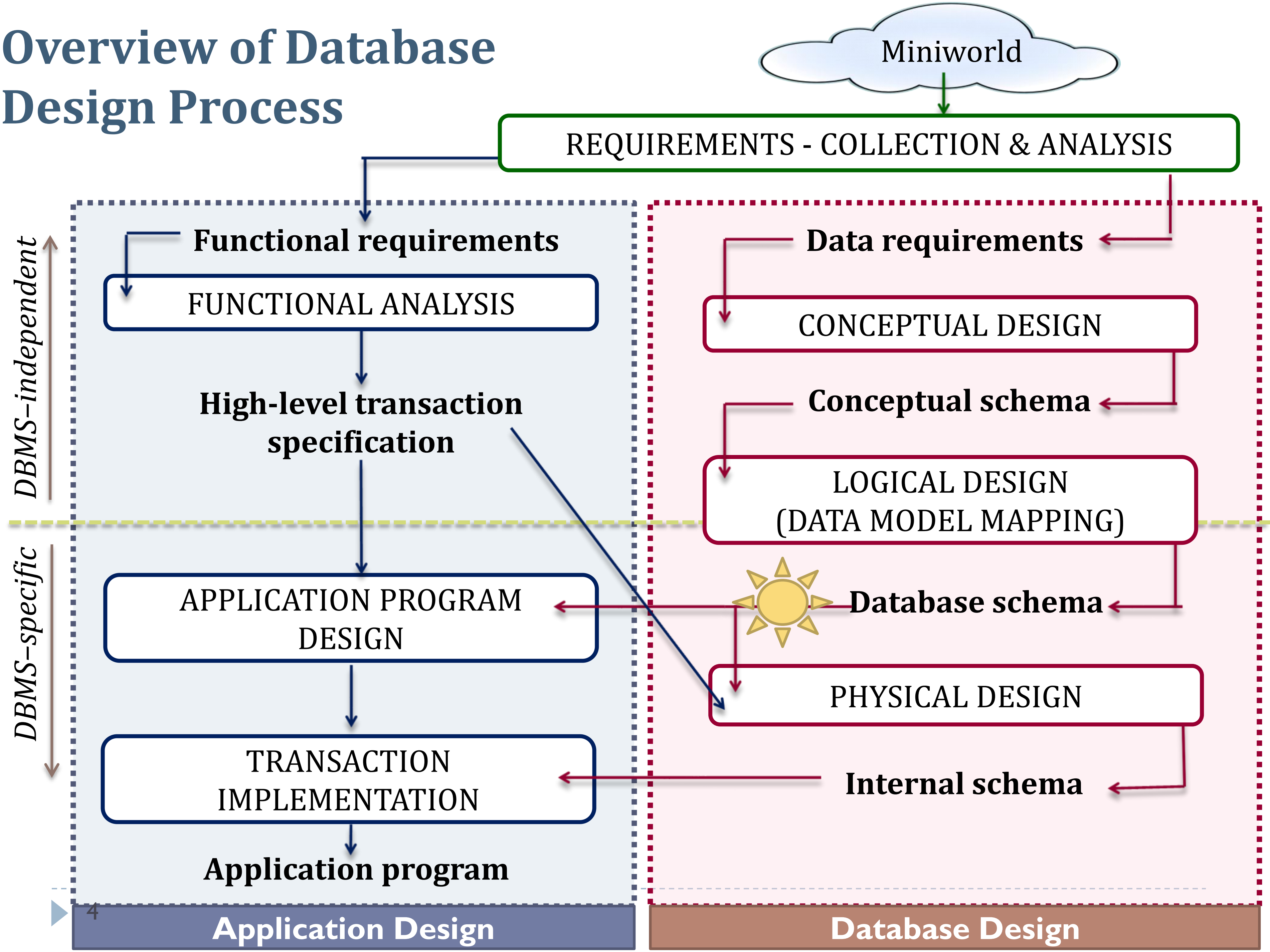
Functional Dependencies & Normalization for Relational DBs

Chapter 7

Contents

Contents

Overview of Database Design Process



Introduction

- Each relation schema consists of a number of attributes and the relational database schema consists of a number of relation schemas
- Attributes are grouped to form a relation schema
- Need some formal measure of why one grouping of attributes into a relation schema may be better than another

Introduction

- “Goodness” measures:
  - Redundant information in tuples
  - Update anomalies: modification, deletion, insertion
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples

Introduction

- **Redundant information** in tuples: the attribute values pertaining to a particular department (DNUMBER, DNAME, DMGRSSN) are repeated for every employee who works for that department.

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
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

Introduction

- **<u>Update anomalies</u>**: modification, deletion, insertion
  - Modification
    - As the manager of a dept. changes we have to update many values according to employees working for that dept.
    - Easy to make the DB ***<u>inconsistent</u>***

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
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

Introduction

- Deletion: if Borg James E. leaves, we delete his tuple and lose the existing of dept. 1, the name of dept. 1, and who is the manager of dept. 1

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
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

Introduction

- Insertion:
  - How can we create a department before any employees are assigned to it ?

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
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

Introduction

- Reducing the NULL values in tuples
  - Employees not assigned to any dept.: waste the storage space
  - Other difficulties: aggregation operations (e.g., COUNT, SUM) and joins

Introduction

- Disallowing the possibility of generating spurious tuples

EMP_PROJ (SSN, PNumber, Hours, EName, PName, PLocation)

EMP_LOCS (EName, PLocation)
EMP_PROJ1 (SSN, PNumber, Hours, PName, PLocation)

- Generation of invalid and spurious data during JOINS: PLocation is the attribute that relates EMP_LOCS and EMP_PROJ1, and PLocation is neither a primary key nor a foreign key in either EMP_LOCS or EMP_PROJ1

Introduction

- Disallowing the possibility of generating spurious tuples

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

Introduction

- Disallowing the possibility of generating spurious tuples

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

⋮

Introduction

- "Goodness" measures:
  - Redundant information in tuples
  - Update anomalies: modification, deletion, insertion
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples

☞ **Normalization**

Introduction

- Normalization helps DB designers determine the best relation schemas
  - A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes
  - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree
- It is based on the concept of normal form 1NF, 2NF, 3NF, BCNF, 4NF, 5NF
- It is a process which ensures that the data is structured in such a way that attributes are grouped with the PK. Attributes that do not directly depend on PK may be extracted to form a new relation

Contents

Functional Dependencies (FDs)

- **Definition of FDs**
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs

Functional Dependencies (FDs)

- 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

$X \rightarrow Y$

Functional Dependencies (FDs)

- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they must have the same value for Y
- For any two tuples t_1 and t_2 in any relation instance r(R):
  If $t_1[X]=t_2[X]$, then $t_1[Y]=t_2[Y]$
- $X \rightarrow Y$ in R specifies a constraint on all relation instances r(R)
- Examples:
  - 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

Functional Dependencies (FDs)

- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with $t_1[K]=t_2[K]$)

Functional Dependencies (FDs)

- Definition of FDs
- **Direct, indirect, partial dependencies**
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs

Functional Dependencies (FDs)

- **Direct dependency** (fully functional dependency): All attributes in a R must be fully functionally dependent on the primary key (or the PK is a determinant of all attributes in R)

  SSN → {Name, BDate, Address, DNO}

EMPLOYEE

<u>**SSN**</u>	Name	BDate	Address	DNO

Functional Dependencies (FDs)

- **Indirect dependency** (transitive dependency): Value of an attribute is not determined directly by the primary key

  DNO → DName

EMP_DEPT

<u>**SSN**</u>	Name	BDate	Address	DNO	DName

Functional Dependencies (FDs)

- **Partial dependency**
  - **Composite determinant** - more than one value is required to determine the value of another attribute, the combination of values is called a composite determinant

    {SSN, PNumber} in EMP_PROJ
  - **Partial dependency** - if the value of an attribute does not depend on an entire composite determinant, but only part of it, the relationship is known as the partial dependency

    SSN → EName , Pnumber → {PName, PLocation}

EMP_PROJ

<u>SSN</u>	<u>PNumber</u>	Hours	EName	PName	PLocation

Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- **Inference Rules for FDs**
- Equivalence of Sets of FDs
- Minimal Sets of FDs

Functional Dependencies (FDs)

- 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 $Y \subseteq X$, then $X \rightarrow Y$
  - **IR2. (Augmentation)** If $X \rightarrow Y$, then $XZ \rightarrow YZ$

    (Notation: XZ stands for $X \cup Z$)
  - **IR3. (Transitive)** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

Functional Dependencies (FDs)

- 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)

Functional Dependencies (FDs)

- 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

Functional Dependencies (FDs)

- **Algorithm 16.1.** 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^+);

Exercise

- Consider a relation $R(A, B, C, D, E)$ with the following dependencies F:

  (1) $AB \rightarrow C$,

  (2) $CD \rightarrow E$,

  (3) $DE \rightarrow B$

- Find $\{A, B\}^+$, $\{A, B, D\}^+$

Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- **Equivalence of Sets of FDs**
- Minimal Sets of FDs

Functional Dependencies (FDs)

- Two sets of FDs F and G are **equivalent** if $\mathbf{F^+ = G^+}$
- Definition:
  - F **covers** G if $\mathbf{G^+ \subseteq F^+}$
  - F and G are **equivalent** if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

Exercise

- Prove that two following sets of FDs are **equivalent:**
  - F= {$A \rightarrow C$, $AC \rightarrow D$, $E \rightarrow AD$, $E \rightarrow H$}
  - G = {$A \rightarrow CD$, $E \rightarrow AH$}

Functional Dependencies (FDs)

- A set of FDs is minimal if it satisfies the following conditions:
  - Every dependency in F has a single attribute for its RHS.
  - We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
  - 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

Functional Dependencies (FDs)

- **Algorithm 16.2.** Finding a Minimal Cover F for a Set of Functional Dependencies E
- **Input:** A set of functional dependencies E.

1. Set $F := E$.

2. Replace each functional dependency $X \rightarrow \{A1, A2, ..., An\}$ in F by the n functional dependencies $X \rightarrow A1$, $X \rightarrow A2$, ..., $X \rightarrow An$.

3. For each functional dependency $X \rightarrow A$ in F

for each attribute B that is an element of X

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

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

4. For each remaining functional dependency $X \rightarrow A$ in F

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

then remove $X \rightarrow A$ from F.

Exercise

- Give set of FDs:

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

- Find the minimal cover of E.

Functional Dependencies (FDs)

- Every set of FDs has at least one equivalent minimal set
- 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**

Contents

Normalization

- **Normalization**: The process of **decomposing** unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- Normal form: Using keys and FDs of a relation to certify whether a relation schema is in a particular normal form
- Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The database designers **need not** normalize to the highest possible normal form (3NF, BCNF or 4NF)

Normalization

- There are two important properties of decompositions:

  1) non-additive or losslessness of the corresponding join

  2) preservation of the functional dependencies

- Note that property (1) is extremely important and cannot be sacrificed. Property (2) is less stringent and may be sacrificed

Normalization

- **Superkey** of R: A set of attributes **SK** of **R** such that no two tuples in any valid relation instance r(R) will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK].

- **Key** of R: A **"minimal" superkey**; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

- If K is a **key** of R, then K **functionally determines all attributes** in R.

Normalization

- Two new concepts:
  - A **Prime attribute** must be a member of some candidate key
  - A **Nonprime attribute** is not a prime attribute: it is not a member of any candidate key

Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations

Normalization

- **First normal form (1NF):** there is only one value at the intersection of each row and column of a relation - no set valued attributes in 1NF

  → Disallows composite attributes, multivalued attributes, and **nested relations**
- To be part of the formal definition of a relation in the basic (flat) relational model
- The only attribute values permitted by 1NF are single **atomic** (or **indivisible**) **values**

1NF

(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}

1NF Normalization

(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

1NF

(a)

EMP_PROJ

Ssn	Ename	Projs	
		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

1NF

(c)

1NF Normalization

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

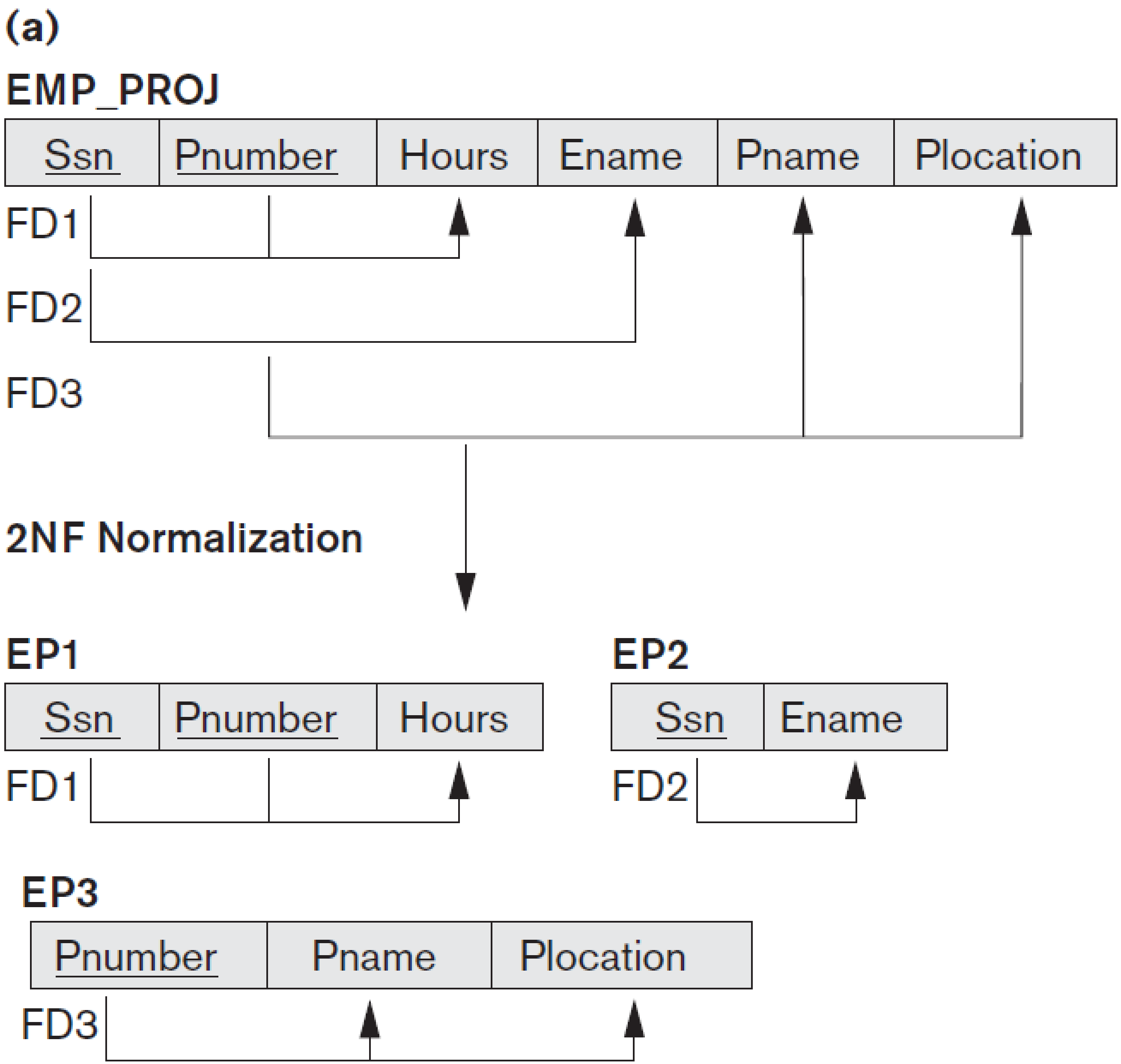
Normalization

- 1NF and dependency problems
- **2NF – solves partial dependency**
- 3NF – solves indirect dependency
- BCNF – well-normalized relations

Normalization

- Second normal form (2NF) – all nonprime attributes must be fully functionally dependent on the primary key
- 2NF solves partial dependency problem in 1NF
- **2NF normalized:** 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.

2NF



(a)
EMP_PROJ
Ssn
Pnumber
Hours
Ename
Pname
Plocation
FD1
FD2
FD3
2NF Normalization
EP1
Ssn
Pnumber
Hours
FD1
EP2
Ssn
Ename
FD2
EP3
Pnumber
Pname
Plocation
FD3

2NF

EMP_DEPT

EMP_DEPT

➢**Problem with 2NF**

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
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

Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- **3NF – solves indirect dependency**
- BCNF – well-normalized relations

3NF

- A relation schema R is in **third normal form** (**3NF**) if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key

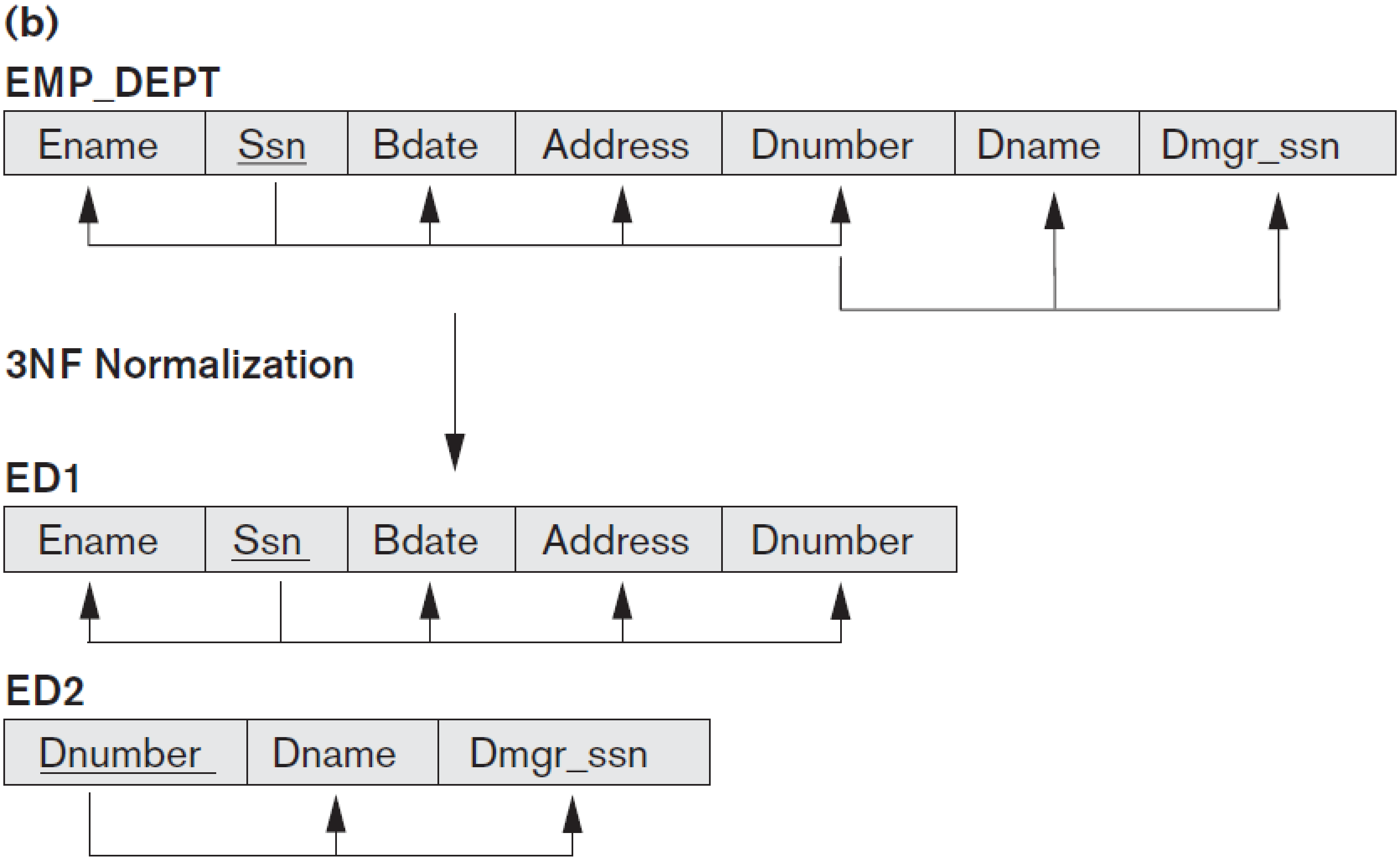
- **NOTE:**
  - In X → Y and Y → Z, with X as the primary key, we consider this a problem only if Y is <u>not</u> a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

  - E.g., Consider EMP (SSN, Emp#, Salary).
  - Here, SSN → Emp# →Salary and Emp# is a candidate key

3NF

- 3NF solves indirect (transitive) dependencies problem in 1NF and 2NF.
- **3NF normalized:** identify all transitive dependencies and each transitive dependency will form a new relation, with non-prime attributes participating in the transitive dependency and the attribute which determines others as the attributes for the new relation.

3NF


(b)
EMP_DEPT
Ename
Ssn
Bdate
Address
Dnumber
Dname
Dmgr_ssn
3NF Normalization
ED1
Ename
Ssn
Bdate
Address
Dnumber
ED2
Dnumber
Dname
Dmgr_ssn

SUMMARY OF NORMAL FORMS based on Primary Keys

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).

General Normal Form Definitions

- The above definitions consider the primary key only.
- The following more general definitions take into account relations with multiple candidate keys.

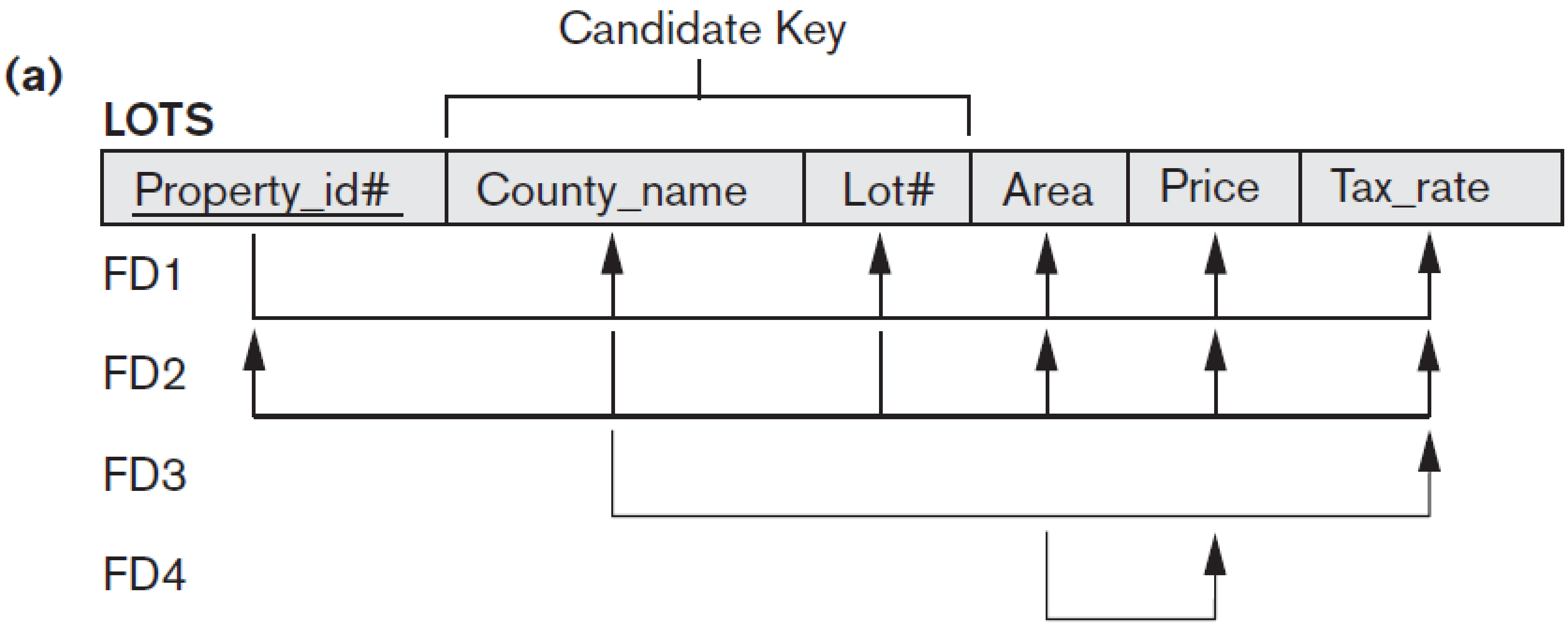
General Normal Form Definitions

- 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.
- 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

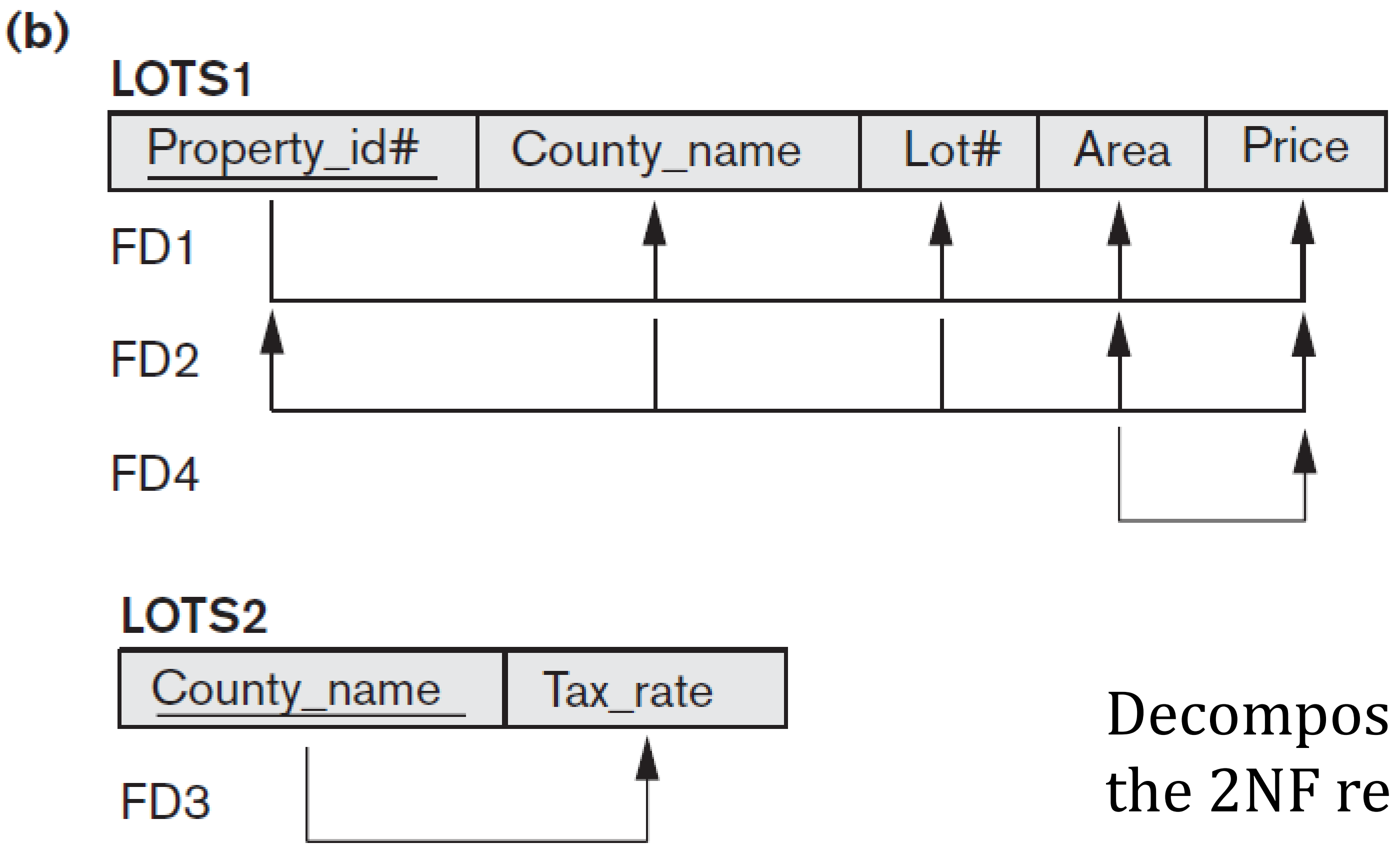
  (b) A is a prime attribute of R

General Normal Form Example



The LOTS relation with its functional dependencies.

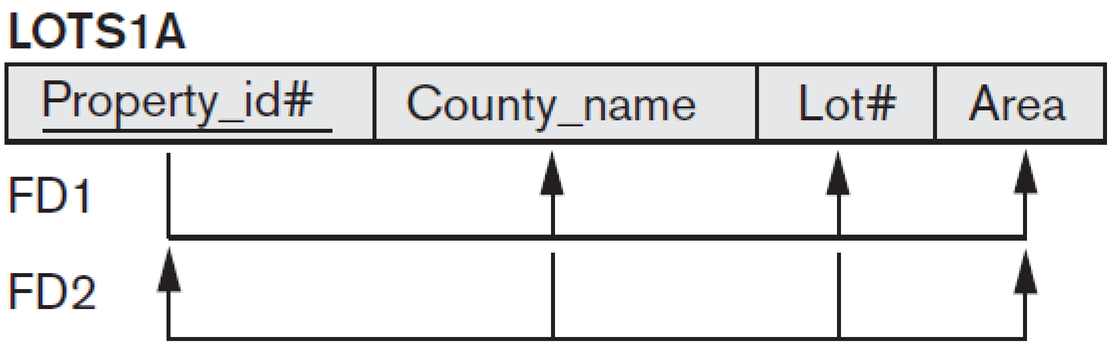
General Normal Form Example

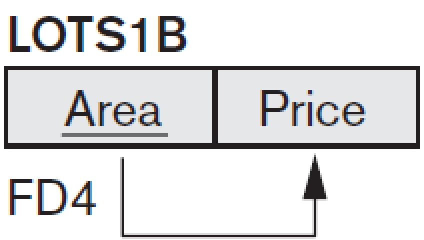


Decomposing into the 2NF relations

General Normal Form Example

(c)





Decomposing LOTS1 into the 3NF relations

Normalization

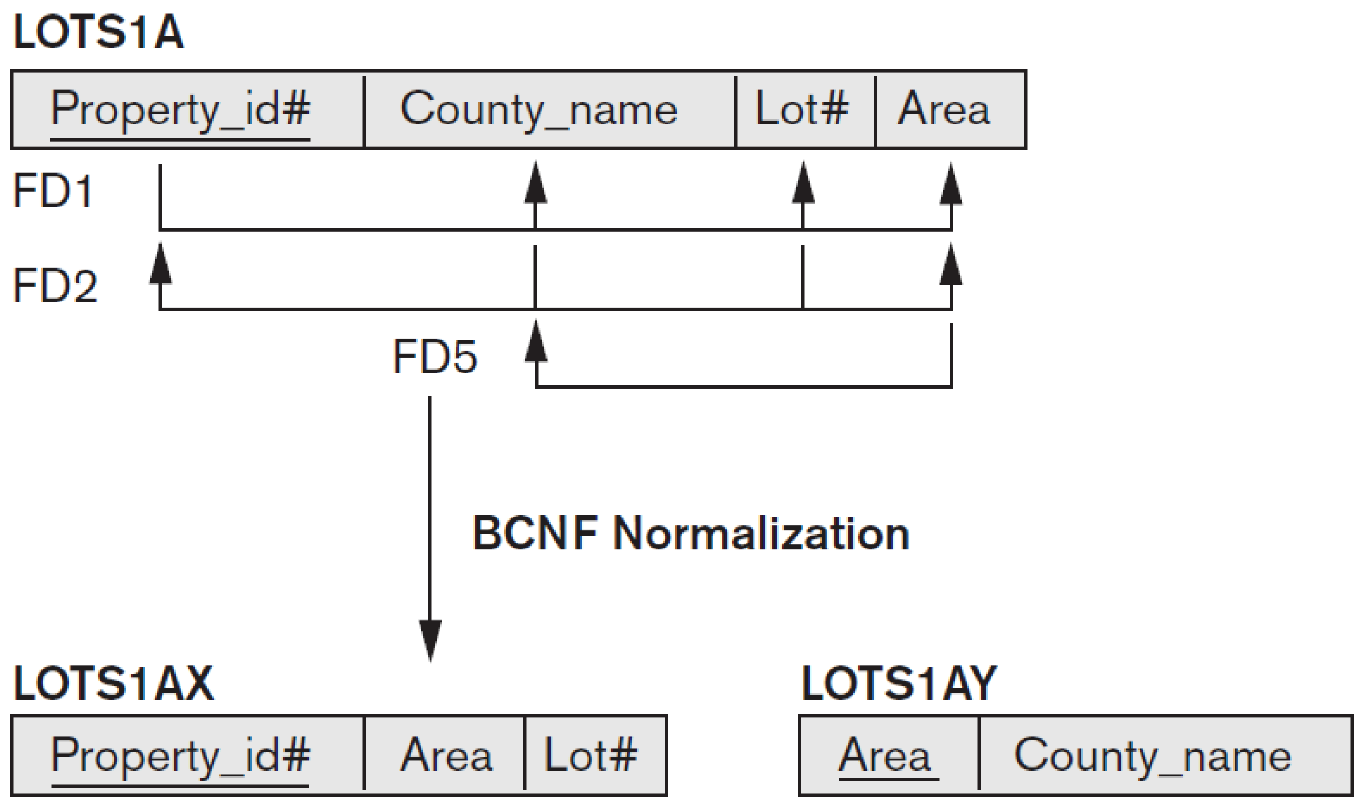
- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- **BCNF – well-normalized relations**

BCNF

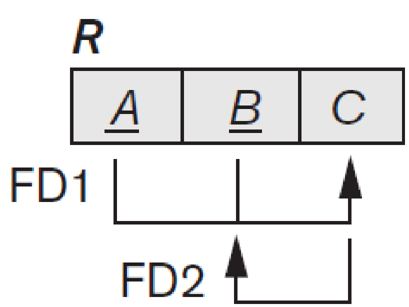
- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R

BCNF

(a)



(b)



(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.

BCNF

- TEACH (Student, Course, Instructor)
  - FD1: {Student, Course} → Instructor
  - FD2: Instructor → Course

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

BCNF

- Three possible pairs:
  1. {<u>Student, Instructor</u>} and {<u>Student, Course</u>}
  2. {Course, <u>Instructor</u>} and {<u>Course, Student</u>}
  3. {<u>Instructor</u>, Course} and {<u>Instructor, Student</u>}

- All three decompositions *lose the functional dependency FD1*. The desirable decomposition of those just shown is 3 because it will not generate spurious tuples after a join.

Notes & Suggestions

- [1], chapter 15:
  - 4NF: based on *multivalued dependency* (MVD)
  - 5NF: based on join dependency
    - Such a dependency is very difficult to detect in practice and therefore, normalization into 5NF is considered very rarely in practice
  - Other normal forms & algorithms
  - ER modeling: top-down database design
    - Bottom-up database design ??
- [1], chapter 16: Properties of Relational Decompositions

Contents

Dependency-Preserving Decomposition into 3NF Schemas

- **Algorithm 16.4.** Relational Synthesis into 3NF with Dependency Preservation
- **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 16.2);

  2. For each left-hand-side *X* of a functional dependency that appears in *G*, create a relation schema in *D* with attributes $\{X \cup \{A_1\} \cup \{A_2\} \dots \cup \{A_k\}\}$, where $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_k$ are the only dependencies in *G* with *X* as the left-hand-side (*X* is the key of this relation);

  3. Place any remaining attributes (that have not been placed in any relation) in a single relation schema to ensure the attribute preservation property.

Nonadditive Join Decomposition into BCNF Schemas

- **Algorithm 16.5.** 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 $X \rightarrow Y$ in *Q* that violates BCNF;

  replace *Q* in *D* by two relation schemas $(Q - Y)$ and $(X \cup Y)$;

  } ;

Dependency-Preserving and Nonadditive (Lossless) Join Decomposition into 3NF Schemas

- **Algorithm 16.6.** 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 16.2).

  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 $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_k$ 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.

  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

Dependency-Preserving and Nonadditive (Lossless) Join Decomposition into 3NF Schemas

- Algorithm 16.6:
  - Preserves dependencies.
  - Has the nonadditive join property.
  - Is such that each resulting relation schema in the decomposition is in 3NF.

- It is preferred over Algorithm 16.4.

Contents

Key-finding algorithm (1)

By Elmasri and Navathe

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

Output: *a key K of R*

1. Set K to contain all attributes in R
2. For each attribute A in K {

   compute $(K – A)^+$ with respect to F;

   if $(K – A)^+$ contains all attributes in R, then

   set K := K – {A}

   };

Key-finding algorithm (1)

By Elmasri and Navathe

- In algorithm (1), we start by setting K to all the attributes of R; we then remove one attribute at a time and check whether the remaining attributes still form a superkey.
- The algorithm (1) determines only **one key** out of the possible candidate keys for R; the key returned depends on the order in which attributes are removed from R in step 2.

Key-finding algorithm (2)

By Hossein Saiedian & Thomas Spencer

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

Output: *all candidate keys of R*

Let:

- $\boldsymbol{U}$ contain **all** attributes of R
- $\boldsymbol{U_L}$ contain attributes of R that occur **only** on the **left-hand side** of FDs in F
- $\boldsymbol{U_R}$ contain attributes of R that occur **only** on the **right-hand side** of FDs in F
- $\boldsymbol{U_B}$ contain attributes of R that occur on **both sides** of FDs in F

Note:

- $\boldsymbol{U_L} \cap \boldsymbol{U_R} = \phi$, $\boldsymbol{U_L} \cap \boldsymbol{U_B} = \phi$ and $\boldsymbol{U_R} \cap \boldsymbol{U_B} = \phi$
- $\boldsymbol{U_L} \cup \boldsymbol{U_R} \cup \boldsymbol{U_B} = \boldsymbol{U}$
- For every attribute $\boldsymbol{A} \in \boldsymbol{U}$, if $\boldsymbol{A} \in \boldsymbol{U_L}$, then $\boldsymbol{A}$ **must be** part of every candidate key of **R**.
- For every attribute $\boldsymbol{A} \in \boldsymbol{U}$, if $\boldsymbol{A} \in \boldsymbol{U_R}$, then $\boldsymbol{A}$ will **not** be part of any candidate key of **R.**

Key-finding algorithm (2)

By Hossein Saiedian & Thomas Spencer

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

Output: *all candidate keys of R*

1. Determine $\boldsymbol{U_L}$, $\boldsymbol{U_R}$ and $\boldsymbol{U_B}$
2. If $\boldsymbol{U_L}^+ = \boldsymbol{U}$ under F, then $\boldsymbol{U_L}$ forms the only key of R and the algorithm stops here.

   Else: move to step 3 // $\boldsymbol{U_L}^+ \neq \boldsymbol{U}$ under F
3. Consider every subsets $\boldsymbol{U_{Bi}}$ of $\boldsymbol{U_B}$: $\boldsymbol{U_{Bi}} \subset \boldsymbol{U_B}$

   For each $\boldsymbol{U_{Bi}}$, if $(\boldsymbol{U_L} \cup \boldsymbol{U_{Bi}})^+ = \boldsymbol{U}$ under F, then $\boldsymbol{K_i} = (\boldsymbol{U_L} \cup \boldsymbol{U_{Bi}})$ is a candidate key of R (*)

(*) *If* $\boldsymbol{K_i} = (\boldsymbol{U_L} \cup \boldsymbol{U_{Bi}})$ *is a candidate key of R, then we need not to check* $\boldsymbol{U_{Bj}} \subset \boldsymbol{U_b}$ *where* $\boldsymbol{U_{Bi}} \subset \boldsymbol{U_{Bj}}$

Key-finding algorithm (2)

By Hossein Saiedian & Thomas Spencer

- A simple categorization of attributes into the sets U_L, U_R and U_B allows to distinguish between those attributes that will participate in the candidate keys of a relational database schema and those that do not.
- The algorithm (2) finds all candidate keys.


Q
A

Exercise 1

Consider the universal relation R = {A, B, C, D, E, F} and the set of functional dependencies:

1) *A, B, C -> E, F*
2) *B -> E*
3) *E -> D*

What is the key for *R?* Decompose R into 2NF, 3NF, and BCNF relations.

Exercise 2

Consider the universal relation R = {*A, B, C, D, E, F*} and the set of functional dependencies:

1) $A, D \rightarrow B$
2) $A, B \rightarrow E$
3) $C \rightarrow D$
4) $B \rightarrow C$
5) $A, C \rightarrow F$

What is the key for *R?* Decompose *R* into 2NF, then 3NF relations.

Exercise 3

Consider the universal relation R = {*A, B, C, D, E, F*} and the set of functional dependencies:

1) $A \rightarrow B$
2) $C \rightarrow A, D$
3) $A, F \rightarrow C, E$

What is the key for *R?* Decompose *R* into 2NF, 3NF, and BCNF relations.

Exercise 4

Consider the universal relation R = {*A, B, C, D, E, F, G, H, I, J*} and the set of functional dependencies:

1) $A, B \rightarrow C$
2) $A \rightarrow D, E$
3) $B \rightarrow F$
4) $F \rightarrow G, H$
5) $D \rightarrow I, J$

What is the key for *R?* Decompose *R* into 2NF, 3NF, and BCNF relations.

Exercise 5

Consider the following relation books:

BOOK *(Book_title, Author_name, Book_type, List_price, Author_affil, Publisher)*

Suppose the following dependencies exist:

1) *Book_title → Publisher, Book_type*
2) *Book_type → List_price*
3) *Author_name → Author_affil*

a. What normal form is the relation in?
b. Apply normalization until you cannot decompose the relations further.

Exercise 6

- Consider the relation:

  BOOK (Book_Name, Author, Edition, Year)

- Based on a common-sense understanding of the data, what are the possible candidate keys of this relation?

Book_Name	Author	Edition	Copyright_Year
DB_fundamentals	Navathe	4	2004
DB_fundamentals	Elmasri	4	2004
DB_fundamentals	Elmasri	5	2007
DB_fundamentals	Navathe	5	2007