employee_id**,**date**,turnover, department_name,department_manager)
- 统计每个职工的日营业额，记录职工所在部门和经理是谁
- 找出基本函数依赖
- 是否是3NF？不是则分解。

example

- **employee_id,date->**turnover
- department_name->department_manager
- **employee_id->**department_name, department_manager

Example

- R(A, B, C, D, E),
- F = {A → B, C → D} [Same as previous]
- R is in BCNF? 3NF?
- Not in BCNF. This is the case where all attributes in the FDs appear in R. We consider A, and C to see if either is a superkey or not. Obviously, neither A nor C is a superkey, and hence R is not in BCNF. More precisely, we have A → B is in F+ and non-trivial, but A is not a superkey of R.

- We already know that it's not in BCNF. Not in 3NF either. We have A → B is in F+ and non-trivial, but A is not a superkey of R. Furthermore, B is not in any candidate key (since the only candidate key is ACE).

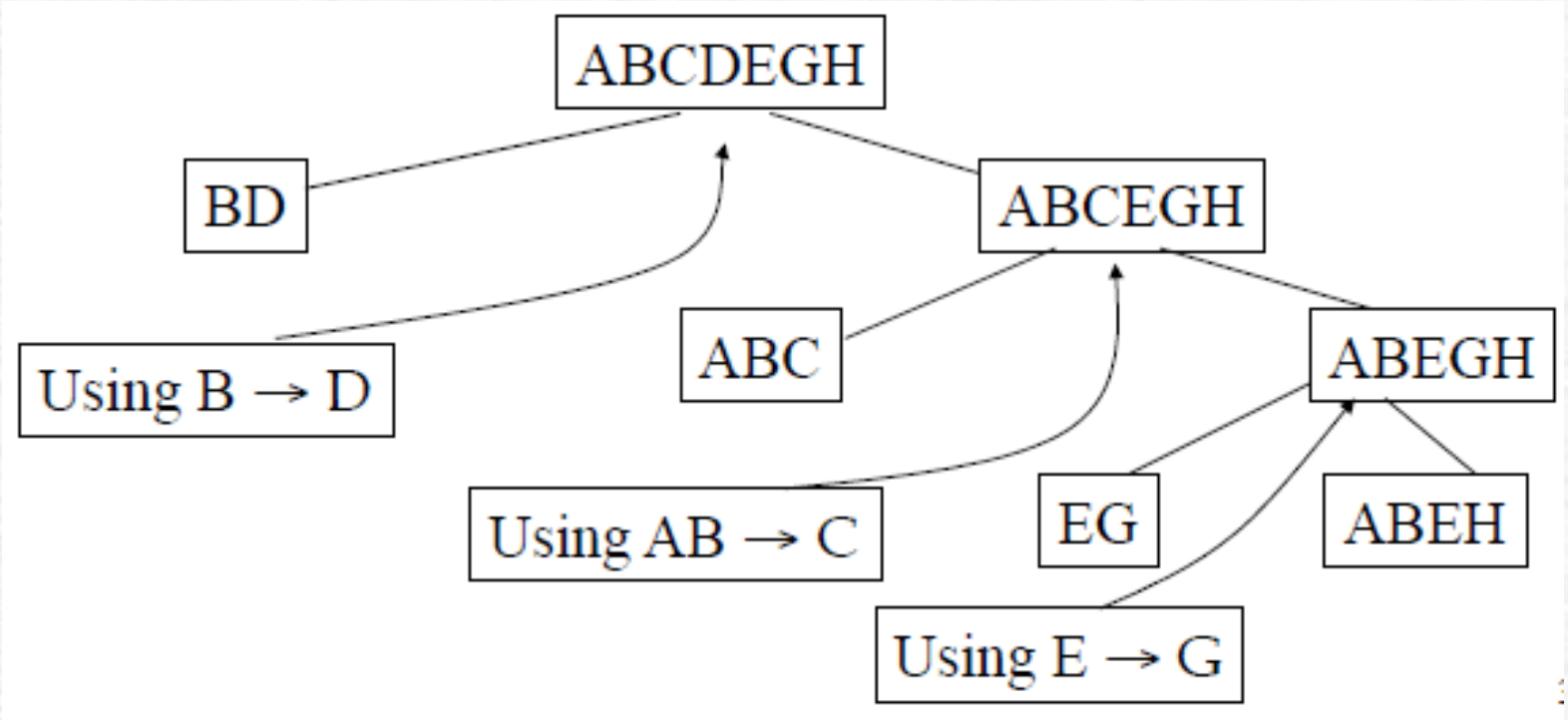
Example

- R(A, B, C, D, E, G, H)
- F={AB → C, AC → B, B → D, BC → A, E → G}
- Candidate keys?
  - H has to be in all candidate keys
  - E has to be in all candidate keys
  - G cannot be in any candidate key (since E is in all candidate keys already).
  - Since AB → C, AC → B and BC → A, we know no candidate key can have ABC together.
  - AEH, BEH, CEH are not superkeys.
  - Try ABEH, ACEH, BCEH. They are all superkeys. And we know they are all candidate keys (since above properties)
  - These are the only candidate keys: (1) each candidate key either contains A, or B, or C since no attributes other than A,B,C determine A, B, C, and (2) if a candidate key contains A, then it must contain either B, or C, and so on.

Example

- Same as previous
- Not in BCNF, not in 3NF, why?
- Decomposition:

R(A, B, C, D, E, G, H)
F={AB → C, AC → B, B → D, BC → A, E → G}



Example

- R(A, B, C, D, E, G, H)
  F={AB → C, AC → B, B → D, BC → A, E → G}
- Decomposition: BD, ABC, EG, ABEH
- Why good decomposition?
  - They are all in BCNF
  - Lossless-join decomposition
  - All dependencies are preserved.

Example

- R(A, B, D, E) decomposed into R1(A, B, D), R2 (A, B, E)
- F={AB → DE}
- It is a dependency preserving decomposition?
  - AB → D can be checked in R1
  - AB → E can be checked in R2
  - {AB → DE} is equivalent to {AB → D, AB → E}

设计关系r(R)模式为：

R=(职工名,项目名,工资,部门号,部门经理)

F:{ 项目名→部门号　　部门号→部门经理　　职工名,项目名→工资 }

1. 求R的候选键

2. 指出使R违反3NF的所有函数依赖

3.将 R保持函数依赖且无损分解为一组3NF

解1：候选键是(职工名,项目名)。因$(职工名,项目名)^{+}=R$

解2：

因：项目名→部门号　　存在非主属性对键的部分函数依赖

　　部门号→部门经理　　存在非主属性对键的传递函数依赖，

故：R不是3NF，甚至不是2NF。

解3：因F不存在无关属性，已是最小覆盖。

故直接将R 保持依赖且无损分解为如下一组3NF：

R1={项目名 , 部门号}，

R2={部门号,部门经理}，

R3={职工名,项目名,工资}，

F1={项目名→部门号}　F2={部门号→部门经理}　F3={职工名,项目名→工资}

注：因键已包含在R3中，故不需单独做成一个新关系模式

练习

- R(A, B, C, D, E, F)
- F={A->C, C->A, B->AC, D->AC}
- 找出R的键
- 是否符合3NF，如果是，说明原因，如果否，分解。