Relational Database Design

Functional Dependencies and Normalization for Relational Databases

October 2023

Normalization of Relations

- **Normalization:**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.
  - Concept built around the concept of normal form.
- **Normal form:**
  - A relation is said to be in a particular normal form if it satisfies some condition.

Normalized Design

- **Produce a set of relations which:**
  - Eliminate redundancy of non-key attributes.
  - Uses foreign keys to express relationships.
  - Has good update properties.
  - Satisfy referential and entity integrity.
- **Created through "normalization" process**
  - Successive decomposition of relations.
  - Create relations exhibiting certain properties.

Normal Forms

- Normal forms are based on FDs.
- Data normalized to,
  - Minimize redundancies
  - Minimize anomalies
- Relations are decomposed to normalize.
- Concerns with decomposition;
  - Loss-less join or nonadditive join property
  - Dependency preserving property

Normal Forms

- First Normal Form
  - Domain of an attribute must include only **atomic** (simple, indivisible) values
  - Value of any attribute in a tuple must be a single value from domain of that attribute.
  - Now considered part of definition of a relation in relational model.
- 2NF, 3NF, BCNF
  - based on keys and FDs of a relation schema

Normalization of Relations

- 4NF -
  - based on keys, multi-valued dependencies
- 5NF
  - based on keys, join dependencies
- **Denormalization:**
  - The process of storing the join of higher normal form relations as a base relation — which is in a lower normal form

Remember: Definitions of Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[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.

Remember: Definitions of Keys

- 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.

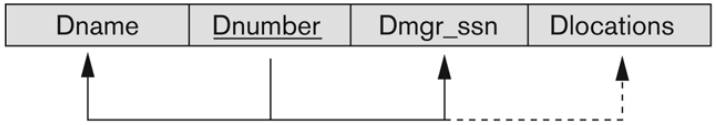
First Normal Form

- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic
- Considered to be part of the definition of relation

Normalization into 1NF

(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

Figure 10.8
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Normalization of nested relations into 1NF

(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, AliciaJ.	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

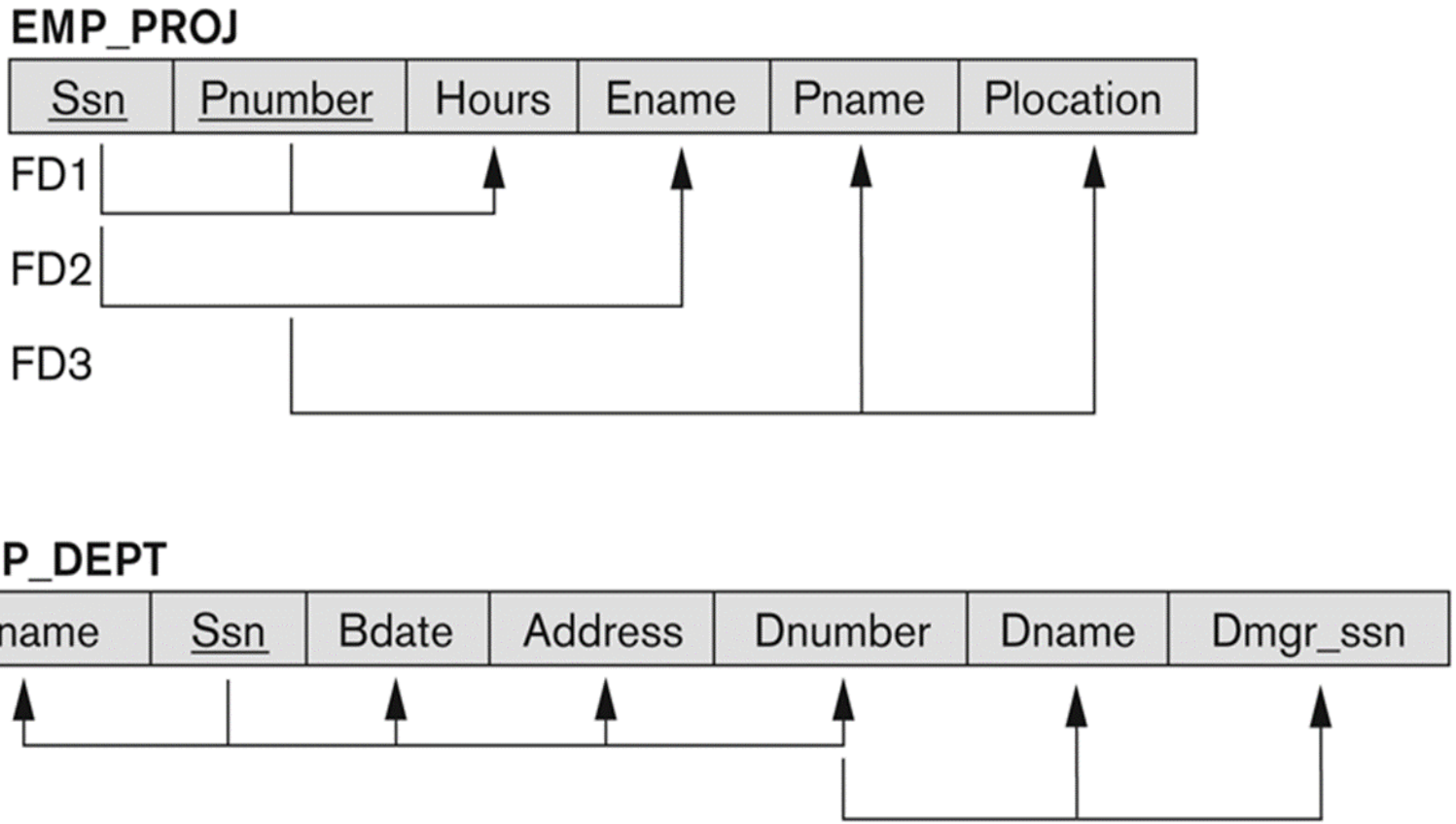
Ssn	Pnumber	Hours

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example 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.

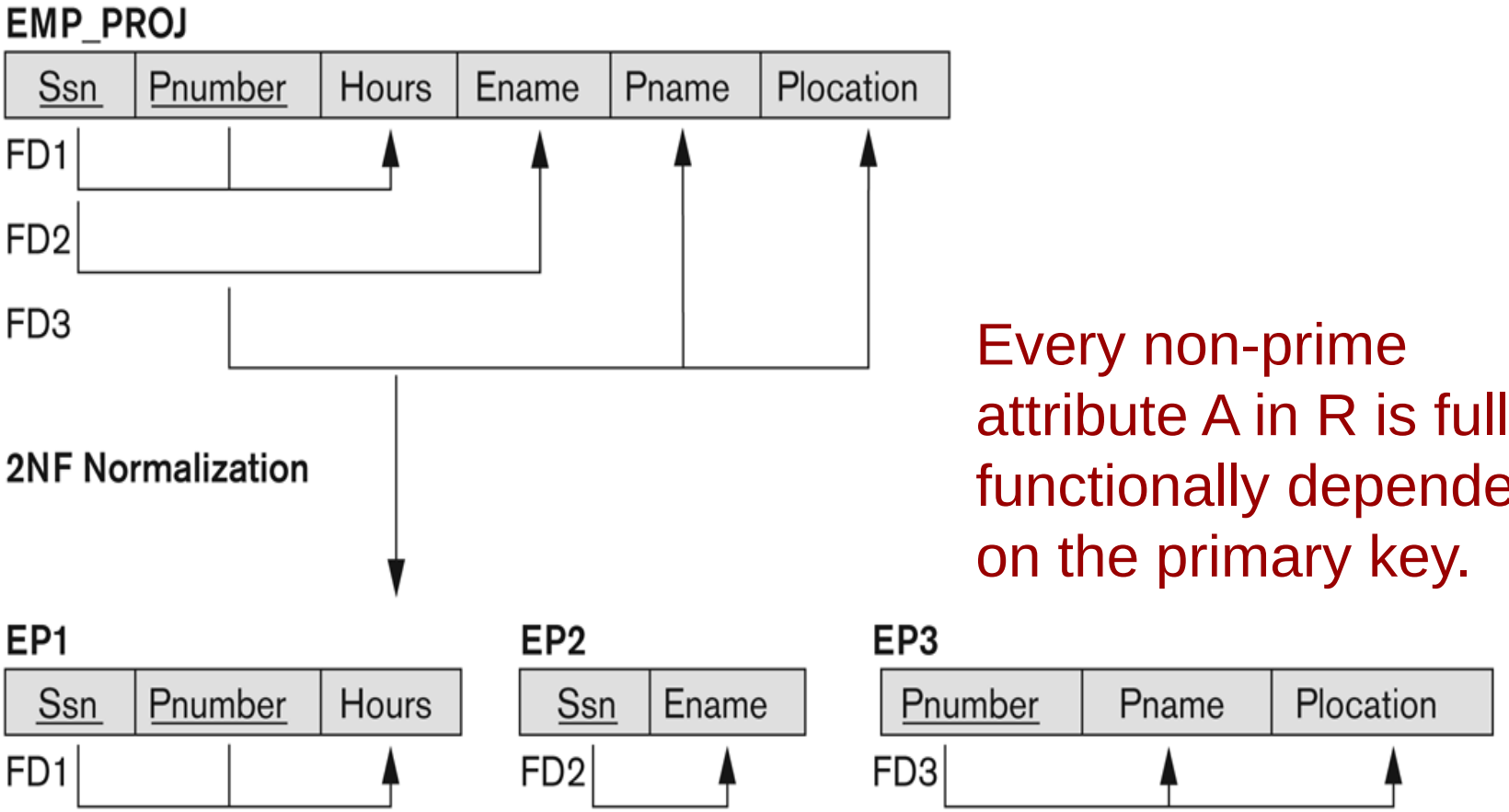
Second Normal Form

- Uses the concepts of **primary key, FDs.**
- 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.

Two Relational Schemas Suffering from Update Anomalies



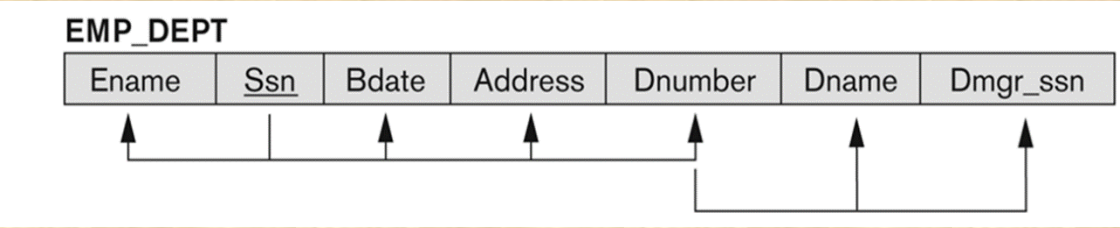
Normalizing into 2NF - Example



Every non-prime attribute A in R is fully functionally dependent on the primary key.

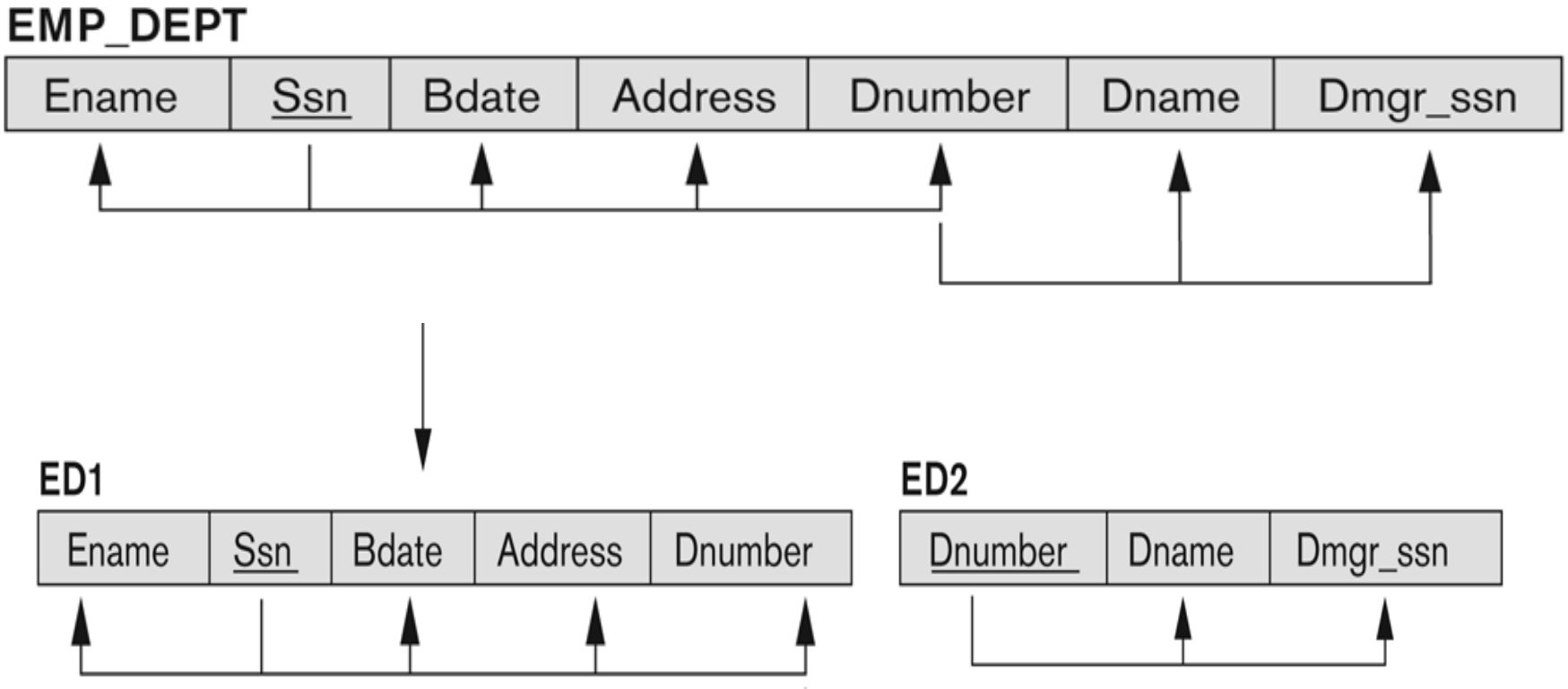
Third Normal Form

- **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
    - SSN → DNUMBER and DNUMBER → DMGRSSN
  - SSN -> ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN → X and X → ENAME

Normalizing into 3NF - Example



No non-prime attribute A in relation schema R is transitively dependent on the primary key.

Third Normal Form

- 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
- 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 .

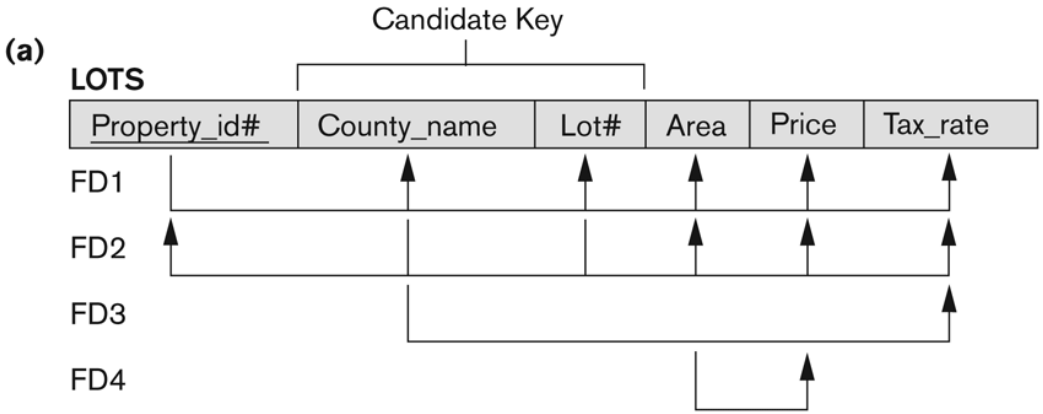
Normal Forms Defined Informally

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

General Normal Form Definitions (For Multiple Keys)

- Previous definitions consider the primary key only
- 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

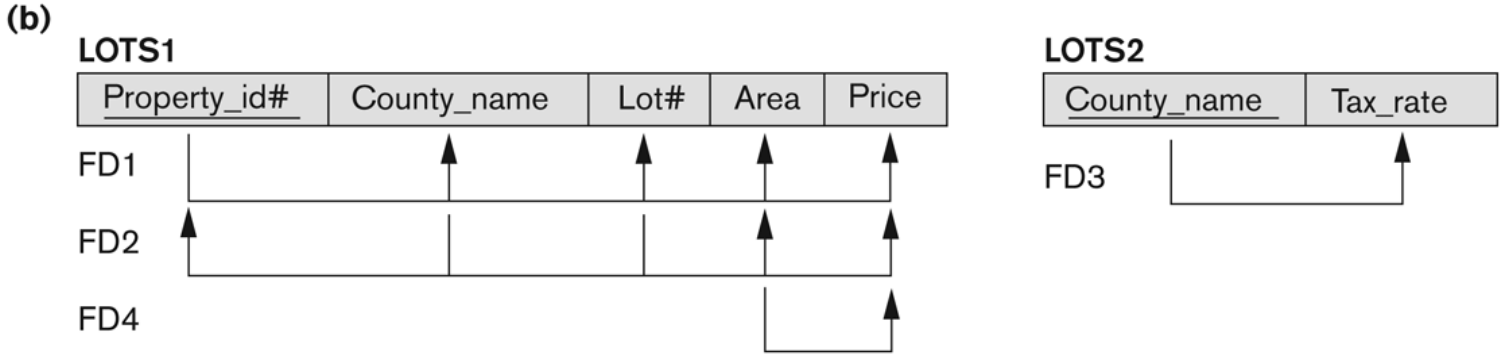
Another Example: 2NF



The LOTS relation with its FDs

FD3 violates the condition of 2NF

Decomposition into the 2NF relations LOTS1 and LOTS2

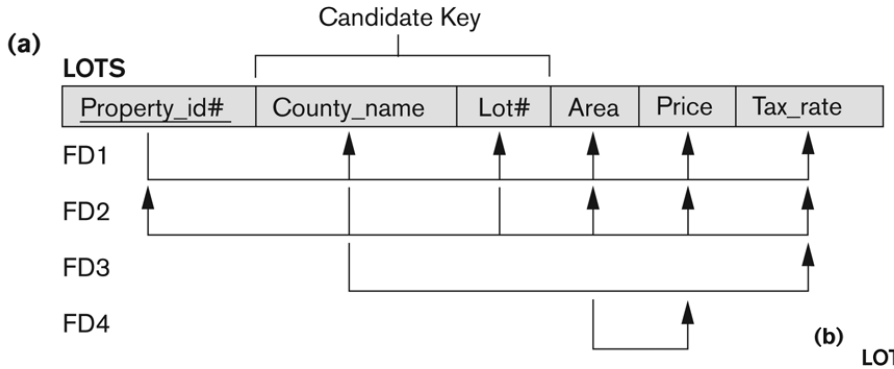


FD4 does not violates the condition of 2NF and is carried over to LOTS1

General Normal Form Definitions (For Multiple Keys)

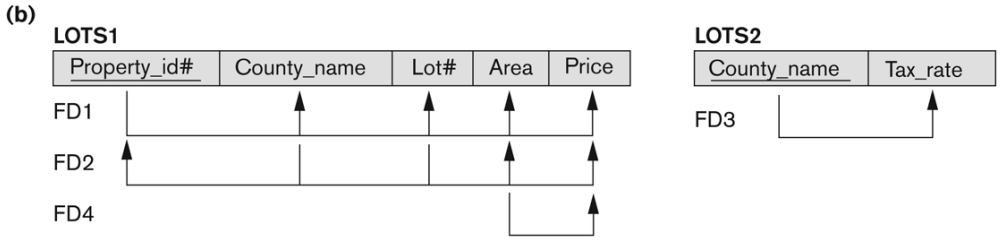
- A relation schema R is in **third normal form (3NF)** if every non-prime attribute of R is
  - Fully functionally dependent on every key of R; and
  - Nontransitively dependent on every key of R.

Another Example: 2NF and 3NF



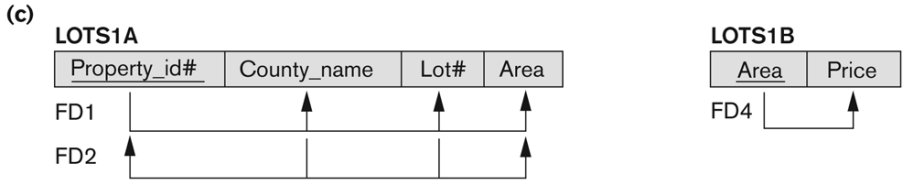
The LOTS relation with its FDs

Decomposition into the 2NF relations LOTS1 and LOTS2

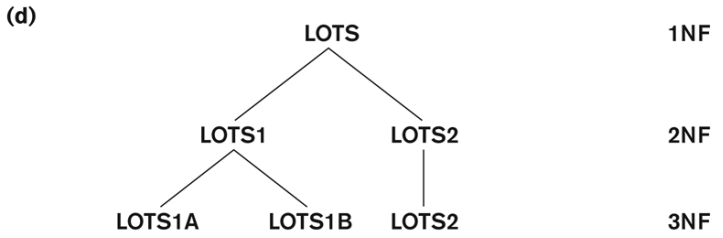


FD4 violates 3NF

Decomposing LOTS1 into the 3NF relations and LOTS1A and LOTS1B



Progressive normalization of lots LOTS1B

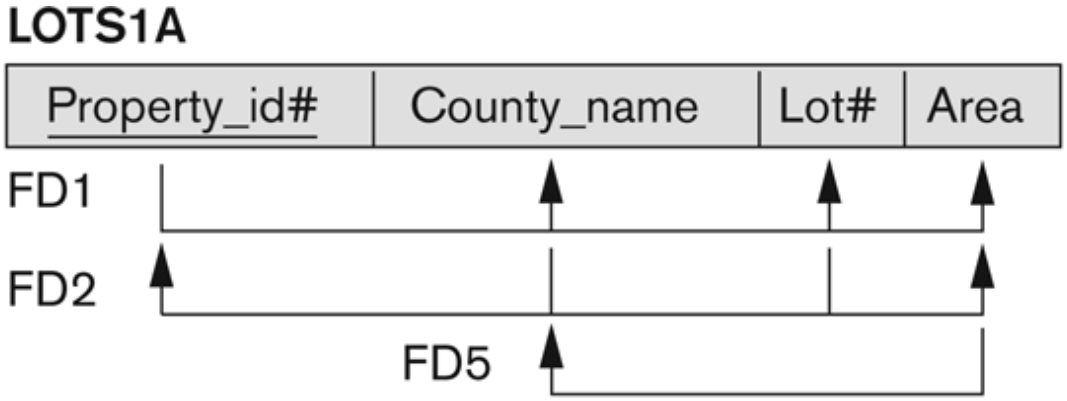


Boyce-Codd Normal Form

- An FD is trivial if RHS is subset of LHS.
- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** iff every nontrivial, left-irreducible FD has a candidate key as its determinant.
- Less formal definition
  - A relation is in BCNF iff the only determinants are candidate keys.
- In other words, there will always be arrows out of candidate keys.

BCNF- Example

LOTS1A with additional FD (FD5)



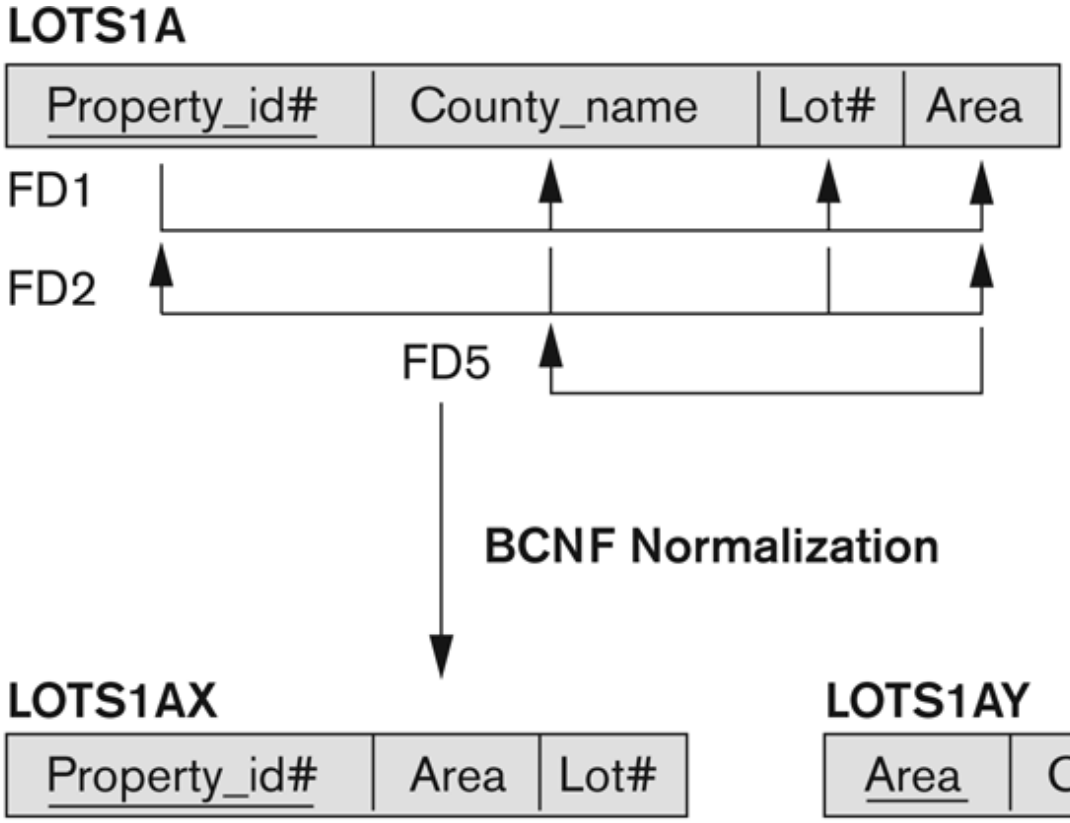
LOTS1A is still in 3NF because county_name is a prime attribute

A relation schema R is in third normal form (3NF) if every non-prime attribute of R is fully functionally dependent on every key of R; and nontransitively dependent on every key of R.

It does not talk about prime attributes.

BCNF- Example

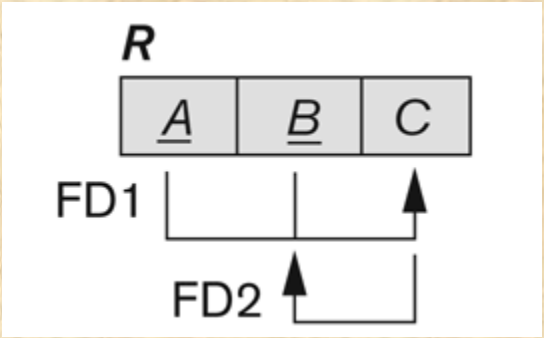
LOTS1A with additional FD (FD5)



LOTS1A is still in 3NF because county_name is a prime attribute

Boyce-Codd Normal Form

- 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

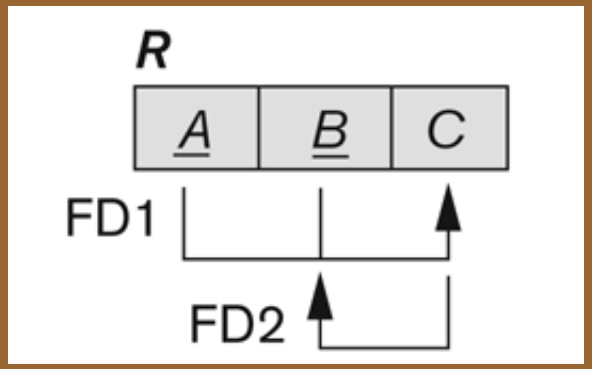


BCNF - Another Example

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

A relation that is in 3NF but not in BCNF



- Two FDs exist in the relation TEACH:
  - fd1: { student, course} → instructor
  - fd2: instructor → course
- {student, course} : candidate key

BCNF - Another Example

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

fd1: { student, course} -> instructor
fd2: instructor -> course

Which decomposition should be followed ?

- Three possible decompositions for this relation
  - {student, instructor} and {student, course}
  - {course, instructor } and {course, student}
  - {instructor, course } and {instructor, student}

BCNF - Another Example

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

fd1: { student, course} -> instructor
fd2: instructor -> course

See that all three decompositions will lose fd1 and only the 3rd decomposition satisfies lossless join property

- Three possible decompositions for this relation
  - {student, instructor} and {student, course}
  - {course, instructor } and {course, student}
  - **{instructor, course } and {instructor, student}**

- Here we have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice lossless join property

Designing a Set of Relations

- **Goals:**
  - Lossless join/nonadditive property (a must)
  - Dependency preservation property

- **Algorithms are available**
  - To tests for general losslessness.
  - To decompose a relation into BCNF components by sacrificing the dependency preservation.

Multivalued Dependencies

- In many cases relations have constraints that can not be specified as FDs.
- Multivalued dependencies (MVDs) are a consequence of 1NF, which disallows an attribute in a tuple to have a set of values.
- A multivalued dependency occurs when a determinant determines a particular set of values
- Consider the problem when we have two/more multivalued independent attributes in the same relation schema ….

Multivalued Dependencies

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

The EMP relation with two MVDs:

ENAME —>> PNAME and

ENAME —>> DNAME.

This constraint is specified as a multivalued dependency on the EMP

- An employee may work on several projects and may have several dependents.
- The employee's projects and dependents are independent of one another.
- Constraint: a separate tuple to represent every combination of an employee's dependent and an employee's project.
- Need to repeat every value of one of the attributes with every value of the other attribute

Multivalued Dependencies

- An MVD A —>> B in R is a trivial MVD if
  - B is a subset of A, or AuB = R.
- A trivial MVD will hold in any relation state r of R
  - It is called trivial because it does not specify any significant or meaningful constraint on R.
- If we have nontrivial MVD in a relation, we may have to repeat values redundantly in the tuples
  - This redundancy is clearly undesirable.

EMP schema is in BCNF because no functional dependencies hold in EMP

Need to define a fourth normal form

Relations containing nontrivial MVDs tend to be all - key relations

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

Multivalued Dependencies

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

The EMP relation with two MVDs:

ENAME —>> PNAME and

ENAME —>> DNAME.

- Whenever two independent 1:N relationships A:B and A:C are mixed in the same relation, R(A, B, C) an MVD may arise
- MVDs are a generalization of FDs, in the sense that every FD is an MVD but the converse is not true.

Multivalued Dependencies and Fourth Normal Form - Example

(a) The EMP relation with two MVDs:
ENAME —>> PNAME and ENAME —>> DNAME

(a)

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(b)

EMP_PROJECTS

ENAME	PNAME
Smith	X
Smith	Y

EMP_DEPENDENTS

ENAME	DNAME
Smith	John
Smith	Anna

Fourth Normal Form

- Fagin's Theorem
  - A relation R with attributes A, B, and C, can be nonloss-decomposed into its two projections R1(A, B) and R2 (A, C) iff the MVD A —>> B|C holds in R.

- A relation R is in 4NF iff, whenever there exist an MVD in R, say A —>> B, then all attributes of R are also functionally dependent on A (i.e., A->x for all attributes x of R)

Two Projections Always?

- So far every relation was non-loss decomposable into two projections
  - is this always possible?

- **n-decomposable** relations

Example: Relation CTL

- Courses - Tutors - Levels (CTL)

Course	Tutor	Level
Databases	Usha	Level3
Databases	Mahesh	Level2
Programming	Usha	Level2
Databases	Usha	Level2

Example: Relation CTL

- Courses - Tutors - Levels (CTL)

Course	Tutor	Level
Databases	Usha	Level3
Databases	Mahesh	Level2
Programming	Usha	Level2
Databases	Usha	Level2

Two Attribute Projections of CTL

- Courses - Tutors - Levels (CTL)

Course	Tutor	Level
Databases	Usha	Level3
Databases	Mahesh	Level2
Programming	Usha	Level2
Databases	Usha	Level2

CT

Course	Tutor
Databases	Usha
Databases	Mahesh
Programming	Usha

TL

Tutor	Level
Usha	Level3
Mahesh	Level2
Usha	Level2

Two Attribute Projections of CTL

CT

Course	Tutor
Databases	Usha
Databases	Mahesh
Programming	Usha

Tutor	Level
Usha	Level3
Mahesh	Level2
Usha	Level2

TL

Join(CT, TL)

The join of any two projections is not CTL

Course	Tutor	Level
Databases	Usha	Level3
Databases	Usha	Level2
Databases	Mahesh	Level2
Programming	Usha	Level3
Programming	Usha	Level2

CTL is a 3-decomposable relation

Two Attribute Projections of CTL

- Courses - Tutors - Levels (CTL)

CT

Course	Tutor
Databases	Usha
Databases	Mahesh
Programming	Usha

TL

Tutor	Level
Usha	Level3
Mahesh	Level2
Usha	Level2

CL

Course	Level
Databases	Level3
Databases	Level2
Programming	Level2

3-Decomposable Relation

- Constraint: Let R be a degree 3 relation.

  IF (**a**, **b**, x) $\in$ R
  AND (**a**, y, **c**) $\in$ R
  AND (z, **b**, **c**) $\in$ R
  THEN (**a**, **b**, **c**) $\in$ R

- Constraint illustrated on the CTL relation

  IF tutor t1 teaches subject s1
  AND level l1 studies subject s1
  AND tutor t1 teaches level l1
  THEN tutor t1 teaches subject s1 for level l1

- Note: this constraint is **not** expressed in CTL

Join Dependencies and 5NF

- Very peculiar semantic constraint that is very difficult to detect in practice
  - Normalization into 5NF - rarely done in practice.
- Definition of Join Dependency:
  - A join dependency denoted by JD(R1, R2, .., Rn), specified on relation schema R, specifies a constraint on the states of R.
  - The constraint states that every legal state r of R should have a nonadditive join decomposition into R1, R2, .., Rn
  - i.e., for every such r we have

    $*(\pi_{R1}(r), \pi_{R2}(r), \ldots \pi_{Rn}(r)) = r$

Join Dependencies and 5NF

- MVD is a special case of a JD where n = 2.
- A relation schema R is in fifth normal form (5NF) iff every join dependency in R is implied by the candidate keys of R.
- Let R be a relation. Let A, B, ..., Z be arbitrary subsets of R's attributes. R satisfies the * (R1, R2, .., Rn) if and only if R is equal to the join of its projections on (R1, R2, .., Rn)
- Also called projection-join form (PJ/NF)

Let us take some problems….

- Consider a relation R(A,B,C,D,E) with functional dependencies:

  $A \rightarrow B$, $BC \rightarrow E$, and $D \rightarrow A$

  Find the key for R.

Let us take some problems….

- Consider a relation R(A,B,C,D,E) with functional dependencies:

  $A \rightarrow D$, $C \rightarrow AB$, and $DB \rightarrow E$

  Find the key for R.

Let us take some problems….

- Consider the following relation:

CAR_SALE (car#, date_sold, salesman#, commission%, discount)

Assume that a car may be sold by multiple salesmen, and hence {car#, salesman#} is the primary key.

Additional dependencies are date_sold → discount and

Salesman# → commission%.

Based on the given primary key, is the relation in 1NF, 2NF, or 3NF? Why or why not?

How would you successively normalize it completely?