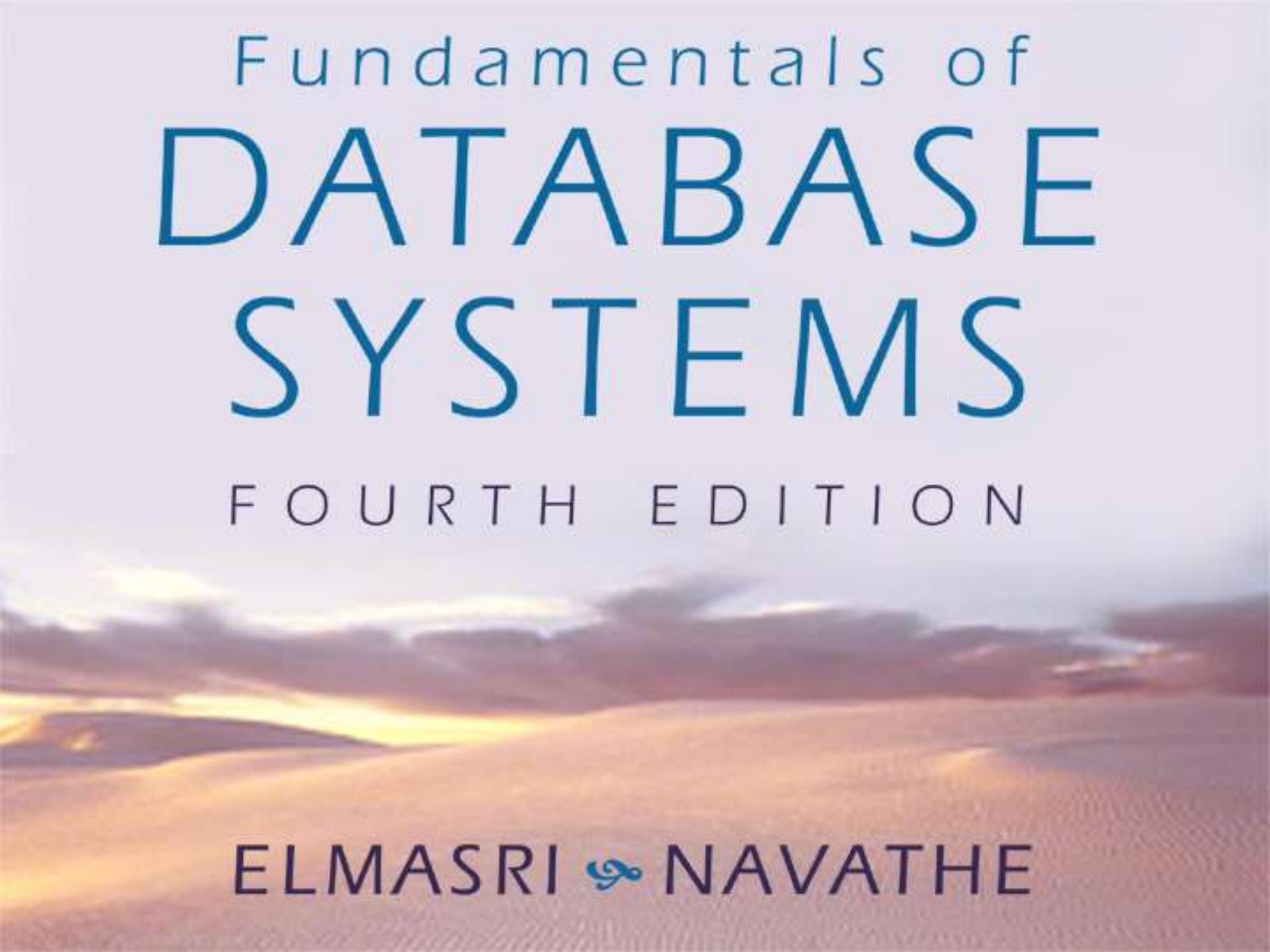
Fundamentals of DATABASE SYSTEMS

FOURTH EDITION



ELMASRI NAVATHE

Chapter 10

Functional Dependencies and Normalization for Relational Databases

Chapter Outline

1 Informal Design Guidelines for Relational Databases

- 1.1Semantics of the Relation Attributes
- 1.2 Redundant Information in Tuples and Update Anomalies
- 1.3 Null Values in Tuples
- 1.4 Spurious Tuples

2 Functional Dependencies (FDs)

- 2.1 Definition of FD
- 2.2 Inference Rules for FDs
- 2.3 Equivalence of Sets of FDs
- 2.4 Minimal Sets of FDs

Chapter Outline(contd.)

3 Normal Forms Based on Primary Keys

- 3.1 Normalization of Relations
- 3.2 Practical Use of Normal Forms
- 3.3 Definitions of Keys and Attributes Participating in Keys
- 3.4 First Normal Form
- 3.5 Second Normal Form
- 3.6 Third Normal Form

4 General Normal Form Definitions (For Multiple Keys)

5 BCNF (Boyce-Codd Normal Form)

1 Informal Design Guidelines for Relational Databases (1)

- What is relational database design?

  The grouping of attributes to form "good" relation schemas

- Two levels of relation schemas
  - The logical "user view" level
  - The storage "base relation" level
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 11

1.1 Semantics of the Relation Attributes

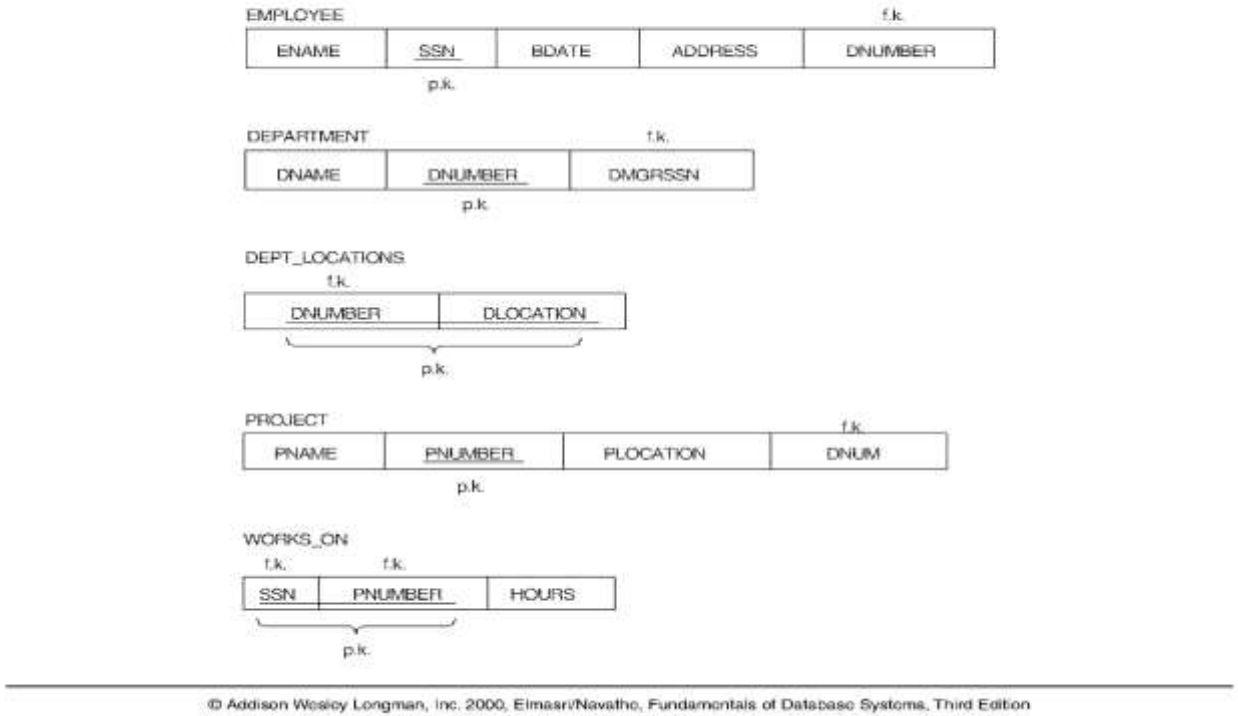
GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

Figure 10.1 A simplified COMPANY relational database schema

Figure 14.1 Simplified version of the COMPANY relational database schema.



Note: The above figure is now called Figure 10.1 in Edition 4

1.2 Redundant Information in Tuples and Update Anomalies

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:

EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.

EXAMPLE OF AN UPDATE ANOMALY (2)

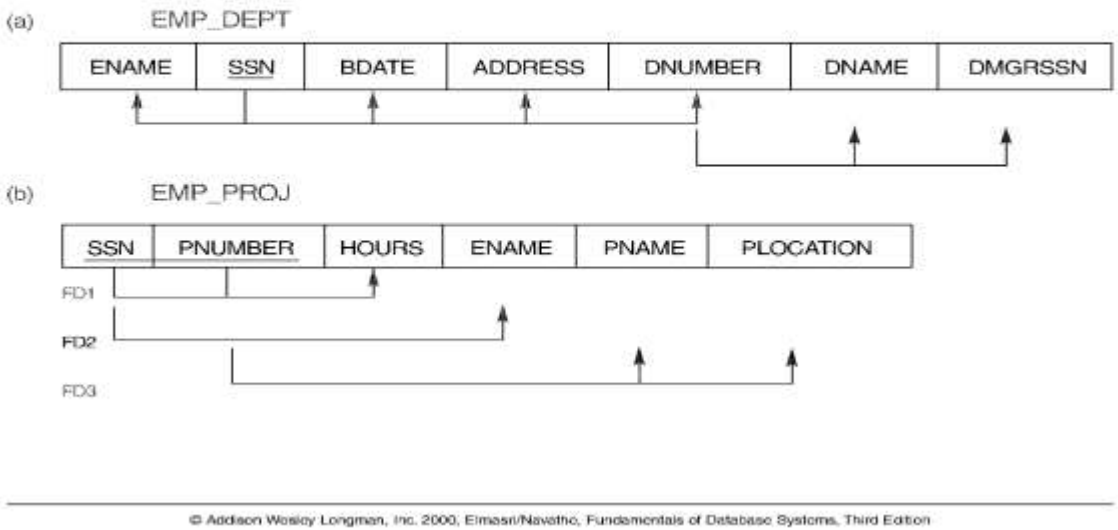
- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely **-** Cannot insert an employee unless an he/she is assigned to a project.

- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Figure 10.3 Two relation schemas suffering from update anomalies

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Note: The above figure is now called Figure 10.3 in Edition 4

Figure 10.4 Example States for EMP_DEPT and EMP_PROJ

Figure 14.4 Example relations for the schemas in Figure 14.3 that result from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

Note: The above figure is now called Figure 10.4 in Edition 4

Guideline to Redundant Information in Tuples and Update Anomalies

- **GUIDELINE 2:** Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account

1.3 Null Values in Tuples

GUIDELINE 3: Relations should be designed such that their tuples will have as few NULL values as possible

- Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable

1.4 Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

GUIDELINE 4: The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

Spurious Tuples (2)

There are two important properties of decompositions:

(a) non-additive or losslessness of the corresponding join

(b) preservation of the functional dependencies.

Note that property (a) is extremely important and *cannot* be sacrificed. Property (b) is less stringent and may be sacrificed. (See Chapter 11).

2.1 Functional Dependencies (1)

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

Functional Dependencies (2)

- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- For any two tuples t1 and t2 in any relation instance r(R): *If* t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints (1)

- Ex SSN -> Ename

SSN	Ename
1	A
2	B
3	C
4	D

SSN	Ename
1	A
2	A
3	C
4	D

SSN	Ename
1	A
1	X
2	B
3	C
4	D

SSN	Ename
1	A
1	A
2	B
3	C
4	D

Examples of FD constraints (1)

- Ex

1. A -> BC
2. DE -> C
3. C -> DE
4. BC -> A

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

Examples of FD constraints (1)

- social security number determines employee name

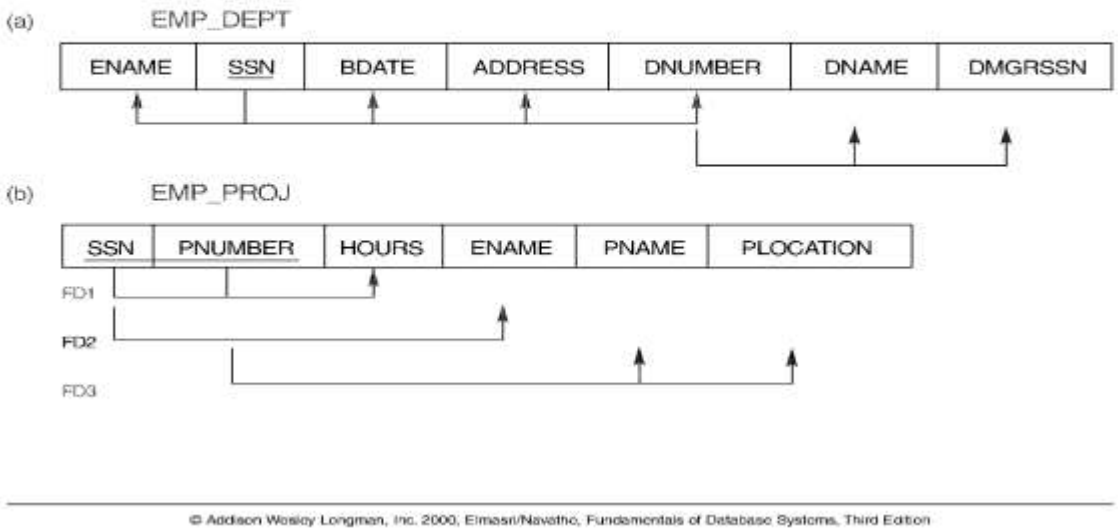
  SSN -> ENAME
- project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}
- employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} -> HOURS

Figure 10.3 Two relation schemas suffering from update anomalies

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Note: The above figure is now called Figure 10.3 in Edition 4

Examples of FD constraints (2)

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every relation instance* r(R)
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with t1[K]=t2[K])

2.2 Inference Rules for FDs (1)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold

Armstrong's inference rules:

IR1. (**Reflexive**) If Y *subset-of* X, then X -> Y

IR2. (**Augmentation**) If X -> Y, then XZ -> YZ

(Notation: XZ stands for X ∪ Z)

IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z

- IR1, IR2, IR3 form a *sound* and *complete* set of inference rules

Inference Rules for FDs (2)

Some **additional inference rules** that are useful:

(**Decomposition**) If X -> YZ, then X -> Y and X -> Z

(**Union**) If X -> Y and X -> Z, then X -> YZ

(**Psuedotransitivity**) If X -> Y and WY -> Z, then WX -> Z

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Inference Rules for FDs (3)

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

2.3 Equivalence of Sets of FDs

- 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 $F^+ = G^+$

Definition: F **covers** G if every FD in G can be inferred from F (i.e., if G^+ *subset-of* F^+)

- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

2.4 Minimal Sets of FDs (1)

- A set of FDs is **minimal** if it satisfies the following conditions:

(1) Every dependency in F has a single attribute for its RHS.

(2) We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.

(3) We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.

Minimal Sets of FDs (2)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set (e.g., see algorithms 11.2 and 11.4)

3 Normal Forms Based on Primary Keys

3.1 Normalization of Relations

3.2 Practical Use of Normal Forms

3.3 Definitions of Keys and Attributes Participating in Keys

3.4 First Normal Form

3.5 Second Normal Form

3.6 Third Normal Form

3.1 Normalization of Relations (1)

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Normalization of Relations (2)

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies : MVDs; 5NF based on keys, join dependencies : JDs (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)

3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization:** the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

3.3 Definitions of Keys and Attributes Participating in Keys (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is a set of attributes S *subset-of* R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$

- A **key** K is a superkey with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more.

Definitions of Keys and Attributes Participating in Keys (2)

- If a relation schema has more than one key, each is called a **candidate key.** One of the candidate keys is *arbitrarily* designated to be the **primary key,** and the others are called *secondary keys*.
- A **Prime attribute** must be a member of *some candidate key*
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

3.3 Definitions of Keys and Attributes Participating in Keys (1)

- Ex: Employee
  - **Key:** SSN
  - **Superkeys:** {SSN}, {SSN, Ename}, {SSN, Ename, BDate} & any set of attributes that includes SSN are all superkeys
- {BookID}, {Name, Author},{Name, BookID, Author}
- fdS : {EF->G, F->IJ, EH->KL, K->M, L->N}
  - EF+ => {EF, G, IJ} => {E, F, G, I,J}
  - EFH+ => {EFH, G,IJ,KL,M,N} => {E,F,H,G,I,J,K,L,M,N}
  - EFHKL+ => {E,F,H,G,I,J,K,L,M,N}
  -
  - EFHKL & **EFH**

R(ABCDE) = {AB->C, B->D, C->E, D->A}

- (B)+ ={B,D,A,C,E}
- **$(C,D)^+$ = {C,D,E,A}**
- **$(B,C)^+$ ={B,C,D,E,A}**

- Example

Consider the given relation R with attributes F1,F2,F3,F4 and F5.

R(F1,F2,F3,F4,F5)
{
F1->F3
F3->F2
F2->F4
F4->F5
}

Solution:

closure(F1) = {F1,F3,F2,F4,F5} [Reason: self is always included, from F1->F3, As F1->F3 and F3->F2 hence F1->F2(Transitive rule), Similarly others also]
closure(F3) = {F3,F2,F4,F5}
closure(F2) = {F2,F4,F5}
closure(F4) = {F4,F5}

Now look for the best closure among all the closures you have found which can derive all the attributes of the given relation R. Here you can see that F1 is the attribute which can derive all the other attributes of the given relation. Hence **F1 is the candidate key** of the given relation.

- Example:

Consider the given relation R with attributes F1,F2,F3,F4 and F5.

R(F1,F2,F3,F4,F5)
{
F1->F3
F2->F4
F1F2->F5
}

Solution:

closure(F1) = {F1,F3}
closure(F2) = {F2,F4}
closure(F1F2) = {F1,F2,F5,F3,F4} [Reason: F1F2 can self derive F1 and F2, F1 can derive F3 and F2 can derive F4 hence by transitivity rule F1F2 can derive F3 and F4 also]

As F1F2 can derive all the attributes of the relation, here**F1F2 is the candidate**

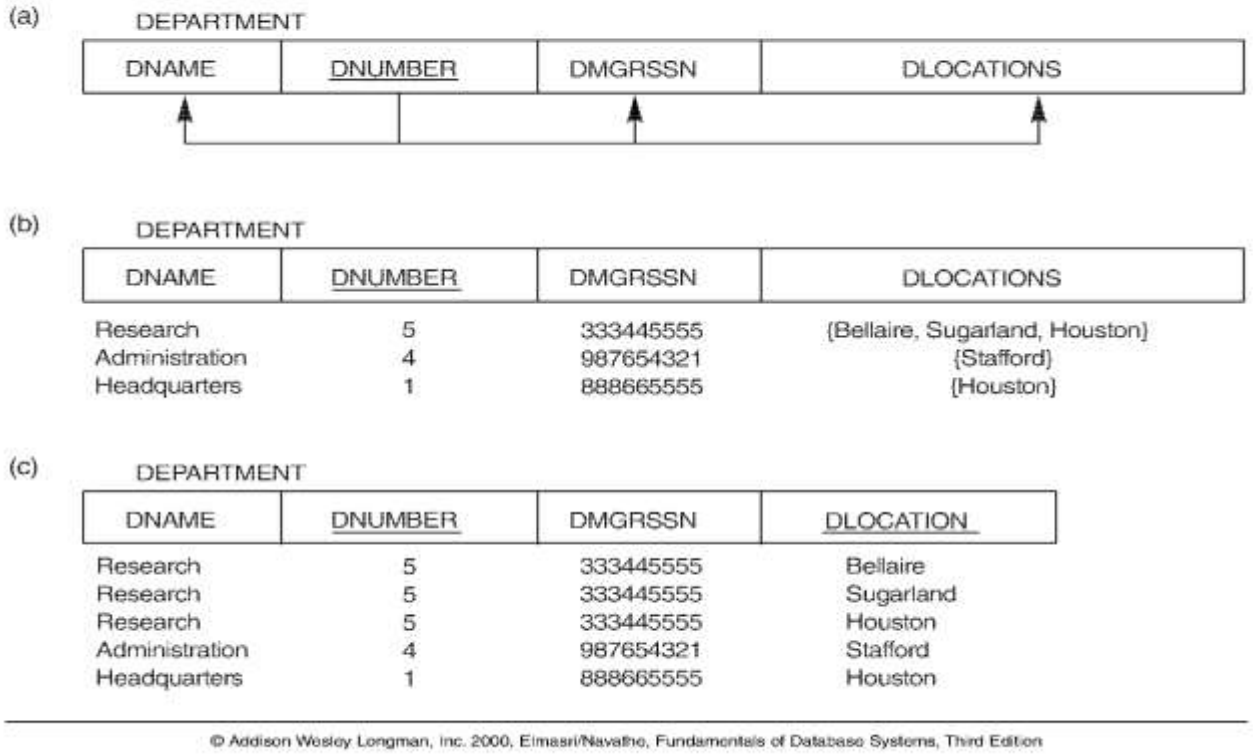
3.2 First Normal Form

- Disallows composite attributes, multivalued attributes, and **nested relations**; attributes whose values *for an individual tuple* are non-atomic

- Considered to be part of the definition of relation

Figure 10.8 Normalization into 1NF

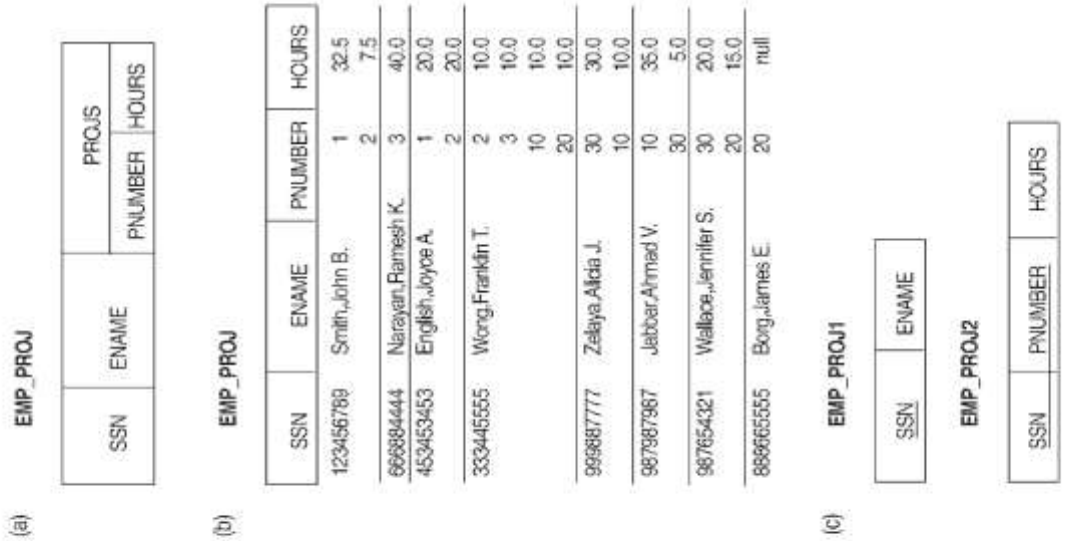
Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.



Note: The above figure is now called Figure 10.8 in Edition 4

Figure 10.9 Normalization nested relations into 1NF

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.



Note: The above figure is now called Figure 10.9 in Edition 4

3.3 Second Normal Form (1)

- Uses the concepts of **FD**s, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

- {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a *partial dependency*) since SSN -> ENAME also holds

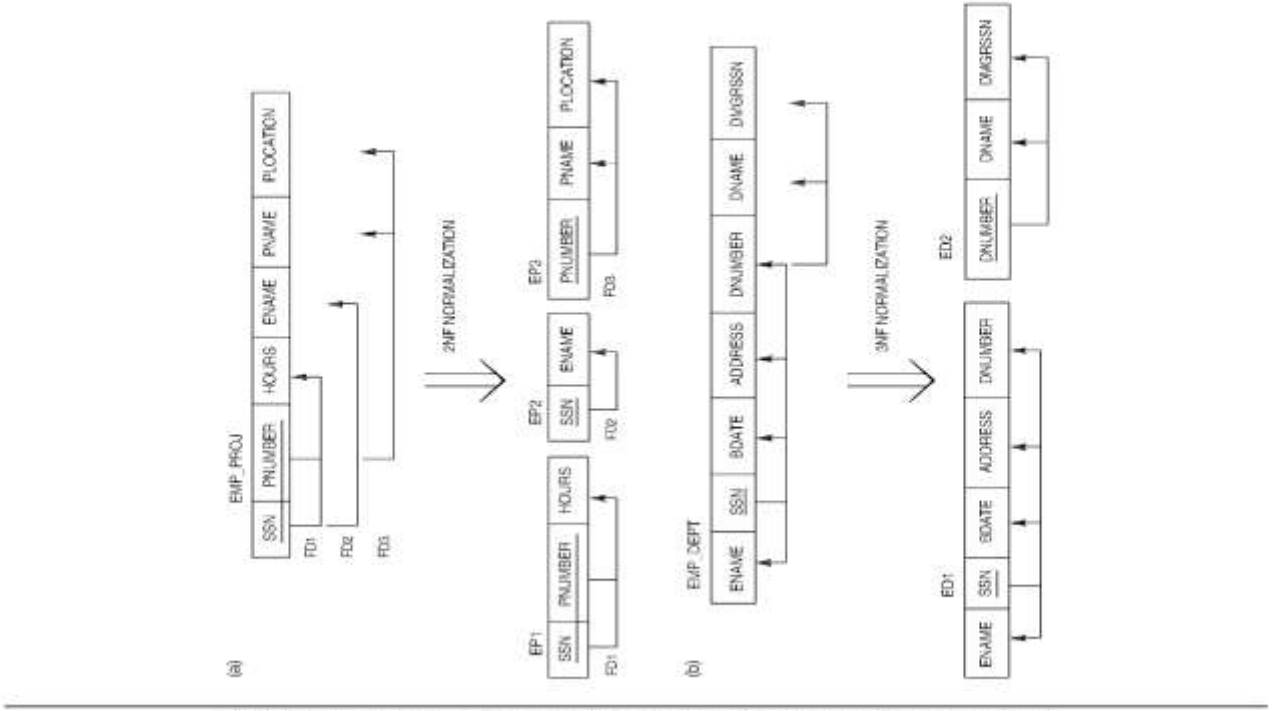
Second Normal Form (2)

- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on the primary key

- R can be decomposed into 2NF relations via the process of 2NF normalization

Figure 10.10 Normalizing into 2NF and 3NF

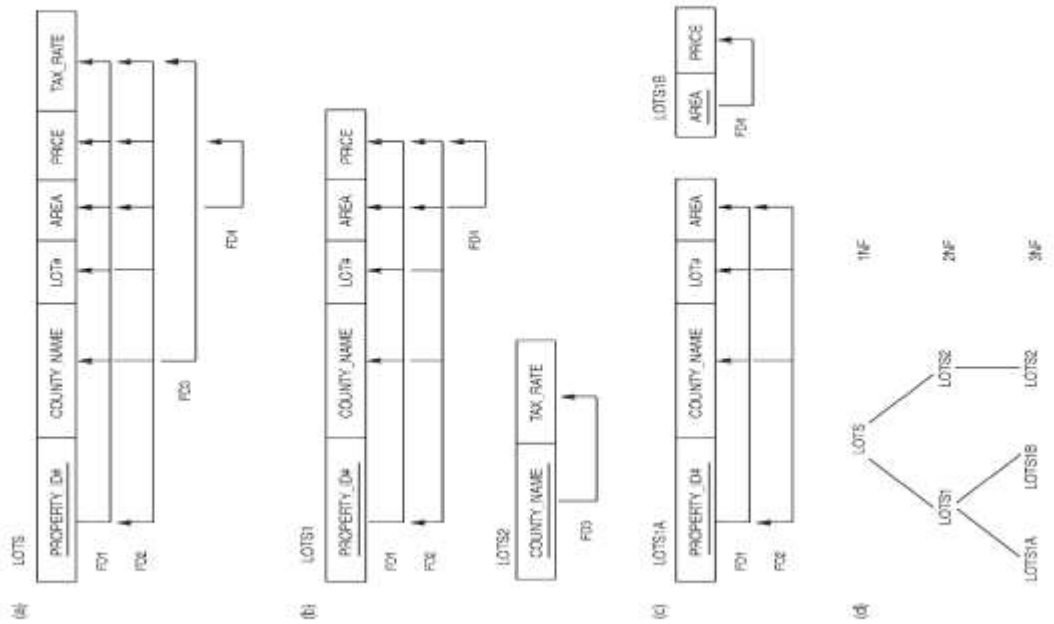
Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



Note: The above figure is now called Figure 10.10 in Edition 4

Figure 10.11 Normalization into 2NF and 3NF

Figure 14.11 Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies fd1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.



Note: The above figure is now called Figure 10.11 in Edition 4

3.4 Third Normal Form (1)

Definition:

- **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

  Examples:

  - SSN -> DMGRSSN is a *transitive* FD since

  SSN -> DNUMBER and DNUMBER -> DMGRSSN hold

  - SSN -> ENAME is *non-transitive* since there is no set of attributes X where SSN -> X and X -> ENAME

Third Normal Form (2)

- A relation schema R is in **third normal form** (**3NF**) if,
  1) it is in 2NF
  2) no non-prime attribute A in R is transitively dependent on the primary key OR
  2) A non-trivial FD X->A holds in R either
     i) X is a Super Key of R OR
     ii) A is prime attribute of R
- R can be decomposed into 3NF relations via the process of 3NF normalization

NOTE:

In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

E.g., Consider EMP (SSN, Emp#, Salary).

Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

Example

- R1(ABCD)
- FDs: ACD -> B
  AC -> D
  D->C
  AC-> B
- Find all candidate key and check for 3NF

Answer

- R1(ABCD)
- FDs: ACD -> B, AC -> D, D->C, AC-> B
- Ans:

  Candidate key : AC & AD

  Prime attributes: A, C &D

  Non Prime attributes: B
- 1NF: No nested relations & no multivalued attributes
- 2NF:All FDs are satisfying condition of full functional dependency
- 3NF: No transitive dependency.

Example

- R1(ABCDEF)
- FDs: A->C, C->D, D->B, E->F
- Find all candidate key.

Answer:

Superkey set :{A+ ->ACDB, …. AC+ -> ACDB, AE+ -> ABCDEF, ….. ADE+ -> ABCDEF …. }

Candidate keys: AE+ -> ABCDEF

Prime Attribute: A,E

Non Prime Attribute: B,C,D,F

Example

- R1(ABCDEFGH)
- FDs: CH>G, A->BC, B->CFH, E->A, F->EG
- Find all candidate key.

Answer:

Superkey set :{D+->D, DA+ ->DABCFHEG, DB+ -> DBCFHEGA, DC+ -> DC, DE+ -> DEA, DF+ -> DFE}

Candidate keys: AD, DB, DE, DF

Prime Attribute: A,B,D,E,F

Non Prime Attribute: C,G,H

Example

- Given **R**(E-ID, E-NAME, E-CITY, E-STATE)
  - FD **E-ID->E-NAME** holds because for each E-ID, there is a unique value of E-NAME.
  - FD **E-ID->E-CITY** and **E-CITY->E-STATE** also holds.
  - FD E-NAME->E-ID **does not hold** because E-NAME 'John' is not uniquely determining E-ID. There are 2 E-IDs corresponding to John (E001 and E003).

- **FDs** = { E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE }

Answer

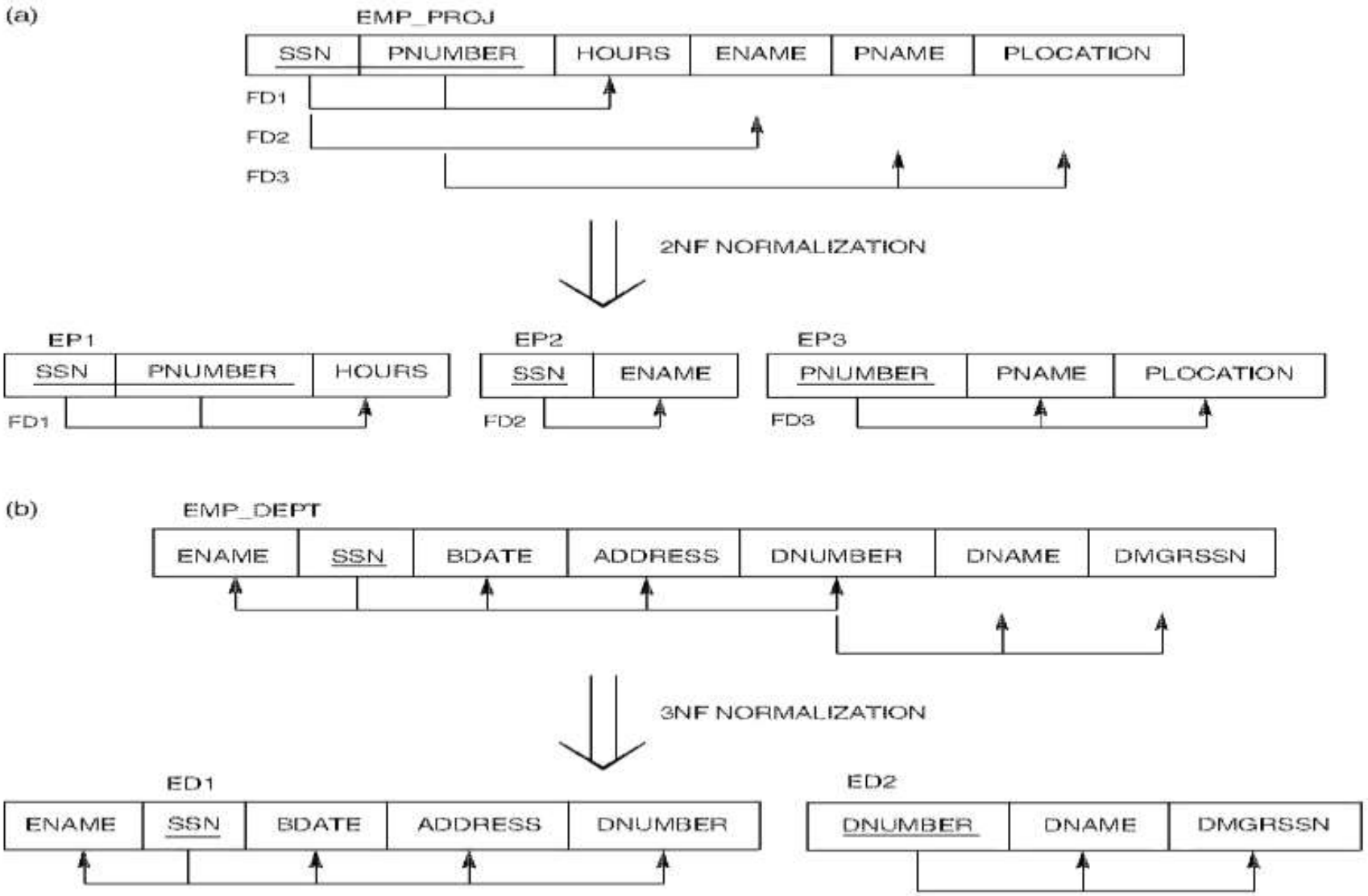
- **Attribute closure:**
  - (E-ID)+ = **{E-ID, E-NAME,E-CITY,E-STATE}**
  - (E-ID,E-NAME)+ = **{E-ID, E-NAME,E-CITY,E-STATE}**
  - (E-ID,E-CITY)+ = **{E-ID, E-NAME,E-CITY,E-STATE}**
  - (E-ID,E-STATE)+ = **{E-ID, E-NAME,E-CITY,E-STATE}**
  - (E-ID,E-CITY,E-STATE)+ = **{E-ID, E-NAME,E-CITY,E-STATE}**
  - (E-NAME)+ = {E-NAME}
  - (E-CITY)+ = {E-CITY,E-STATE}

Answer

- **Superkey set**
  - (E-ID)
  - (E-ID, E-NAME)
  - (E-ID, E-CITY)
  - (E-ID, E-STATE)
  - (E-ID, E-CITY, E-STATE)

- These FDs give set of all attributes of relation EMPLOYEE.
- So all of these are super keys of relation.
- As shown above, (E-ID) is set of all attributes of relation and it is minimal.
- So E-ID will be candidate key.
- On the other hand (E-ID, E-NAME)+ also is set of all attributes but it is not minimal because its subset (E-ID)+ is equal to set of all attributes.
- So (E-ID, E-NAME) is not a candidate key.

Figure 10.10 Normalizing into 2NF and 3NF



Check following schema for normal form

order(orderno,customerno,name,address,orderdate,
(productno,description,quantity,unitprice)
)

Solution:

- 1NF

  ORDER(orderno,customerno,name,address,orderdate)

  ORDERLINE(orderno,productno,description,quantity, unitprice)

- FD
  1. Orderno->orderdate
  2. Customerno->name,address
  3. Orederno->customerno,name,address
  4. Productno->description,unitprice
  5. Productno,orderno -> quantity

Check for 2NF

FD4 is partial dependency

Sol:

- Order(oderno,customerno,name,address,orderdate)
- Orderline(orderno,productno,quantity)
- Product(productno,description,unitprice)

FD

1. Orderno->customerno
2. Customerno->name,address
3. Projectno->description,unitprice
4. Productno,orderno -> quantity

Check for 3NF

- FD1&2 transitive dependency
- Sol:
  - Order(oderno,customerno,orderdate)
  - Customer(Customerno,name,address)
  - Orderline(orderno,productno,quantity)
  - Product(productno,description,unitprice)

Examples

Question 1 Suppose you are given a relation $R = (A,B,C,D,E)$ with the following functional dependencies: $\{CE \rightarrow D, D \rightarrow B, C \rightarrow A\}$.

a. Find all candidate keys.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

c. If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

Answer.

a. The only key is $\{C, E\}$

b. The relation is in 1NF

c. Decompose into R1=(A,C) and R2=(B,C,D,E). R1 is in BCNF, R2 is in 2NF. Decompose R2 into, R21=(C,D,E) and R22=(B,D). Both relations are in BCNF.

Note: The above figure is now called Figure 10.11 in Edition 4

Examples

Question 2 Suppose you are given a relation R=(A,B,C,D,E) with the following functional dependencies: $\{BC \rightarrow ADE, D \rightarrow B\}$.

a. Find all candidate keys.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

c. If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

Question 10 Consider the relation R(V, W, X, Y, Z) with functional dependencies $\{Z \rightarrow Y, Y \rightarrow Z, X \rightarrow Y, X \rightarrow V, VW \rightarrow X\}$.

a) List the possible keys for relation R based on the functional dependencies above.

b) Show the closure for attribute X given the functional dependencies above.

c) Suppose that relation R is decomposed into two relations, R1(V, W, X) and R2(X, Y, Z). Is this decomposition a lossless decomposition? Explain your answer.

Note: The above figure is now called Figure 10.11 in Edition 4

4 General Normal Form Definitions (For Multiple Keys) (1)

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R

General Normal Form Definitions (2)

Definition:

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form** (**3NF**) if whenever a FD X -> A holds in R, then either:

    (a) X is a superkey of R, or

    (b) A is a prime attribute of R

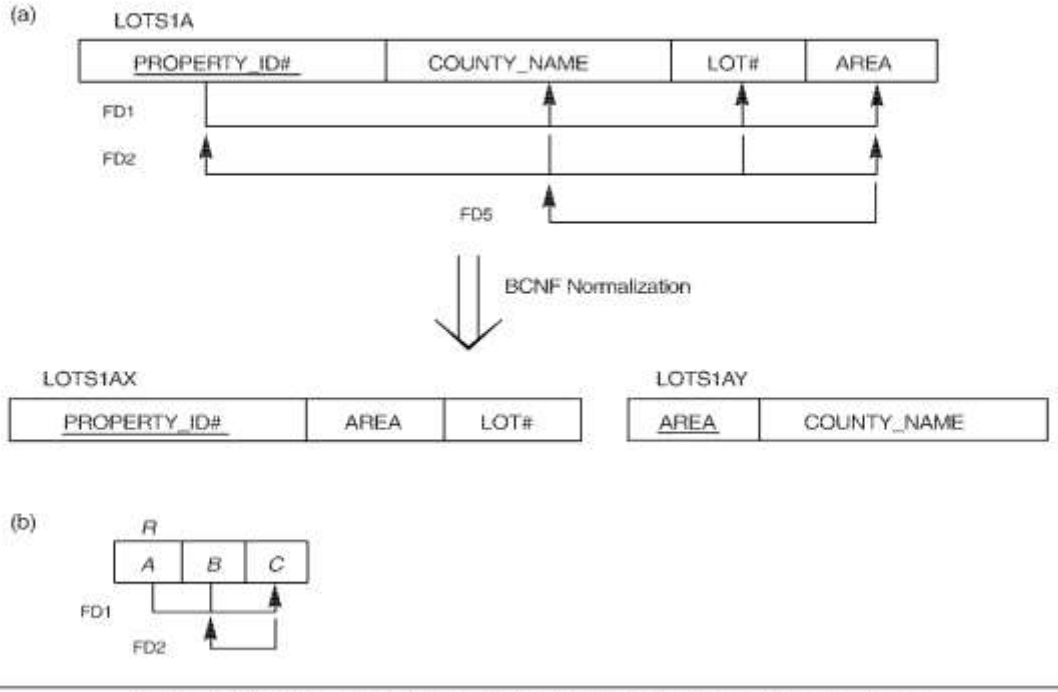
NOTE: Boyce-Codd normal form disallows condition (b) above

5 BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Figure 10.12 Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation *R* in 3NF but not in BCNF.



Note: The above figure is now called Figure 10.12 in Edition 4

Figure 10.13 a relation TEACH that is in 3NF but not in BCNF

Figure 14.13 A relation TEACH that is in 3NF but 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

Note: The above figure is now called Figure 10.13 in Edition 4

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:

  fd1: { student, course} -> instructor

  fd2: instructor -> course

- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 10.12 (b). So this relation is in 3NF but not in BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations. (See Algorithm 11.3)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  1. {student, instructor} and {student, course}
  2. {course, instructor } and {course, student}
  3. {instructor, course } and {instructor, student}
- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join.(and hence has the non-additivity property).
- A test to determine whether a binary decomposition (decomposition into two relations) is nonadditive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.