DBMS

UNIT - 4

Database Design - SQL

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

Database Design

- Grouping of attributes to form good relation schemas

- Two levels of relational schema
  - logical /conceptual level
  - storage/ implementation level

- Bottom-up design, top-down design

Informal Guidelines

- Quality of relational schema

  1. Semantics of attributes are clear
  2. Reducing redundant information in tuples
  3. Reducing NULL values in tuples
  4. Disallow possibility of generating spurious tuples

1. Semantics of attributes are clear

Guideline 1. Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation.

2. Reducing redundant information in tuples

- Storing joins leads to update anomalies
  (a) Insertion anomalies
  (b) Deletion anomalies
  (c) Modification anomalies

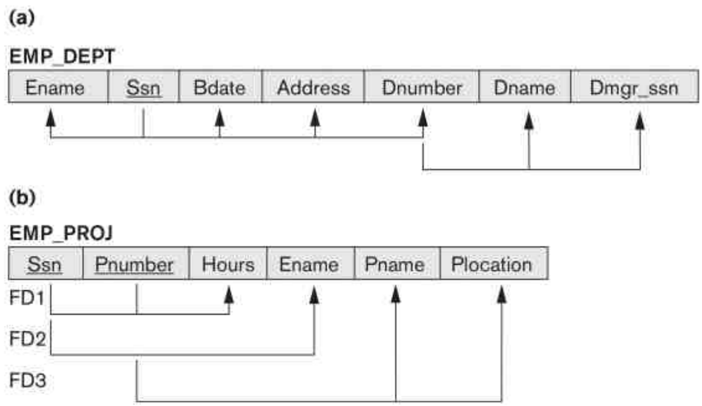


Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.

Guideline 2. Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

3. Reducing NULL values in tuples

- Fat relations: when many attributes grouped together
- If many of the attributes do not apply to all the tuples, there will be many NULL values

Guideline 3. As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.

- Null ratio threshold (eg: 15%)

4. Disallow possibility of generating spurious tuples

Guideline 4. Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

Figure 14.5
Particularly poor design for the EMP_PROJ relation in Figure 14.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 14.4 onto the relations EMP_LOCS and EMP_PROJ1.

(a)

EMP_LOCS

Ename	Plocation

P.K.

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation

P.K.

(b)

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

- If natural join → spurious tuples : lossy join
- Else: lossless join

Formal Guidelines

- Functional dependencies (FDs) specify formal measures of the goodness of relational designs
- Keys are used to define normal forms for the relations
- Constraints derived from meaning and interrelationships of data attributes
- Set of attributes X functionally determines set of attributes Y if the value of X determines a unique value for Y

Definition. A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

$$X \rightarrow Y \not\Rightarrow Y \rightarrow X$$

Eg: In employee table

ssn $\rightarrow$ ename

if X is CK, $X \rightarrow R$ (candidate key)

- Schema designer gives functional dependencies (domain knowledge, data from DB)

$$F = \{FD_1, FD_2, \ldots, FD_N\}$$

Functional Dependencies in Company DB

(1) Project relation

pnumber → pname

(2) Employee

ssn → fname

(3) works-on

ssn, pnumber → hours

FDs May Exist

Figure 14.8
A relation *R*(A, B, C, D) with its extension.

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

$B \to C$ may exist
$C \to B$ may exist

Figure 14.8. Here, the following FDs *may hold* because the four tuples in the current extension have no violation of these constraints: $B \to C$; $C \to B$; $\{A, B\} \to C$; $\{A, B\} \to D$; and $\{C, D\} \to B$. However, the following *do not* hold because we already have violations of them in the given extension: $A \to B$ (tuples 1 and 2 violate this constraint); $B \to A$ (tuples 2 and 3 violate this constraint); $D \to C$ (tuples 3 and 4 violate it).

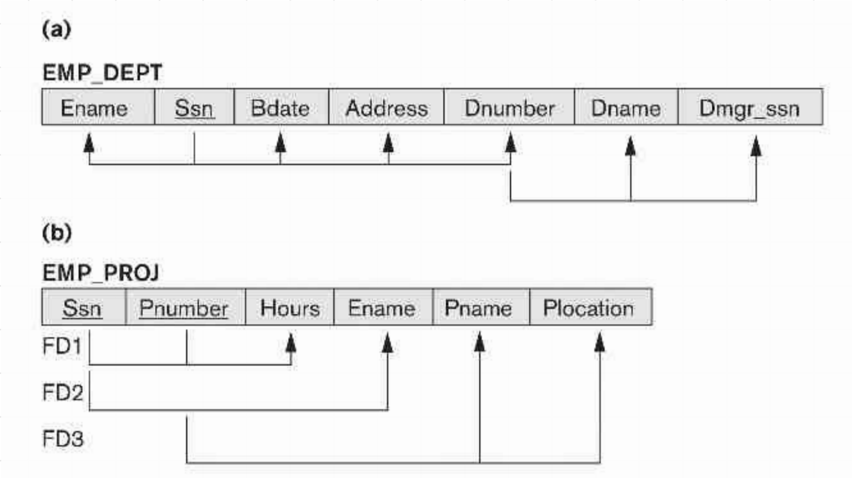
TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure 14.7
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE, TEXT → TEACHER and COURSE → TEXT are ruled out.

FDs Depicted in Schema

$X \rightarrow Y$: X is LHS , Y is RHS



Normal Forms

- Prime attribute: member of candidate keys

FIRST NORMAL FORM

- Domain attributes must be atomic

- Disallow composite, multivalued attributes
- Disallow nested relations

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

- Best solution: separate table DEPT_LOCATIONS with no redundancy

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 14.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

SECOND NORMAL FORM

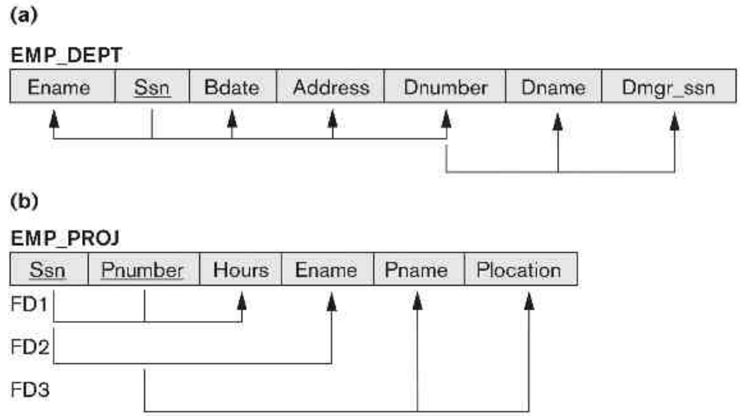
- Full functional dependency

$$Y \rightarrow Z$$

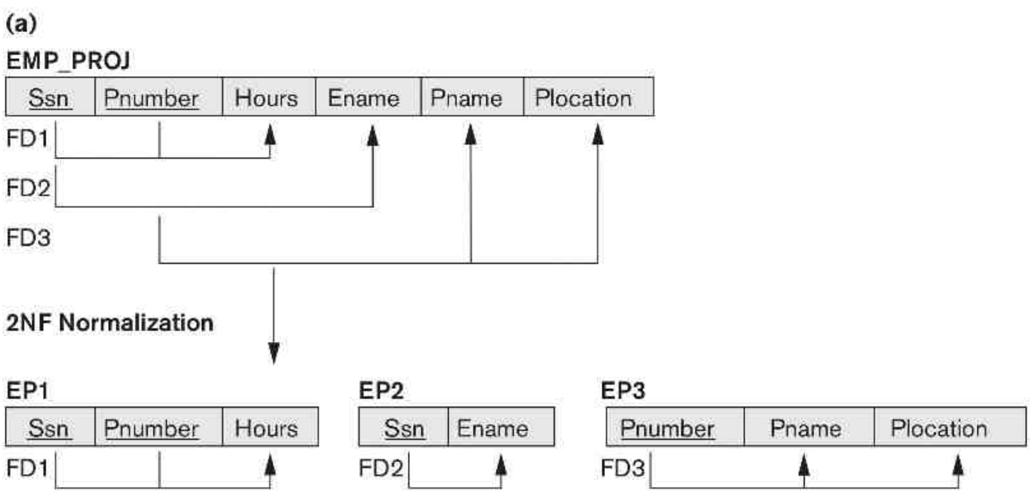
$$[Y - A] \not\rightarrow Z$$

- Eg: Figure 14.3(b), {Ssn, Pnumber} → Hours is a full dependency (neither Ssn → Hours nor Pnumber → Hours holds). However, the dependency {Ssn, Pnumber} → Ename is partial because Ssn → Ename holds.

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



- Every nonprime attribute A in R is fully functionally dependent on PK of R



2NF **Definition.** A relation schema R is in 2NF if every nonprime attribute A in R is *fully functionally dependent* on the primary key of R.

every member of PK

THIRD NORMAL FORM

- No transitive dependencies unless A_2 is a candidate key

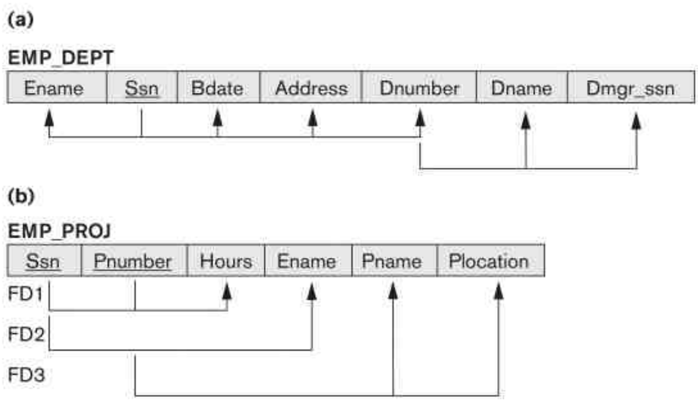
$$A_1 \to A_2 \quad \text{and} \quad A_2 \to A_3$$

$$A_1 \to A_3$$

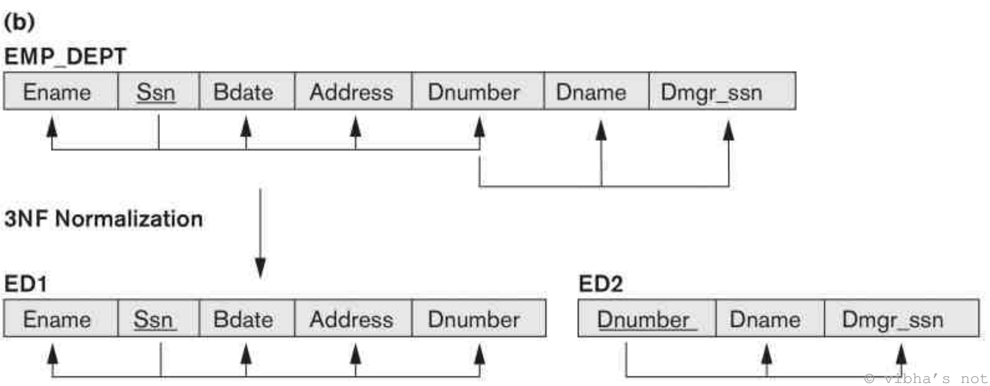
- Eg: SSN → DMGRSSN is transitive

The relation schema EMP_DEPT in Figure 14.3(a) is in 2NF, since no partial dependencies on a key exist. However, EMP_DEPT is not in 3NF because of the transitive dependency of Dmgr_ssn (and also Dname) on Ssn via Dnumber.

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



Convert



3NF

Definition. A relation schema R is in **third normal form (3NF)** if, whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in R, either (a) X is a superkey of R, or (b) A is a prime attribute of R.[13]

Figure 14.12
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.

Table 14.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

BOYCE - CODD NORMAL FORM

- Stricter than 3NF

BCNF **Definition.** A relation schema R is in **BCNF** if whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in R, then X is a superkey of R.

Non-Trivial FD: if $X \rightarrow Y$ is an FD and $X \subseteq Y$, it is a trivial FD. Else, non-trivial FD

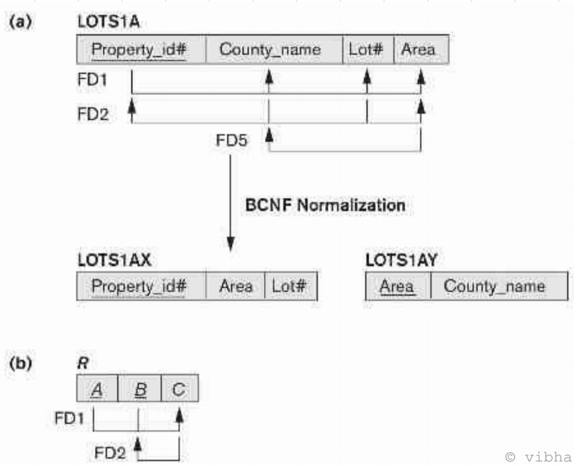


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. $C \rightarrow B$.

Properties of BCNF Normalised Relations

1. Non additive join / lossless join property
   - no spurious tuples upon joining decomposed relations
   - critical

2. Dependency preservation property
   - each FD represented in some individual relation resulting after decomposition
   - desirable but sometimes sacrificed

Eg: Relation in 3NF and not in BCNF

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

Figure 14.14
A relation TEACH that is in 3NF but not BCNF.

{Student, Course} → Instructor
Instructor → Course

prime att

Testing Binary Decompositions for Lossless Join (Non-Additive Join) Property

NJB (Nonadditive Join Test for Binary Decompositions). A decomposition $D = \{R_1, R_2\}$ of R has the lossless (nonadditive) join property with respect to a set of functional dependencies F on R *if and only if* either

- The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in F^{+}[15], or
- The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in F^{+}

[15] The notation F^+ refers to the cover of the set of functional dependencies and includes all f.d.'s implied by F. It is discussed in detail in Section 15.1. Here, it is enough to make sure that one of the two f.d.'s actually holds for the nonadditive decomposition into R_1 and R_2 to pass this test.

3 Possible Decompositions for TEACH

1. R1 (Student, Instructor) and R2(Student, Course) — R1 ∩ R2 : Student
2. R1 (Course, Instructor) and R2(Course, Student) — R1 ∩ R2 : Course
3. R1 (Instructor, Course) and R2(Instructor, Student) — R1 ∩ R2 : Instructor

1. Student → Instructor ✗
   Student → Course ✗

2. Course → Student ✗
   Course → Instructor ✗

3. Instructor → Student ✗
   Instructor → Course ✓
   } good composition

- All 3 lose {student, course} → instructor

Achieving BCNF

Let R be the relation not in BCNF, let $X \subseteq R$, and let $X \rightarrow A$ be the FD that causes a violation of BCNF. R may be decomposed into two relations:

(i) R − A
(ii) X ∪ A

If either $R - A$ or XA. is not in BCNF, repeat the process.

In prev eg, FD that violated BCNF:

instructor $\rightarrow$ course

BCNF decomposition

i) student, instructor and instructor, course

MULTIVALUED DEPENDENCY

- $X \twoheadrightarrow Y$, r on R and two tuples t_1 and t_2 exist in r
- If $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 must also exist in r such that (we use $Z = (R - (X \cup Y))$)

1. $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
2. $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$
3. $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$

Note: t_1, t_2, t_3, t_4 may not be distinct

- An MVD $X \twoheadrightarrow Y$ is called trivial if
(a) $Y \subseteq X$ or
(b) $X \cup Y = R$
- $X \twoheadrightarrow Y$: X multidetermines Y
- $X \twoheadrightarrow Y \Rightarrow X \twoheadrightarrow Z$ and is also written as $X \twoheadrightarrow Y | Z$

MVD

(a) EMP

	Ename	Pname	Dname
1	Smith	X	John
2	Smith	Y	Anna
3	Smith	X	Anna
4	Smith	Y	John

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Figure 14.15
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS. → trivial
(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

Ename $\twoheadrightarrow$ Pname is trivial

(a) Ename $\twoheadrightarrow$ Pname

X = Ename
Y = Pname

Z = R − (X ∪ Y) = Dname

Let $t_1 = 1$, $t_2 = 2$, $t_3 = 3$, $t_4 = 4$

(i) $t_3[X] = t_4[X] = t_1[X] = t_2[X]$ ✓

(ii) $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$ ✓

(iii) $t_3[Z] = t_2[Z]$ and $t_1[Z] = t_4[Z]$ ✓

FOURTH NORMAL FORM

- Relations containing MVD tend to be all-key relations

Definition. A relation schema R is in 4**NF** with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every *nontrivial* multivalued dependency $X \twoheadrightarrow Y$ in F^+,[21] X is a superkey for R.

- Points

  - An all-key relation is always in BCNF since it has no FDs.
  - An all-key relation such as the EMP relation in Figure 14.15(a), which has no FDs but has the MVD Ename $\twoheadrightarrow$ Pname | Dname, is not in 4NF.
  - A relation that is not in 4NF due to a nontrivial MVD must be decomposed to convert it into a set of relations in 4NF.
  - The decomposition removes the redundancy caused by the MVD.

FIFTH NORMAL FORM

Join Dependency

Definition. A **join dependency** (JD), denoted by JD(R_1, R_2, … , R_n), specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into R_1, R_2, … , R_n. Hence, for every such r we have

$$* \, (\pi_{R_1}(r), \pi_{R_2}(r), \ldots, \pi_{R_n}(r)) = r$$

- MVD is a special case of a JD where $n=2$
- Trivial JD: if one of the schemas R_i in JD(R_1, R_2, …, R_n) is equal to R

Definition. A relation schema R is in **fifth normal form (5NF)** (or **project-join normal form (PJNF)**) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R_1, R_2, … , R_n) in F^+ (that is, implied by F),[22] every R_i is a superkey of R.

[22]Again, F^+ refers to the cover of functional dependencies F, or all dependencies that are implied by F. This is defined in Section 15.1.

Inference Rules

Definition: An FD $X \to Y$ is **inferred from** or **implied by** a set of dependencies F specified on R if $X \to Y$ holds in *every* legal relation state r of R; that is, whenever r satisfies all the dependencies in F, $X \to Y$ also holds in r.

Armstrong's Axioms

IR1 (reflexive rule)[2]: If $X \supseteq Y$, then $X \to Y$.

IR2 (augmentation rule)[3]: $\{X \to Y\} \models XZ \to YZ$.

IR3 (transitive rule): $\{X \to Y, Y \to Z\} \models X \to Z$.

Proofs

Proof of IR1. Suppose that $X \supseteq Y$ and that two tuples t_1 and t_2 exist in some relation instance r of R such that $t_1\,[X] = t_2\,[X]$. Then $t_1[Y] = t_2[Y]$ because $X \supseteq Y$; hence, $X \to$ Y must hold in r.

Proof of IR2 (by contradiction). Assume that $X \to Y$ holds in a relation instance r of R but that $XZ \to YZ$ does not hold. Then there must exist two tuples t_1 and t_2 in r such that (1) $t_1\,[X] = t_2\,[X]$, (2) $t_1\,[Y] = t_2\,[Y]$, (3) $t_1\,[XZ] = t_2\,[XZ]$, and (4) $t_1\,[YZ] \neq t_2\,[YZ]$. This is not possible because from (1) and (3) we deduce (5) $t_1\,[Z] = t_2\,[Z]$, and from (2) and (5) we deduce (6) $t_1\,[YZ] = t_2\,[YZ]$, contradicting (4).

Proof of IR3. Assume that (1) $X \to Y$ and (2) $Y \to Z$ both hold in a relation r. Then for any two tuples t_1 and t_2 in r such that $t_1\,[X] = t_2\,[X]$, we must have (3) $t_1\,[Y] = t_2\,[Y]$, from assumption (1); hence we must also have (4) $t_1\,[Z] = t_2\,[Z]$ from (3) and assumption (2); thus $X \to Z$ must hold in r.

Additional Rules

IR4 (decomposition, or projective, rule): $\{X \to YZ\} \models X \to Y$.

IR5 (union, or additive, rule): $\{X \to Y, X \to Z\} \models X \to YZ$.

IR6 (pseudotransitive rule): $\{X \to Y, WY \to Z\} \models WX \to Z$.

CLOSURE of F

Definition. Formally, the set of all dependencies that include F as well as all dependencies that can be inferred from F is called the **closure** of F; it is denoted by F^+.

The closure F^+ of F is the set of all functional dependencies that can be inferred from F. To determine a systematic way to infer dependencies, we must discover a set of **inference rules** that can be used to infer new dependencies from a given set of dependencies. We consider some of these inference rules next. We use the notation $F \models X \rightarrow Y$ to denote that the functional dependency $X \rightarrow Y$ is inferred from the set of functional dependencies F.

CLOSURE of X

Definition. For each such set of attributes X, we determine the set X^+ of attributes that are functionally determined by X based on F; X^+ is called the **closure of X under F**.

Algorithm to Calculate X^+

Algorithm 15.1. Determining X^+, the Closure of X under F

Input: A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

$X^+ := X$;
repeat
    old$X^+ := X^+$;
    for each functional dependency $Y \rightarrow Z$ in F do
        if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
    until (X^+ = oldX^+);

Example

Schema

```
CLASS (Classid, Course#, Instr_name, Credit_hrs, Text,
Publisher, Classroom, Capacity).
```

Let F, the set of functional dependencies for the above relation include the following f.d.s:

FD1: Class id $\rightarrow$ Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity;

FD2: Course# $\rightarrow$ Credit_hrs;

FD3: {Course#, Instr_name} $\rightarrow$ Text, Classroom;

FD4: Text $\rightarrow$ Publisher

FD5: Classroom $\rightarrow$ Capacity

{ Classid }$^+$ = { Classid , Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity } = CLASS

{ Course#}$^+$ = { Course#, Credit_hrs}

{ Course#, Instr_name }$^+$ = { Course#, Credit_hrs, Text, Publisher, Classroom, Capacity }

Equivalence of Sets of FDs

- 2 sets of FDs are equivalent if
  - G covers F (every FD in F can be inferred from G)
  - F covers G (every FD in G can be inferred from F)
  - $\therefore$ if $F^+ = G^+$

Q: $R(A, C, D, E, H)$

$F = \{ A \to C,\ AC \to D,\ E \to AD,\ E \to H \}$

$G = \{ A \to CD,\ E \to AH \}$

Are F & G equivalent?

F covers G

1. Using G, compute A^+ and E^+
2. Using F, compute A^+ and E^+

1. $A^+ = \{ACD\}$

   $E^+ = \{EAHCD\}$

2. $A^+ = \{ACD\}$

   $E^+ = \{EADHC\}$

G covers F

1. Using F, compute A^+, AC^+, E^+
2. Using G, compute A^+, AC^+, E^+

1. $A^+ = \{ACD\}$

   $AC^+ = \{ACD\}$

   $E^+ = \{EADHC\}$

2. $A^+ = \{ACD\}$

   $AC^+ = \{ACD\}$

   $E^+ = \{EAHCD\}$

$\therefore$ $F^+ = G^+$ and F & G are equivalent

Algorithm to Determine key of a Relation

Algorithm 15.2(a). Finding a Key K for R Given a Set F of Functional Dependencies

Input: A relation R and a set of functional dependencies F on the attributes of R.

1. Set $K := R$.
2. For each attribute A in K
   {compute $(K - A)^+$ with respect to F;
   if $(K - A)^+$ contains all the attributes in R, then set $K := K - \{A\}$ };

Procedure to Find the candidate key for the relation R and FD set F

1. Find attributes that are neither on the LHS nor on the RHS of any FD

2. Find the attributes that are only on the RHS of the FDs

3. Find the attributes that are only on the LHS of the FDs

4. Combine attributes from 1 & 3 and test for closure. If closure gives all attributes, 1 & 3 combined is CK

5. Else, combine attributes not in 2 & ones in 4 to get different combinations of CKs and test for closure

Q: $R(A, B, C, D, E)$

$F = \{AB \to CD, E \to A, D \to A\}$

Identify CK

1. ϕ

2. Only in RHS = $\{C\}$

3. Only in LHS = $\{BE\}$

4. ① & ③ = $\{BE\}$

$BE^+ = \{BEACD\} \longrightarrow$ all attributes

CK = BE

Q: $R(A, B, C, D, E)$

$F = \{A \to B, BC \to E, ED \to A\}$

Find CK

1. ϕ

2. Only RHS = ϕ

3. Only LHS = $\{CD\}$

4. ① & ③ = $\{CD\}$

$CD^+ = \{CD\} \longrightarrow$ not a key

5. Combine not in ② & ④

try $ACD^+ = \{ACDBE\}$

$BCD^+ = \{BCDEA\}$

$ECD^+ = \{ECDAB\}$

all 3 are CKs

Q: R(A, B, C, D, E, F, G)

$F = \{AB \to F,\ AD \to E,\ F \to G\}$

Identify CK

1. $\{C\}$

2. Only RHS = $\{GE\}$

3. Only LHS = $\{ABD\}$

4. ① & ③ = $\{ABCD\}$

$ABCD^+ = \{ABCDFEG\}$ → all attr

$\therefore$ CK = ABCD

Minimal Cover of F

- Extraneous attributes

Definition. A **minimal cover** of a set of functional dependencies E is a minimal set of dependencies (in the standard canonical form[5] and without redundancy) that is equivalent to E. We can always find *at least one* minimal cover F for any set of dependencies E using Algorithm 15.2.

[5]It is possible to use the inference rule IR5 and combine the FDs with the same left-hand side into a single FD in the minimum cover in a nonstandard form. The resulting set is still a minimum cover, as illustrated in the example.

- Conditions for minimal form

1. Every dependency in F has a single attribute for its right-hand side.

2. We cannot replace any dependency X → A in F with a dependency Y → A, where Y is a proper subset of X, and still have a set of dependencies that is equivalent to F.

3. We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F.

Algorithm (Irreducible Equivalent)

Algorithm 15.2. Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

Note: Explanatory comments are given at the end of some of the steps. They follow the format: (*comment*).

1. Set $F := E$.
2. Replace each functional dependency $X \to \{A_1, A_2, \ldots, A_n\}$ in F by the n functional dependencies $X \to A_1, X \to A_2, \ldots, X \to A_n$. (*This places the FDs in a canonical form for subsequent testing*)
3. For each functional dependency $X \to A$ in F

   for each attribute B that is an element of X

   if $\{ \{F - \{X \to A\} \} \cup \{ (X - \{B\}) \to A\} \}$ is equivalent to F

   then replace $X \to A$ with $(X - \{B\}) \to A$ in F.

   (*This constitutes removal of an extraneous attribute B contained in the left-hand side X of a functional dependency $X \to A$ when possible*)
4. For each remaining functional dependency $X \to A$ in F

   if $\{F - \{X \to A\} \}$ is equivalent to F,

   then remove $X \to A$ from F. (*This constitutes removal of a redundant functional dependency $X \to A$ from F when possible*)

Q: $E = \{B \to A,\ D \to A,\ AB \to D\}$

1. Already canonical
2. Does $AB \to D$ have extraneous attr?

   $B \to A \Rightarrow B \to AB$ and $AB \to D \Rightarrow B \to D$

   $\therefore AB \to D$ can be replaced with $B \to D$

$E = \{B \to A,\ D \to A,\ B \to D\}$

Q: Find minimal cover of G

$G: \{A \to BCDE,\ CD \to E\}$

1. $G: \{A \to B,\ A \to C,\ A \to D,\ A \to E,\ CD \to E\}$

2. for $CD \to E$

$A \to CD$ and $CD \to E$
$A \to E$ is redundant

$G: \{A \to BCD,\ CD \to E\}$

DESIGNING A SET OF RELATIONS

Goals

1. Lossless join property - must
2. Dependency preservation property
3. Additional Normal Forms

Properties of Relational Decomposition

1. Relational Decomposition and Insufficiency of Normal Forms

1.1 Universal Relation Schema

- Relation schema $R = \{A_1, A_2, \ldots, A_n\}$ that includes all attributes of the DB

1.2 Universal Relation Assumption

- Every attribute name is unique

1.3 Decomposition

- Decomposing universal relation schema R into a set of relation schemas $D = \{R1, R2, \ldots, Rm\}$

- D = relational DB schema = decomposition of R

1.4 Attribute Preservation Condition

- Each attribute R appears in at least one relation schema R_i in the decomposition D of R

- No attributes left out

- Formally, $\bigcup_{i=1}^{m} R_i = R$

- Every relation R_i follows 3NF or BCNF

2. Dependency Preservation Property of Decomposition

Definition. Given a set of dependencies F on R, the **projection** of F on R_i, denoted by $\pi_{R_i}(F)$ where R_i is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ are all contained in R_i. Hence, the projection of F on each relation schema R_i in the decomposition D is the set of functional dependencies in F^+, the closure of F, such that all the left- and right-hand-side attributes of those dependencies are in R_i. We say that a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ of R is **dependency-preserving** with respect to F if the union of the projections of F on each R_i in D is equivalent to F; that is, $((\pi_{R_1}(F)) \cup \text{K} \cup (\pi_{R_m}(\text{F})))^+ = F^+$.

3. Non-Additive or Lossless Join Property

Definition. Formally, a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for *every* relation state r of R that satisfies F, the following holds, where $*$ is the NATURAL JOIN of all the relations in D: $*(\pi_{R_1}(r), \ldots, \pi_{R_m}(r)) = r$.

Testing for Lossless Join Property in n-ary Decomposition

Algorithm 15.3. Testing for Nonadditive Join Property

Input: A universal relation R, a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ of R, and a set F of functional dependencies.

Note: Explanatory comments are given at the end of some of the steps. They follow the format: (**comment**).

1. Create an initial matrix S with one row i for each relation R_i in D, and one column j for each attribute A_j in R.
2. Set $S(i, j) := b_{ij}$ for all matrix entries. (*Each b_{ij} is a distinct symbol associated with indices (i, j)*)
3. For each row i representing relation schema R_i
   {for each column j representing attribute A_j
      {if (relation R_i includes attribute A_j) then set $S(i, j) := a_j$;};}; (*Each a_j is a distinct symbol associated with index (j)*)
4. Repeat the following loop until a *complete loop execution* results in no changes to S
   {for each functional dependency $X \rightarrow Y$ in F
      {for all rows in S *that have the same symbols* in the columns corresponding to attributes in X
         {make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows: If any of the rows has an a symbol for the column, set the other rows to that *same a* symbol in the column. If no a symbol exists for the attribute in any of the rows, choose one of the b symbols that appears in one of the rows for the attribute and set the other rows to that same b symbol in the column ;} ; } ;};
5. If a row is made up entirely of a symbols, then the decomposition has the nonadditive join property; otherwise, it does not.

Figure 15.1
Nonadditive join test for *n*-ary decompositions. (a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test. (b) A decomposition of EMP_PROJ that has the lossless join property. (c) Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test.

(a) R = {Ssn, Ename, Pnumber, Pname, Plocation, Hours} $D = \{R_1, R_2\}$
R_1 = EMP_LOCS = {Ename, Plocation}
R_2 = EMP_PROJ1 = {Ssn, Pnumber, Hours, Pname, Plocation}

F = {Ssn → Ename; Pnumber → {Pname, Plocation}; {Ssn, Pnumber} → Hours}

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(No changes to matrix after applying functional dependencies)

(b)

EMP

Ssn	Ename

PROJECT

Pnumber	Pname	Plocation

WORKS_ON

Ssn	Pnumber	Hours

(c) R = {Ssn, Ename, Pnumber, Pname, Plocation, Hours} $D = \{R_1, R_2, R_3\}$
R_1 = EMP = {Ssn, Ename}
R_2 = PROJ = {Pnumber, Pname, Plocation}
R_3 = WORKS_ON = {Ssn, Pnumber, Hours}

F = {Ssn → Ename; Pnumber → {Pname, Plocation}; {Ssn, Pnumber} → Hours}

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(Original matrix S at start of algorithm)

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}\ a_2$	a_3	$\cancel{b_{34}}\ a_4$	$\cancel{b_{35}}\ a_5$	a_6

(Matrix S after applying the first two functional dependencies; last row is all "a" symbols so we stop)

5. Successive Non-Additive Join Decomposition

Claim 2 (Preservation of Nonadditivity in Successive Decompositions). If a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ of R has the nonadditive (lossless) join property with respect to a set of functional dependencies F on R, and if a decomposition $D_i = \{Q_1, Q_2, \ldots, Q_k\}$ of R_i has the nonadditive join property with respect to the projection of F on R_i, then the decomposition $D_2 = \{R_1, R_2, \ldots, R_{i-1}, Q_1, Q_2, \ldots, Q_k, R_{i+1}, \ldots, R_m\}$ of R has the nonadditive join property with respect to F.

ALGORITHMS FOR RELATIONAL DATABASE SCHEMA DESIGN

1. Relational Synthesis into 3NF

Algorithm 15.4 Relational Synthesis into 3NF with Dependency Preservation and Nonadditive Join Property

Input: A universal relation R and a set of functional dependencies F on the attributes of R.

1. Find a minimal cover G for F (use Algorithm 15.2).
2. For each left-hand-side X of a functional dependency that appears in G, create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$, where $X \to A_1, X \to A_2, \ldots, X \to A_k$ are the only dependencies in G with X as left-hand side (X is the key of this relation).
3. If none of the relation schemas in D contains a key of R, then create one more relation schema in D that contains attributes that form a key of R. (Algorithm 15.2(a) may be used to find a key.)
4. Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation R is considered redundant if R is a projection of another relation S in the schema; alternately, R is subsumed by S.[7]

[7]Note that there is an additional type of dependency: R is a projection of the join of two or more relations in the schema. This type of redundancy is considered join dependency, as we discussed in Section 15.7. Hence, technically, it may continue to exist without disturbing the 3NF status for the schema.

Eg 1:

Example 1 of Algorithm 15.4. Consider the following universal relation:

U (Emp_ssn, Pno, Esal, Ephone, Dno, Pname, Plocation)

Emp_ssn, Esal, and Ephone refer to the Social Security number, salary, and phone number of the employee. Pno, Pname, and Plocation refer to the number, name, and location of the project. Dno is the department number.

The following dependencies are present:

FD1: Emp_ssn → {Esal, Ephone, Dno}

FD2: Pno → { Pname, Plocation}

FD3: Emp_ssn, Pno → {Esal, Ephone, Dno, Pname, Plocation}

By virtue of FD3, the attribute set {Emp_ssn, Pno} represents a key of the universal relation. Hence F, the set of given FDs, includes {Emp_ssn → Esal, Ephone, Dno; Pno → Pname, Plocation; Emp_ssn, Pno → Esal, Ephone, Dno, Pname, Plocation}.

By applying the minimal cover Algorithm 15.2, in step 3 we see that Pno is an extraneous attribute in Emp_ssn, Pno → Esal, Ephone, Dno. Moreover, Emp_ssn is extraneous in Emp_ssn, Pno → Pname, Plocation. Hence the minimal cover consists of FD1 and FD2 only (FD3 being completely redundant) as follows (if we group attributes with the same left-hand side into one FD):

Minimal cover G: {Emp_ssn → Esal, Ephone, Dno; Pno → Pname, Plocation}

The second step of Algorithm 15.4 produces relations R_1 and R_2 as:

R_1 (<u>Emp_ssn</u>, Esal, Ephone, Dno)

R_2 (<u>Pno</u>, Pname, Plocation)

In step 3, we generate a relation corresponding to the key {Emp_ssn, Pno} of U. Hence, the resulting design contains:

R_1 (<u>Emp_ssn</u>, Esal, Ephone, Dno)

R_2 (<u>Pno</u>, Pname, Plocation)

R_3 (<u>Emp_ssn</u>, <u>Pno</u>)

This design achieves both the desirable properties of dependency preservation and nonadditive join.

Q: Consider relation U {P, C, L, A}

FD1: $P \rightarrow \{L, C, A\}$
FD2: $LC \rightarrow \{A, P\}$
FD3: $A \rightarrow \{C\}$

The universal relation with abbreviated attributes is U (P, C, L, A). If we apply the minimal cover Algorithm 15.2 to *F*, (in step 2) we first represent the set *F* as

F: {P → L, P → C, P → A, LC → A, LC → P, A → C}

In the set *F*, P → A can be inferred from P → LC and LC → A; hence P → A by transitivity and is therefore redundant. Thus, one possible minimal cover is

Minimal cover *GX*: {P → LC, LC → AP, A → C}

In step 2 of Algorithm 15.4, we produce design *X* (before removing redundant relations) using the above minimal cover as

Design *X*: R_1 (P, L, C), R_2 (L, C, A, P), and R_3 (A, C)

In step 4 of the algorithm, we find that R_3 is subsumed by R_2 (that is, R_3 is always a projection of R_2 and R_1 is a projection of R_2 as well). Hence both of those relations are redundant. Thus the 3NF schema that achieves both of the desirable properties is (after removing redundant relations)

Design *X*: R_2 (L, C, A, P).

or, in other words it is identical to the relation LOTS1A (Property_id, Lot#, County, Area) that we had determined to be in 3NF in Section 14.4.2.

2. Relational Decomposition into BCNF

Algorithm 15.5. Relational Decomposition into BCNF with Nonadditive Join Property

Input: A universal relation *R* and a set of functional dependencies *F* on the attributes of *R*.

1. Set $D := \{R\}$;
2. While there is a relation schema *Q* in *D* that is not in BCNF do

```
{
    choose a relation schema Q in D that is not in BCNF;
    find a functional dependency X → Y in Q that violates BCNF;
    replace Q in D by two relation schemas (Q – Y) and (X ∪ Y);
};
```

Q: Given relational schema $R(PQRSTUV)$ having
$FD = \{P \rightarrow Q,\ QR \rightarrow ST,\ PTU \rightarrow V\}$

(i) Determine $(QR)^+$
(ii) Determine $(PR)^+$
(iii) Identify a key

(i) $QR^+ = QRST$

(ii) $PR^+ = PRQST$

(check incoming edge)

(iii) Step 1 : ϕ

Step 2: only RHS = SV

Step 3: only LHS = PRU

Step 4: 3 & 1 = PRU

$PRU^+ = PRQSTUV$

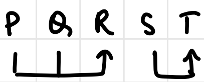
Q: Given $R(PQRST)$

$FD = \{P \rightarrow QR,\ RS \rightarrow T,\ Q \rightarrow S,\ T \rightarrow P\}$

(i) T^+

(i) $T^+ = TPQRS$

Q: Given R(PQRST) and FD = {PQ → R, S → T}. Determine whether R is in 2NF. If not, convert to 2NF.



$PQS^+ = PQSRT$ → key

Prime attributes : PQS
Non-prime attr : RT

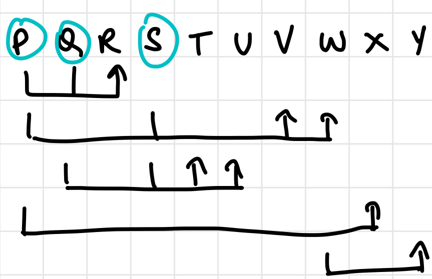
R is dependent on PQ (partial key)
T is dependent on S (partial key)

Decompose
- R1(PQR)
- R2(ST)
- R3(PQS)

Q: R(PQRSTUVWXY),

FD = {PQ → R, PS → VW, QS → TU, P → X, W → Y}

Identify if R in 2NF. If not, convert



$PQS^+ = PQSRVWTUXY$
$= PQRSTUVWXY$

$PQS \rightarrow$ key

- Violation of partial dependency

$PQ \rightarrow R$
$PS \rightarrow VW$
$QS \rightarrow TU$
$P \rightarrow X$

- Decomposition

R1(PQR)
R2(PSVW)
R3(QSTU)
R4(PX)
R5(WY)
R6(PQS)

Q: R(ABCDE) FD = {$A \rightarrow B$, $B \rightarrow E$, $C \rightarrow D$}. 2NF?

A B C D E

$AC^+ = ACBED \longrightarrow$ key

FDs violating

$A \to B$
$C \to D$

Decomposition

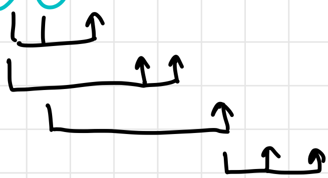
R1(A, B)
R2(C, D)
R3(A, C)
R4(B, E)

Q: R(ABCDEFGH)

FD = $\{AB \to C,\ A \to DE,\ B \to F,\ F \to GH\}$

which NF?

A B C D E F G H



$AB^+ = ABCDEFGH$

(i) BCNF : $X \to A$, x : key

no $\because$ $A \to DE$ & $B \to F$

(ii) 3NF : a) $x \to$ key

b) $A \to$ prime att

no ∵ $A \to DE$ & $B \to F$

(iii) 2NF : non-prime partial dependence key

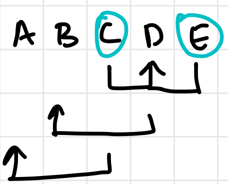
no ∵ partial dependence on key $A \to DE$

$B \to F$

∴ 1NF

Q: R(ABCDE)

$FD = \{CE \to D,\ D \to B,\ C \to A\}$



$CE^+ = CEDBA$

prime: CE

non : ABD

fails 2NF ⇒ 1NF

Q: R(ABCDEF)

FD = {AB → C, DC → AE, E → F}

A B C D E F

$BD^+ = BD$

$ABD^+ = ABDCEF$ →
$CBD^+ = CBDAEF$ → } 2 possible keys

$EBD^+ = EBDF$ ✗

$FBD^+ = FBD$ ✗

AB → C : partial → violates 2NF

Q: R(ABCDEFGHI)

AB → C, BD → EF, AD → GH, A → I

A B C D E F G H I

- $ABD^+ = ABDCEFGHI \rightarrow$ key

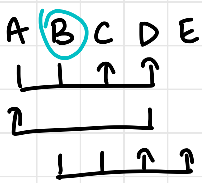
prime: ABD
non: CEFGHI

partial dep $\rightarrow$ violates 2NF

$\therefore$ in 1NF

Q: R(ABCDE)

$AB \rightarrow CD$, $D \rightarrow A$, $BC \rightarrow DE$



$B^+ = B$

$$\left.\begin{aligned} AB^+ &= ABCDE \\ CB^+ &= CBDEA \\ DB^+ &= DBACE \end{aligned}\right\} \text{AB, CB, DB keys}$$

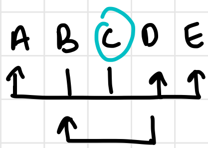
$EB^+ = EB$

(i) BCNF : $D \to A$ fails

(ii) 3NF : yes

Q: $R(ABCDE)$

$BC \to ADE, \quad D \to B$



$C^+ = C \longrightarrow$ no

$$\left.\begin{aligned} BC^+ &= BCADE \\ DC^+ &= DCBAE \end{aligned}\right\} \longrightarrow \text{key}$$

(i) BCNF

no, fails $D \to B$

(ii) 3NF

yes

All key Relations

- All attrs are keys
- Always BCNF
- Eg: $R(A,B,C)$, $FD = \{A \to B,\ B \to C,\ C \to A\}$

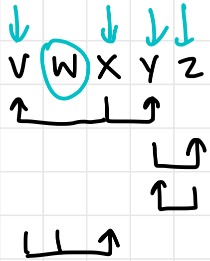
  keys = A, B, C

BCNF — 2 Attribute Relations

- $R(A,B)$ has only 2 possible FDs: $A \to B$ and $B \to A$
- Always in BCNF

Q: $R(VWXYZ)$

$X \to YV,\ Y \to Z,\ Z \to Y,\ VW \to X$



$W^+ = W$

$$\left.\begin{array}{l} VW^+ = VWXYZ \\ XW^+ = XWYVZ \end{array}\right\} \text{keys}$$

$YW^+ = YWZ$

$ZW^+ = ZWY$

prime: VWX   key: VW, XW
non: YZ

(i) BCNF : fails

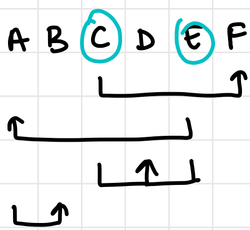
(ii) 3NF : fails $X \rightarrow YV$

(ii) 2NF : fails $X \rightarrow YV$

(iii) 1NF : passes

Q: R(ABCDEF)

$C \rightarrow F$, $E \rightarrow A$, $EC \rightarrow D$, $A \rightarrow B$

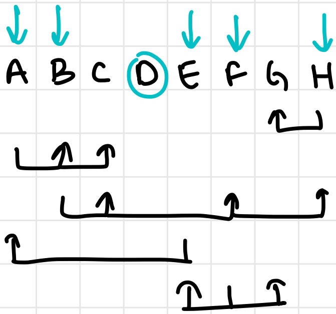


$EC^+ = ECDABF$ key

$\therefore$ 1NF ($C \rightarrow F$, $E \rightarrow A$ violates 2NF)

Q: R(ABCDEFGH)

$CH \to G$, $A \to BC$, $B \to CFH$, $E \to A$, $F \to EG$



$D^+ = D$

$DA^+ = ABCFHGED$

$DB^+ = DBCFHGEA$

$DE^+ = DEABCFHG$

$DF^+ = DFEABCHG$

keys

$DH^+ = DH$ ✗ → should continue finding keys

prime: ABEFD
non: CGH

(i) not BCNF ($CH \to G$)

(ii) not 3NF ($CH \to G$)

(iii) not 2NF ($A \to BC$)

∴ 1NF

Q: R(ABCD)

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow BD$

A B C D

$A^+ = ABCD$

$\therefore$ 2NF

Q: R(ABCDEF)

$AB \rightarrow CD$, $CD \rightarrow EF$, $BC \rightarrow DEF$, $D \rightarrow C$, $CE \rightarrow F$

A B C D E F

$AB^+ = ABCDEF \longrightarrow$ key

prime: AB

non-prime: CDEF

(i) BCNF $\rightarrow$ fails

$CD \rightarrow EF$

(ii) 3NF $\rightarrow$ fails

$CD \rightarrow EF$

(iii) 2NF $\rightarrow$ fails

$BC \rightarrow DEF$

$\therefore$ 1NF

Q: Check equivalency of F and G on R

R(ABC)

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

$G = \{A \rightarrow BC, B \rightarrow A, C \rightarrow A\}$

F covers G

A^+, B^+, C^+ on F

$A^+ = ABC$

$B^+ = BCA$

$C^+ = CAB$

A^+, B^+, C^+ on G

$A^+ = ABC$

$B^+ = BAC$

$C^+ = CAB$

$\therefore$ F covers G $\Rightarrow$ $F \geq G$

G covers F

same LHS

$\therefore$ G covers F

$\therefore$ F and G are equivalent

Q: R(VWXYZ)

$F = \{W \to X, WX \to Y, Z \to WY, Z \to V\}$

$G = \{W \to XY, Z \to WX\}$

F covers G

W^+, Z^+ on G

$W^+ = WXY$

$Z^+ = ZWXY$

W^+, Z^+ on F

$W^+ = WXY$

$Z^+ = ZWYVX$

$F \supseteq G$

G covers F

W^+, WX^+, Z^+ on F

$W^+ = WXY$
$WX^+ = WXY$
$Z^+ = ZWYVX$

W^+, WX^+, Z^+ on G

$W^+ = WXY$
$WX^+ = WXY$
$Z^+ = ZWXY$

$G \not\supseteq F$

$\therefore$ F and G are not equivalent

Q: Find minimal cover of $FD = \{A \rightarrow C,\ AB \rightarrow C,\ C \rightarrow DI,\ CD \rightarrow I,\ EC \rightarrow AB,\ EI \rightarrow C\}$

1) Canonical

$A \rightarrow C,\ AB \rightarrow C,\ C \rightarrow D,\ C \rightarrow I,\ CD \rightarrow I,$

$EC \rightarrow A,\ EC \rightarrow B,\ EI \rightarrow C$

2) Extraneous attr

closures of attributes

(i) $A^+ = ACDI$

(ii) $B^+ = B$

(iii) $C^+ = CDI$

(iv) $D^+ = D$

(v) $E^+ = E$

(vi) $I^+ = I$

$A \to C$ already

$C \to I$ already

$A \to C$, $AB \to C$, $C \to D$, $C \to I$, $CD \to I$,

$EC \to A$, $EC \to B$, $EI \to C$

required required req

$\therefore F' = \{ A \to C, C \to D, C \to I, EC \to A, EC \to B, EI \to C \}$

(iii) Redundant FDs

$F' = \{ A \to C, C \to D, C \to I, EC \to A, EC \to B, EI \to C \}$

$F^n = \{ A \to C,\ C \to D,\ C \to I,\ EC \to A,\ EC \to B \}$

CURSOR

- Can be used inside procedure, function

- Binary
- Insensitive
- Scroll

- DECLARE

- Read syntax for ISA