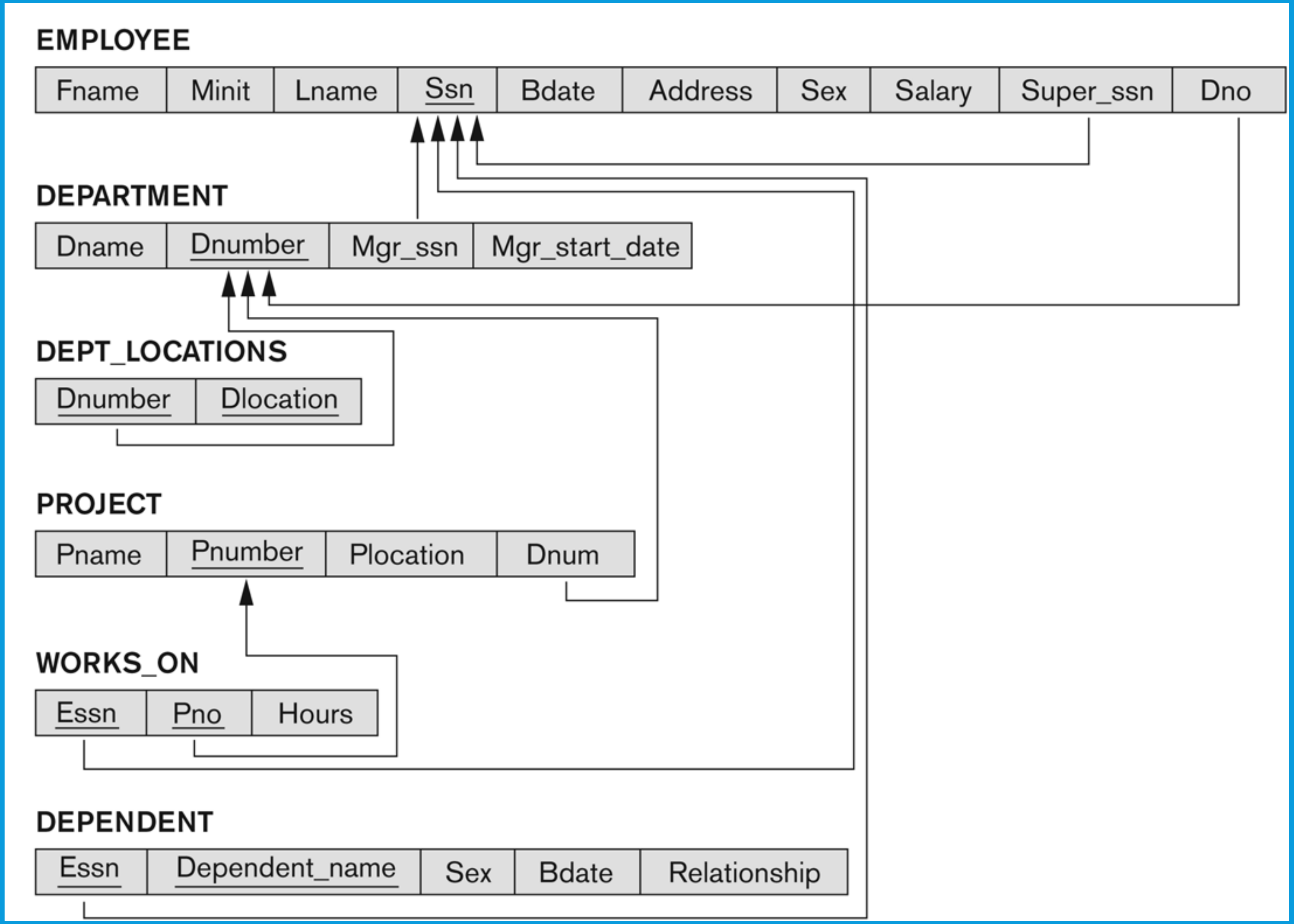
Lecture 6: Functional Dependency & Normalization

CS3402 Database Systems

The COMPANY Relational Database Schema



Populated Database State for COMPANY

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Functional Dependency

- Functional dependency is a constraint between two sets of attributes from the database
  - For example, deptno and dname in DEPARTMENT, if you know the department number, you know the department name
- A functional dependency denoted by $X \rightarrow Y$ specifies a constraint on the possible tuples between two sets of attributes X and Y that are subsets of a relation R 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 $t_1[Y] = t_2[Y]$
  - The values of the Y component of a tuple in r depend on, or are determined by the values of the X component
  - If you know his student ID, then I know his name ($X \rightarrow Y$)

Functional Dependency: Formal Definition

- Let R be a relation schema, and $\alpha \subseteq R$, $\beta \subseteq R$ (i.e., α and β are sets of R's attributes).
- We say $\alpha \rightarrow \beta$, if in any relation instance r(R), for all pairs of tuples t_1 and t_2 in r, we have $(t_1[\alpha] = t_2[\alpha]) \rightarrow (t_1[\beta] = t_2[\beta])$

Functional Dependency: Example

- Movies(title, year, length, type, studioName, starName): {title, year, starName} $\rightarrow$ {length, type, studioName}
  - Attributes {title, year, starName} form a key for the relation Movie
  - If two tuples agree on these three attributes, title, year, and starName, they must agree on the other attributes, length, type and studioName.
  - No proper subset of {title, year, starName} functionally determines all other attributes
    - {title, year} does not determine starName since many movies have more than one star
    - {year, starName} is not a key because we could have a star in two movies in the same year
  - Can it be {title, year, starName, length} $\rightarrow$ type? Yes

Functional Dependency: Candidate Key

- Candidate key
  - If a constraint on R states X is a candidate key of R, then X $\rightarrow$ Y for any subset of attributes Y of R
  - A candidate key uniquely identifies a tuple
  - The values of all remaining attributes are determined
- If X $\rightarrow$ Y in R, this does not say whether or not Y $\rightarrow$ X in R
  - {length, type, studioName} $\rightarrow$ {title, year, starName}? No
- A functional dependency is property of the semantics or meaning of the attributes

Trivial Functional Dependency

- Some functional dependencies are "trivial", since they are always satisfied by all relations:
  - E.g., A $\rightarrow$ A, AB $\rightarrow$ A,
  - E.g., {Ename, Salary} $\rightarrow$ Ename
- A functional dependency is trivial if and only if the right-hand side (the dependent) is a subset of the left-hand side (the determinant)
  - E.g., AB $\rightarrow$ A

Inference Rules for FDs (1/2)

- Given a set of FDs F, we can infer additional FDs that hold whenever the FDs in F hold
- Armstrong's inference rules:
  - **IR1. (Reflexivity) If Y $\subseteq$ X (i.e., Y is a subset of X), then X $\rightarrow$ Y**
  - **IR2. (Augmentation) If X $\rightarrow$ Y, then XZ $\rightarrow$ YZ (Note: XZ stands for X $\cup$ Z)**
  - **IR3. (Transitivity) If X $\rightarrow$ Y and Y $\rightarrow$ Z, then X $\rightarrow$ Z**
- IR1, IR2, IR3 form a sound and complete set of inference rules
  - Sound: These rules are true
  - Complete: All the other rules that are true can be deduced from these rules

Inference Rules for FDs (2/2)

- Some additional inference rules that are useful:
  - **Decomposition: If X $\rightarrow$ YZ, then X $\rightarrow$ Y and X $\rightarrow$ Z**
    - Since X $\rightarrow$ YZ (given) and YZ $\rightarrow$ Y (reflexivity), X $\rightarrow$ Y (transitivity)
    - Since X $\rightarrow$ YZ (given) and YZ $\rightarrow$ Z (reflexivity), X $\rightarrow$ Z (transitivity)
  - **Union: If X $\rightarrow$ Y and X $\rightarrow$ Z, then X $\rightarrow$ YZ**
  - **Pseudo transitivity: If X $\rightarrow$ Y and YW $\rightarrow$ Z, then XW $\rightarrow$ Z**
    - Since YW $\rightarrow$ Z (given) and XW $\rightarrow$ YW (augmentation), XW $\rightarrow$ Z (transitivity)

Inference Rules for FDs: Example

- Suppose we are given a schema R with attributes A, B, C, D, E, F and the FDs are:
  - $A \rightarrow BC$
  - $B \rightarrow E$
  - $CD \rightarrow EF$
  - Show that FD: $AD \rightarrow F$ holds

Solution

1. $A \rightarrow BC$ (given)
2. $A \rightarrow C$ (decomposition from 1)
3. $AD \rightarrow CD$ (augmentation from 2)
4. $CD \rightarrow EF$ (given)
5. $AD \rightarrow EF$ (transitivity from 3 and 4)
6. $AD \rightarrow F$ (decomposition from 5) (proved)

Closure of a Set of FDs (1/2)

- Given a set of FDs F, there are certain other FDs that are logically implied by F based on Armstrong's inference rules (i.e., reflexivity, augmentation and transitivity).
- The set of all FDs logically implied by F is the closure of F, denoted by F^+.
- (Reflexivity) If $Y \subseteq X$, then $X \rightarrow Y$
  - Ssn $\rightarrow$ Ssn
  - {Ssn, Dmgr_ssn} $\rightarrow$ Ssn
  - {Ssn, Dmgr_ssn} $\rightarrow$ Dmgr_ssn

Closure of a Set of FDs (2/2)

- (Augmentation) If $X \rightarrow Y$, then $XZ \rightarrow YZ$
  - Ssn $\rightarrow$ Ename (given)
  - {Ssn, Address} $\rightarrow$ {Ename, Address}
- (Transitivity) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
  - Ssn $\rightarrow$ Dnumber (given)
  - Dnumber $\rightarrow$ {Dname, Dmgr_ssn} (given)
  - Ssn $\rightarrow$ {Dname, Dmgr_ssn}

Closure of a Set of FDs: Example

- $R = \{A, B, C, D, E, F\}$
- FDs in F:
  - $A \rightarrow B$
  - $A \rightarrow C$
  - $CD \rightarrow E$
  - $CD \rightarrow F$
  - $B \rightarrow D$
- Some members of F^+:
  - $A \rightarrow D$
  - $A \rightarrow BC$
  - $AD \rightarrow E$
  - $CD \rightarrow EF$
  - …

Closure of Attribute Sets

- The closure of X under F (denoted by X^+) is the set of attributes that are functionally determined by X under F (X and X^+ are a set of attributes):

$$X \rightarrow Y \text{ in } F^+ \leftrightarrow Y \subseteq X^+$$

- If X^+ consists of all attributes of R, X is a superkey for R. From the value of X, we can determine the values of the whole tuple.
- For example, given Ssn, if Ssn $\rightarrow$ Ename, then Ename is part of Ssn^+, i.e., Ssn^+ = {Ssn, Ename, …}
- If Ssn $\rightarrow$ Dmgr_ssn, then Dmgr_ssn is part of Ssn^+, i.e., Ssn^+ = {Ssn, Ename, Dmgr_ssn, …}

Closure of Attribute Sets: Algorithm

- Input
  - R: a relation schema
  - F: a set of FDs
  - $X \subset R$: the set of attributes for computing the closure
- Output
  - X^+ is the closure of X with respect to F

$X_0 = X$

Repeat

$X_{i+1} = X_i \cup Z$, where Z is the set of attributes such that $Y \rightarrow Z$ in F and $Y \subset X_i$

Until $X_{i+1} = X_i$

Return X_{i+1}

Closure of Attribute Sets: Example

- Given a schema R={A, B, C, D, E, F}, F= {$A \rightarrow BC$, $B \rightarrow E$, $E \rightarrow CF$, $CD \rightarrow EF$}, and X={A}.
- X_0={A}
  $A \rightarrow BC$
- X_1={A, B, C}
  $B \rightarrow E$
- X_2={A, B, C, E}
  $E \rightarrow CF$
- X_3={A, B, C, E, F}
- Output: X^+={A, B, C, E, F}

Equivalence of Sets of FDs

- A set of functional dependencies F is said to cover another set of functional dependency E if every FD in E is also in F^+ (E is a subset of F^+)
- Two sets of FDs F and G are equivalent if
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$
- Example:
  - F: $A \rightarrow BC$; {$A \rightarrow B$, $A \rightarrow C$ (decomposition rule)}
  - G: $A \rightarrow B$, $A \rightarrow C$
  - $F^+ = G^+$

Relational Database Design

- Logical/conceptual DB design
  - Schema
    - What relations (tables) are needed?
    - What their attributes should be?
- What is a “bad” DB design?
  - Repetition of data/information
  - Potential inconsistency
  - Inability to represent certain information
  - Loss of data/information

Normalization (1/2)

- Normalization was proposed by Codd in 1972 to take a relation schema through a series of tests to certify whether it satisfies a certain normal form
- Analyzing the relation schema based on FD and primary keys to achieve
  - Minimizing redundancy
  - Minimizing the insertion, deletion and update anomalies

Normalization (2/2)

- Normalization requires two properties
  - Non-additive or lossless join
    - Decomposition is reversible and no information is loss
    - No spurious tuples (tuples that should not exist) should be generated by doing a natural-join of any relations (extremely important)
  - Preservation of the functional dependencies
    - Ensure each functional dependency is represented in some individual relation (sometimes can be sacrificed)

Lossless Decomposition

S#	Status	City
S3	30	Paris
S5	30	Athens

Decomposition

Natural Join

S#	Status
S3	30
S5	30

S#	City
S3	Paris
S5	Athens

S#	Status
S3	30
S5	30

Status	City
30	Paris
30	Athens

Natural Join

Status	City	S#
30	Paris	S3
30	Paris	S5
30	Athens	S3
30	Athens	S5

First Normal Form with Primary Key (1/4)

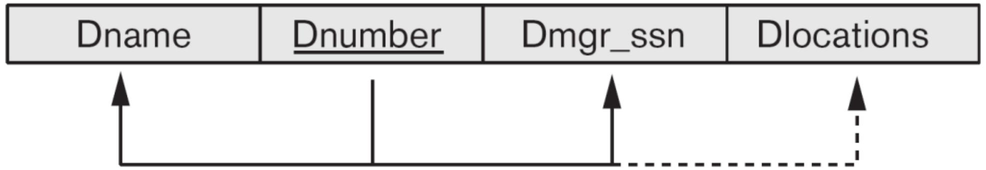
- First normal form (1NF)
  - Disallow multivalued attributes, composite attributes and their combination
  - Disallow multivalued attributes that are themselves composite
  - The domain of an attribute must be atomic (simple and indivisible) values
  - No repeating groups in a relation (no nested relations)
- For example, each department can have a number of locations
  - 1NF: DEPT_LOCATIONS(Dnumber, Dlocation)

First Normal Form with Primary Key (2/4)

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

(a)

DEPARTMENT



(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

- Better solution: DEPARTMENT(Dname, Dnumber, Dmgr_ssn) and DEPT_LOCATIONS (Dnumber, Dlocation)

First Normal Form with Primary Key (3/4)

- 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.

(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

First Normal Form with Primary Key (4/4)

- Example: FIRST(S#, Status, City, P#, Qty)
  - What's the primary key?
  - R = {S#, P#, Qty, Status, City}
  - F = {{S#, P#} → Qty, S# → {Status, City}}
- Possible Solution
  - Replace the original table by two sub-tables
  - SECOND(S#, Status, City)
  - SP(S#, P#, Qty)

Second Normal Form with Primary Key (1/3)

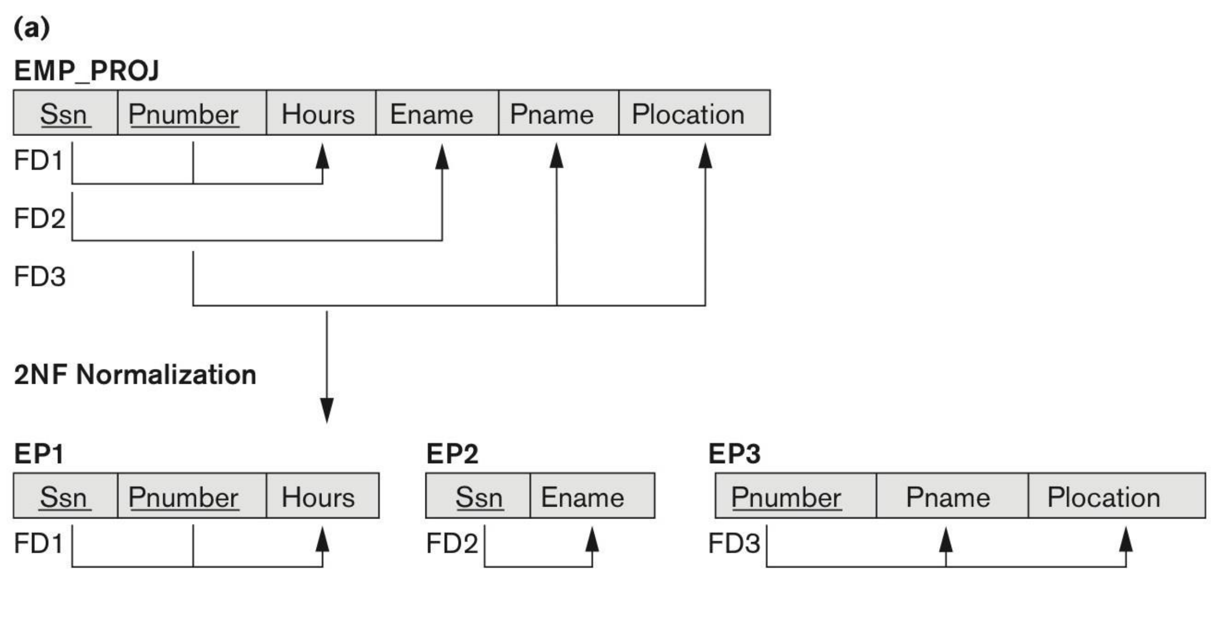
- Full functional dependency
  - If removal of any attribute A from X means that the dependency does not hold any more
  - E.g., {Ssn, Pnumber} → Hours
- Partial functional dependency
  - If some attributes A belonging to X can be removed from X and the dependency still holds
  - E.g., {Ssn, Pnumber} → Ename as Ssn → Ename

Second Normal Form with Primary Key (2/3)

- A relation schema R is in 2NF if every non-prime attributes A in R is fully functional dependent on the primary key of R
- An attribute of R is called prime attribute of R if it is a member of some candidate key of R. Otherwise it is non-prime.
- For example,
  - Non-prime attributes: Hours, Ename, Pname, Plocation
  - Primary key: {Ssn, Pnumber}
- If a relation schema is not in 2NF, it can be 2NF normalized into a number of 2NF relations in which non-prime attributes are associated only with the part of the primary key on which they are fully functional dependent

Second Normal Form with Primary Key (3/3)

- The non-prime attribute Ename violates 2NF because of FD2 (i.e., is not fully functional dependent on the primary key)
- Similarly, Pname and Plocation violate 2NF because of FD3
- Solution: EP1(Ssn, Pnumber, Hours); EP2(Ssn, Ename), and EP3(Pnumber, Pname, Plocation)

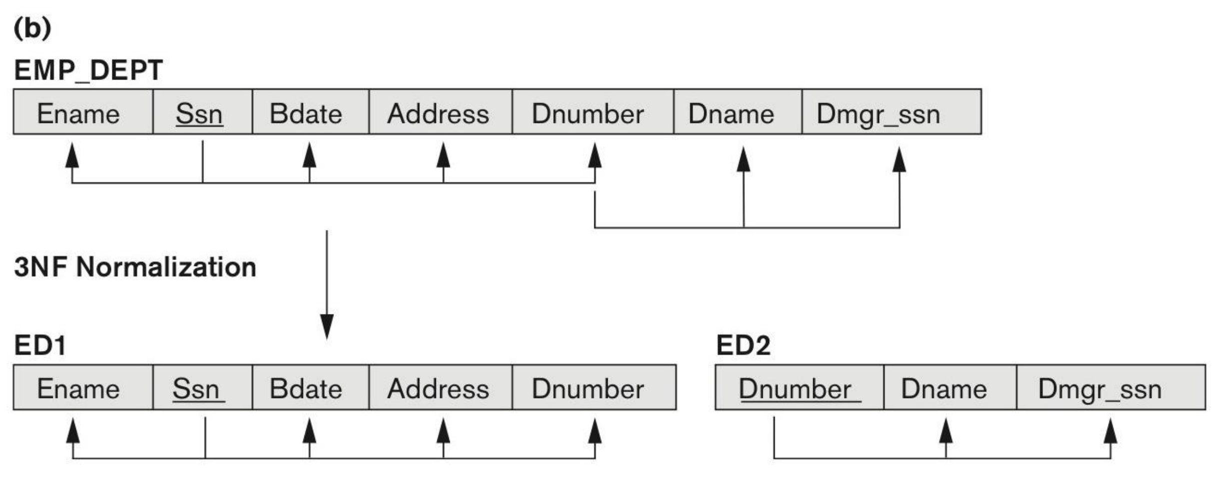


Third Normal Form with Primary Key (1/2)

- A relation schema R is in 3NF if whenever a non-trivial FD X → A holds in R, either (a) X is a Superkey of R or (b) A is a prime attribute of R.
- 3NF is based on the concept of transitive dependency
- A functional dependency X → Y in a relation schema R is transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold
- X → Z → Y (Z is not a candidate key nor a subset of any key)
- For example, dependency Ssn → Dmgr_ssn is transitive through Dnumber
  - Ssn → Dnumber and Dnumber → Dmgr_ssn, and Dnumber is neither a key itself nor a subset of the key
  - Ssn → Dnumber → Dmgr_ssn

Third Normal Form with Primary Key (2/2)

- According to Codd's original definition, a relation scheme R is in 3NF if it satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key
- The Ssn → Dnumber, Dumber → {Dname, Dmgr_ssn} and Dnumber is neither a candidate key nor a subset of any key of EMP_DEPT, so EMP_DEPT violates 3NF
- Solution: ED1(Ename, Ssn, Bdate, Address, Dnumber), ED2(Dnumber, Dname, Dmgr_ssn)



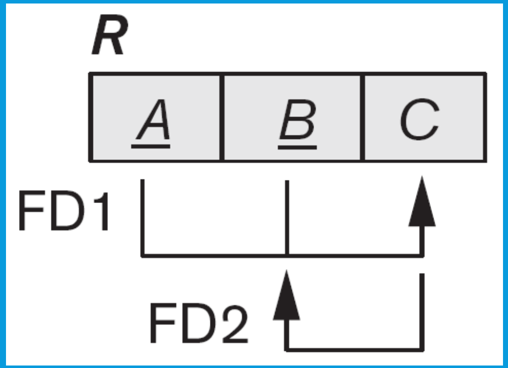
General Definitions of Normal Forms

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

- BCNF was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF
  - Every relation in BCNF is also in 3NF
  - BUT Relation in 3NF is not necessarily in BCNF
- A relation schema R is in BCNF if whenever a non-trivial functional dependency $X \rightarrow A$ holds in R, then X is a superkey of R.
- For example, the relation schema below is in 3NF (B is a prime attribute) but not in BCNF.



Algorithm for BCNF Decomposition

- Let R be the initial table with FDs F and S={R}

  Until all relation schemes in S are in BCNF

  for each R in S

  for each FD X → Y that violates BCNF for R

  S = (S – {R}) ∪ (R-Y) ∪ (X,Y)

  End until

- When we find a table R with BCNF violation X → Y we:
  - Remove R from S
  - Add a table that has the same attributes as R except for Y
  - Add a second table that contains the attributes in X and Y

BCNF Decomposition: Example (1/2)

- Let us consider the relation scheme R=(A,B,C,D,E) and the FDs:
  $\{A\} \rightarrow \{B,E\}$, $\{C\} \rightarrow \{D\}$
- Candidate key: AC
- Both functional dependencies violate BCNF because the LHS is not a candidate key
- Pick $\{A\} \rightarrow \{B,E\}$
  - We can also choose $\{C\} \rightarrow \{D\}$ – different choices
  - Lead to different decompositions.
  - (A,B,C,D,E) generates R_1=(A,C,D) and R_2=(A,B,E)

BCNF Decomposition: Example

- Let us consider the relation scheme R=(A,B,C,D,E) and the FDs: {A} → {B,E}, {C} → {D}
- Candidate key: AC
- Both functional dependencies violate BCNF because the LHS is not a candidate key
- Pick {A} → {B,E}
  - We can also choose {C} → {D} – different choices
  - Lead to different decompositions.
  - (A,B,C,D,E) generates R_1=($\underline{A}$,$\underline{C}$,D) and R_2=($\underline{A}$,B,E)
- We need to decompose R_1=($\underline{A}$,$\underline{C}$,D) because of the FD {C} → {D}, so ($\underline{A}$,$\underline{C}$,D) is replaced with R_3=(A,C) and R_4=(C,D).
- Final decomposition: R_2=(A,B,E), R_3=(A,C), R_4=(C,D)