Functional Dependencies and Normalization for Relational Databases

406.426 Design & Analysis of Database Systems

Jonghun Park

jonghun@snu.ac.kr

Dept. of Industrial Engineering

Seoul National University

outline

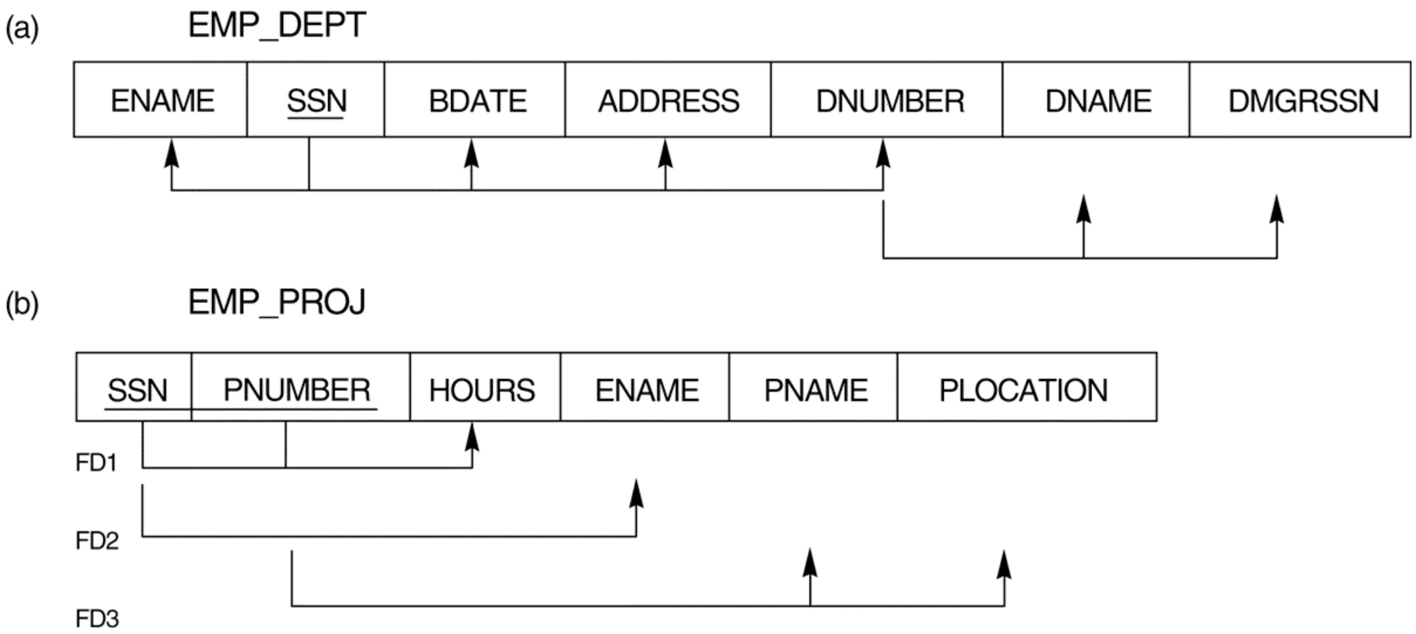
- informal design guidelines for relational databases
- functional dependencies (FDs)
- normal forms based on primary deys
- general normal form definitions (for multiple keys)
- BCNF (Boyce-Codd Normal Form)

informal measures of quality for relation schema

- **semantics** of the attributes
- reducing the **redundant values** in tuples
- reducing the **null values** in tuples
- disallowing the possibility of generating **spurious tuples**

semantics of the relation attributes

- guideline 1: Design a relation schema so that it is **easy to explain its meaning**. Do not combine attributes from multiple entity types and relationship types into a single relation. If **a relation schema corresponds to one entity type or one relationship type**, it is straightforward to explain its meaning.
- examples of poor design



redundant information in tuples & update anomalies

- one goal of schema design is to **minimize the storage space**
- example:

A

EMPLOYEE

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

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

B

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
					redundancy	
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

update anomalies

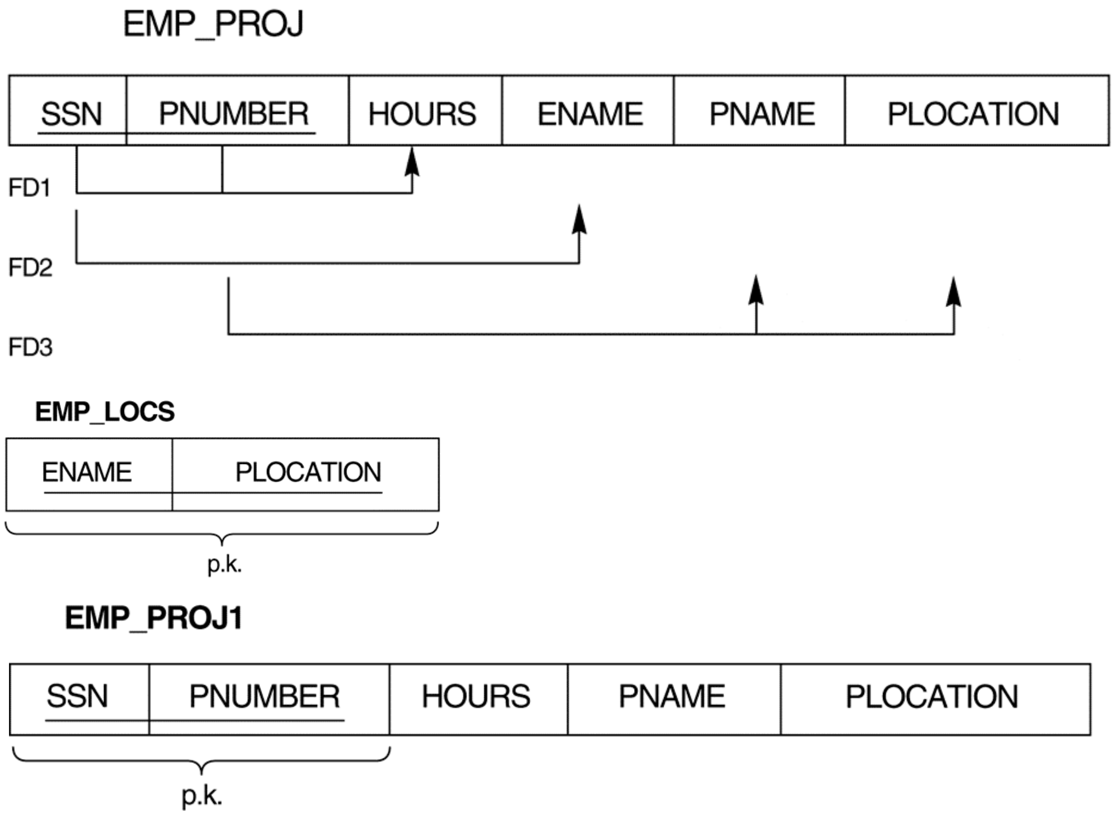
- **insertion** anomalies
  - to insert a new employee tuple into EMP_DEPT, we must include either the **attribute values for the department** that the employee works for, or **nulls**
  - it is difficult to **insert a new department** that has no employees as yet in the EMP_DEPT relation
- **deletion** anomalies
  - if we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, **the information concerning that department is lost**
- **modification** anomalies
  - in EMP_DEPT, if we **change the value of one of the attributes of a particular department**, we must update the tuples of all employees who work in that department
- guideline 2: design the base relation schemas so that **no insertion, deletion, or modification anomalies are present** in the relations

null values in tuples

- grouping many attributes together into a fat relation -> if many of the attributes do not apply to all tuples in the relation, we end up with **many nulls** in those tuples
- example
  - if only 10% of employees have individual offices, there is little justification for including an attribute OFFICE_NUMBER in the EMPLOYEE relation -> A relation EMP_OFFICES(ESSN, OFFICE_NUMBER) can be created
- guideline 3: as far as possible, avoid placing attributes in a base relation whose values may frequently be null

generation of spurious tuples

- example: consider EMP_LOCS and EMP_PROJ1 instead of EMP_PROJ
  - EMP_LOCS: the employee whose name is ENAME works on some project whose location is PLOCATION



generation of spurious tuples (cont.)

- decomposing EMP_PROJ into EMP_LOCS and EMP_PROJ1 is **undesirable** because, when we JOIN them back using NATURAL JOIN, we do not get the correct original information
  - 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

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

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION
123456789	1	32.5	Product X	Bellaire
123456789	2	7.5	Product Y	Sugarland
666884444	3	40.0	Product Z	Houston
453453453	1	20.0	Product X	Bellaire
453453453	2	20.0	Product Y	Sugarland
333445555	2	10.0	Product Y	Sugarland
333445555	3	10.0	Product Z	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston

SSN	PNUMBER	HOURS	PNAME	PLOCATION	
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.

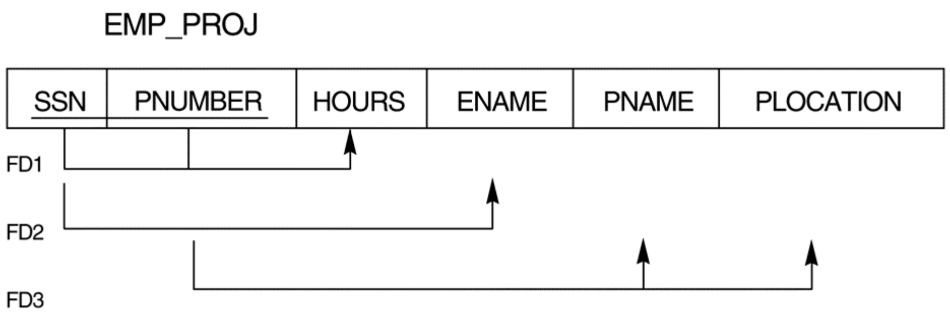
⋮

generation of spurious tuples (cont.)

- guideline 4: design relation schemas so that **they can be joined with equality conditions on attributes that are either primary keys or foreign keys** in a way that guarantees that no spurious tuples are generated

definition

- a **functional dependency** (FD), denoted by *X* -> *Y*, between two sets of attributes *X* and *Y* that are subsets of *R* specifies a **constraint on the possible tuples** that can form a relation state *r* of *R*
    - **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 *X* is a candidate key of *R*, *X* -> *Y* for any subset of attributes *Y* of *R*
- if *X* -> *Y* in *R*, this does not say whether or not *Y* -> *X* in *R*
- example
    - FD1: {SSN, PNUMBER} -> HOURS
    - FD2: SSN -> ENAME
    - FD3: PNUMBER -> {PNAME, PLOCATION}



inference rules for FDs

- F: the set of functional dependencies that are specified on relation schema R
- F^+ (closure of F): the set of all dependencies that include F as well as all dependencies that can be **inferred** from F
- example
  - F = {SSN -> {ENAME, BDATE, ADDRESS, DNUMBER}, DNUMBER -> {DNAME, DMGRSSN}}
  - SSN -> {DNAME, DMGRSSN}
  - SSN -> SSN
  - DNUMBER -> DNAME
- notations
  - $F \models X \rightarrow Y$: X -> Y is inferred from F
  - $\{X,Y\}$ -> Z is abbreviated to XY -> Z

well-known inference rules

- IR1 (reflexive rule)
  - If $X \supseteq Y$, then $X \text{ -> } Y$
- IR2 (augmentation rule)
  - $\{X \text{ -> } Y\} \models XZ \text{ -> } YZ$
- IR3 (transitive rule)
  - $\{X \text{ -> } Y, Y \text{ -> } Z\} \models X \text{ -> } Z$
- IR4 (decomposition rule)
  - $\{ X \text{ -> } YZ\} \models X \text{ -> } Y$
- IR5 (union rule)
  - $\{X \text{ -> } Y, X \text{ -> } Z\} \models X \text{ -> } YZ$
- IR6 (pseudotransitive rule)
  - $\{X \text{ -> } Y, WY \text{ -> } Z\} \models WX \text{ -> } Z$

closure computation

- closure X^+: the set of attributes that are functionally determined by X based on F
- algorithm
  - $X^+ = X$
  - repeat
    - $oldX^+ = X^+$
    - for each FD Y -> Z in F do
      - if $X^+ \supseteq Y$, then $X^+ = X^+ \cup Z$
  - until ($X^+ = oldX^+$)
- example
  - F = {SSN -> ENAME, PNUMBER -> {PNAME, PLOCATION}, {SSN, PNUMBER} -> HOURS}
  - {SSN}$^+$ = {SSN, ENAME}
  - {PNUMBER}$^+$ = {PNUMBER, PNAME, PLOCATION}
  - {SSN, PNUMBER}$^+$ ={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS}

equivalence of sets of FDs

- F: a set of FDs
- F^+: closure of F
  - the set of all FDs logically implied by F
- F is said to **cover** another set of FDs E if every FD in E is also in F^+
- F covers E if
  - for every FD (X -> Y) in E, X^+ (w.r.t. F) $\supseteq Y$
- That is, $X^+ \supseteq Y \Rightarrow X^+ \rightarrow Y \Rightarrow X \rightarrow X^+$; $X^+ \rightarrow Y \Rightarrow X \rightarrow Y$
- two sets of FDs E and F are equivalent if $E^+ = F^+$

minimal sets of FDs

- minimal cover of a set of FDs E: a set of FDs F that satisfies the property that
  - **every** FD in E is in F^+
  - the above property is lost if any FD from F is removed
- formally, F is **minimal** if
  - every FD in F has a single attribute for its **rhs**
  - we cannot replace any FD X -> A in F with a FD Y -> A, where $Y \subset X$, and still have a set of FDs that is equivalent to F
  - we cannot remove any FD from F and still have a set of FDs that is equivalent to F

algorithm for finding a minimal cover F for E

- set $F = E$
- replace each FD X -> $\{A_1, ..., A_n\}$ in F by the n functional dependencies X -> A_1, ..., X -> A_n
- for each FD X -> A in F
  - for each attribute $B \in X$
    - if $\{\{F - \{X \text{ -> } A\}\} \cup \{(X - \{B\}) \text{ -> } A\}\}$ is equivalent to F
    - then replace X -> A with $(X - \{B\})$ -> A in F
- for each remaining FD X -> A in F
  - if $\{F - \{X \text{ -> } A\}\}$ is equivalent to F
  - then remove X -> A from F

normalization of relations

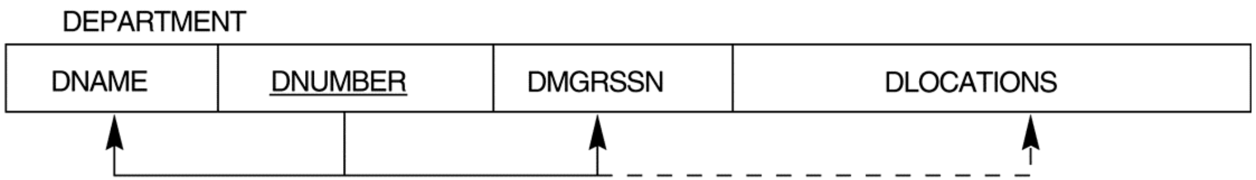
- first proposed by Codd
- takes a relation schema through a series of tests to certify whether it satisfies a certain normal form
- a process of analyzing the given relation schemas based on their **FDs** and **primary keys** to achieve the desirable properties of (1) **minimizing redundancy**, and (2) **minimizing the insertion, deletion, and update anomalies**
- the process of normalization through **decomposition** must confirm the existence of additional properties that the relational schemas should possess: e.g., nonadditive join property, dependency preservation property
- 1NF, 2NF, 3NF, and BCNF: based on the functional dependencies among the attributes of a relation
- 4NF, 5NF: Based on the concepts of **multivalued dependencies** and **join dependencies** respectively

keys and attributes participating in keys

- **superkey** of a relation schema $R = \{A_1, ..., A_n\}$
  - a set of attributes $S \subseteq R$ 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
- 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
- an attribute of relation schema R is called a **prime attribute** of R if it is a **member of some candidate key** of R

first normal form (1NF)

- to disallow **multivalued attributes**, **composite attributes**, and their combinations
- the domain of an attribute must include only **atomic values** and the value of any attribute in a tuple must be a **single value** from the domain of that attribute
- example



DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

3 main techniques to achieve 1NF

- remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key DNUMBER of DEPARTMENT -> generally considered best
- expand the key so that there will be a **separate tuple** in the original DEPARTMENT relation for each location of a DEPARTMENT -> introduces **redundancy**
- if a maximum number of values is known: DLOCATION1, DLOCATION2, ... -> introduces **null values**

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

another example: nested relation

- EMP_PROJ(SSN, ENAME, {PROJS(PNUMBER, HOURS)})
- SSN is the primary key of the EMP_PROJ while PNUMBER is the **partial key** of the nested relation
- for normalization into 1NF, we remove the nested relation attributes into a new relation and propagate the primary key into it

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

EMP_PROJ

SSN	ENAME	PROJS	
		PNUMBER	HOURS

EMP_PROJ1

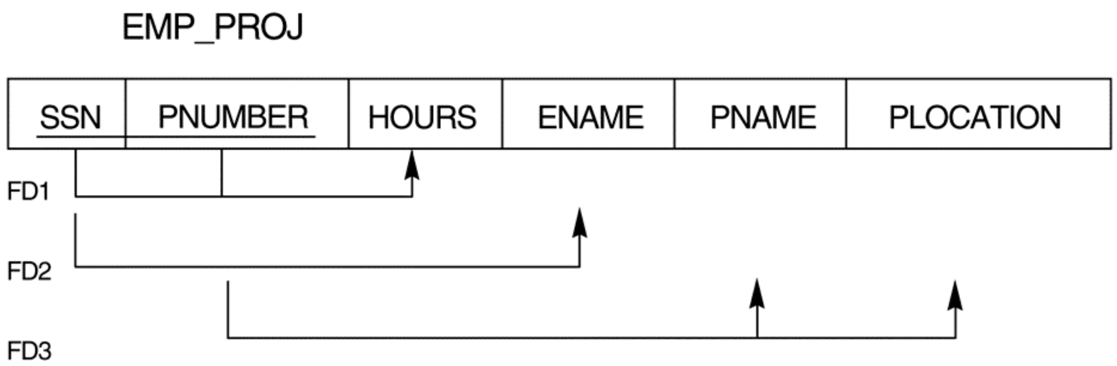
SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

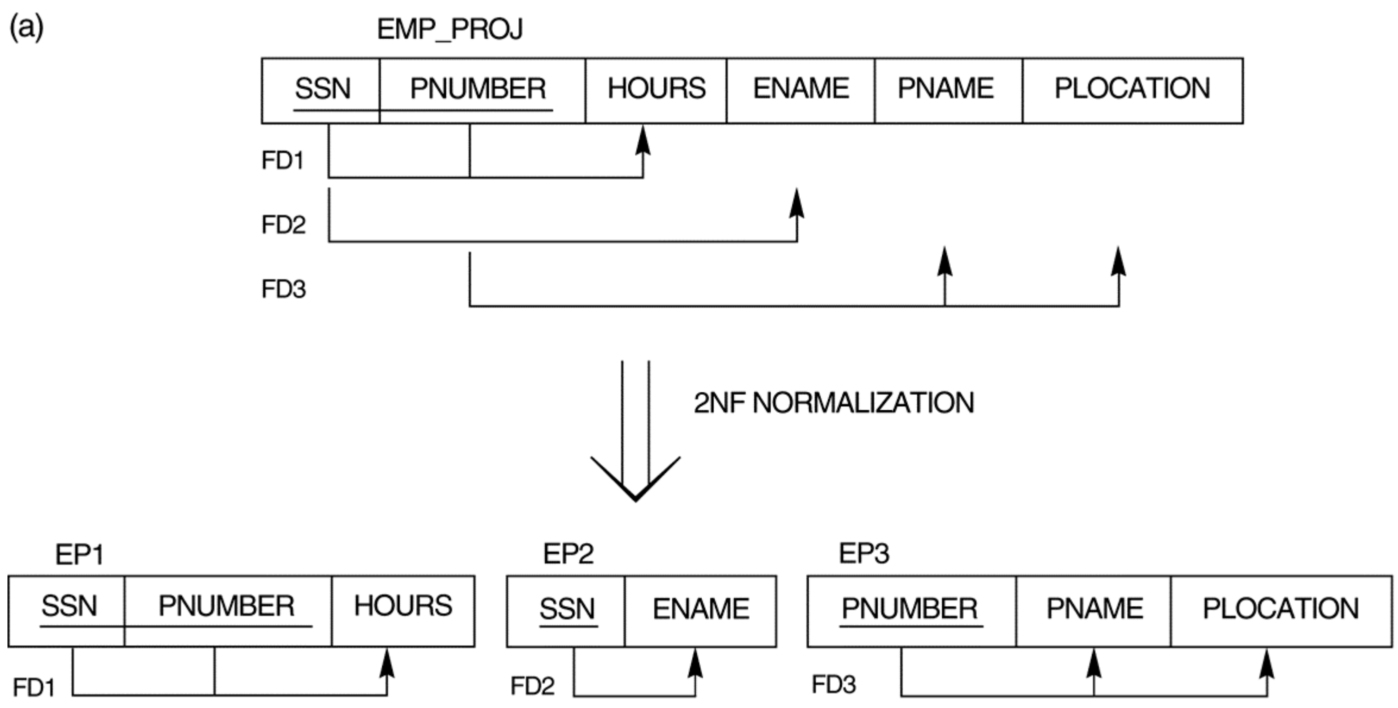
second normal form (2NF)

- an FD *X* -> *Y* is a **full functional dependency (FFD)** if removal of any attribute *A* from *X* means that the dependency does not hold any more
- an FD *X* -> *Y* is a **partial dependency** if some attribute $A \in X$ can be removed from *X* and the dependency still holds
- a relation schema *R* is in **2NF** if **every nonprime attribute *NA*** in *R* is **fully functionally dependent** on the **primary key** of *R*
- example: {SSN, PNUMBER} is a primary key for EMP_PROJ
  - {SSN, PNUMBER} -> ENAME: FFD?
  - {SSN, PNUMBER} -> PNAME: FFD?
  - {SSN, PNUMBER} -> PLOCATION: FFD?



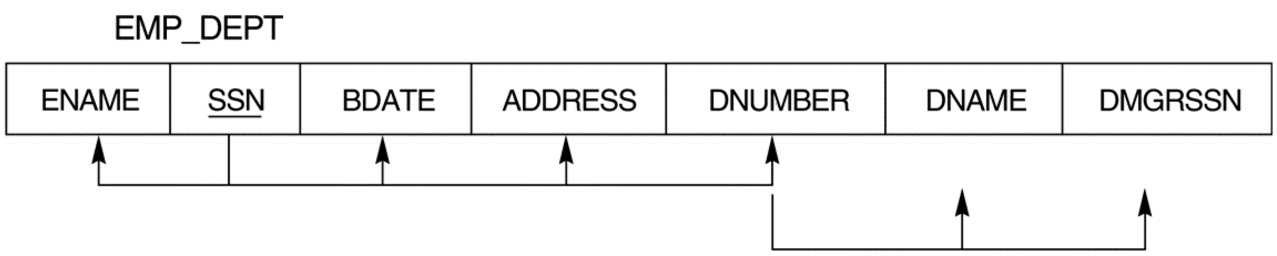
converting into 2NF

- if a relation schema is not in 2NF, it can be 2NF normalized into a number of 2NF relations in which **nonprime attributes are associated only with the part of the primary key** on which they are fully functionally dependent

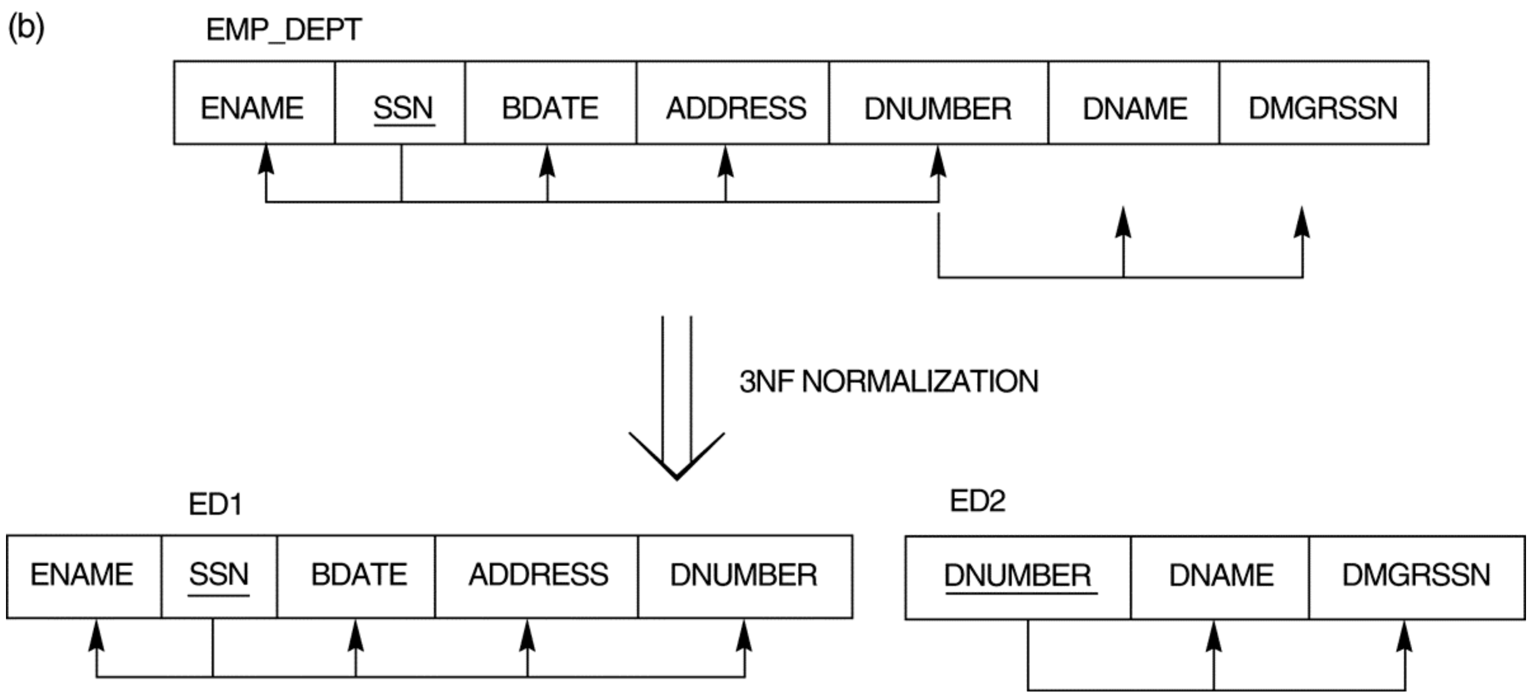


third normal form (3NF)

- an FD *X* -> *Y* in a relation schema *R* is a **transitive dependency** if there is a set of attributes *Z* that is **neither a candidate key nor a subset of any key** of *R*, and both ***X* -> *Z*** and ***Z* -> *Y*** hold
- a relation schema *R* is in **3NF** if it **satisfies 2NF** and **no nonprime attribute** of *R* is **transitively dependent** on the **primary key**
- example
  - SSN -> DMGRSSN is transitively dependent because DNUMBER is a nonprime attribute, SSN -> DNUMBER and DNUMBER -> DMGRSSN hold, and DNUMBER is neither a key nor a subset of the key of EMP_DEPT



example

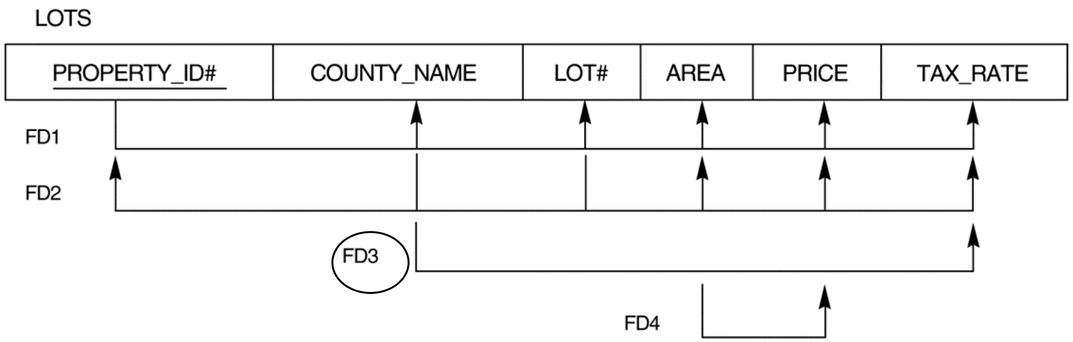

(b)
EMP_DEPT
ENAME
SSN
BDATE
ADDRESS
DNUMBER
DNAME
DMGRSSN
3NF NORMALIZATION
ED1
ENAME
SSN
BDATE
ADDRESS
DNUMBER
ED2
DNUMBER
DNAME
DMGRSSN

general definitions of 2nd and 3rd normal forms

- the previous definition of 3NF disallows partial and transitive dependencies on the primary key to avoid update anomalies
- now the partial and full functional dependencies and transitive dependencies are considered **w.r.t. all candidate keys** of a relation

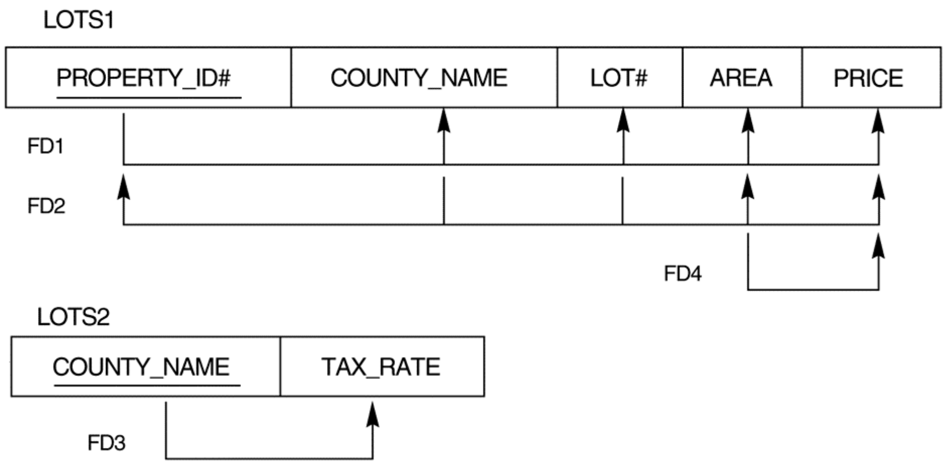
general definition of 2NF

- **prime** attribute: an attribute that is **part** of **some candidate** key
- a relation schema *R* is in 2NF if **every nonprime attribute** *A* in *R* is **not partially dependent** on **<u>any</u>** key of *R*



candidate keys:
PROPERTY_ID#,
{COUNTY_NAME, LOT#}

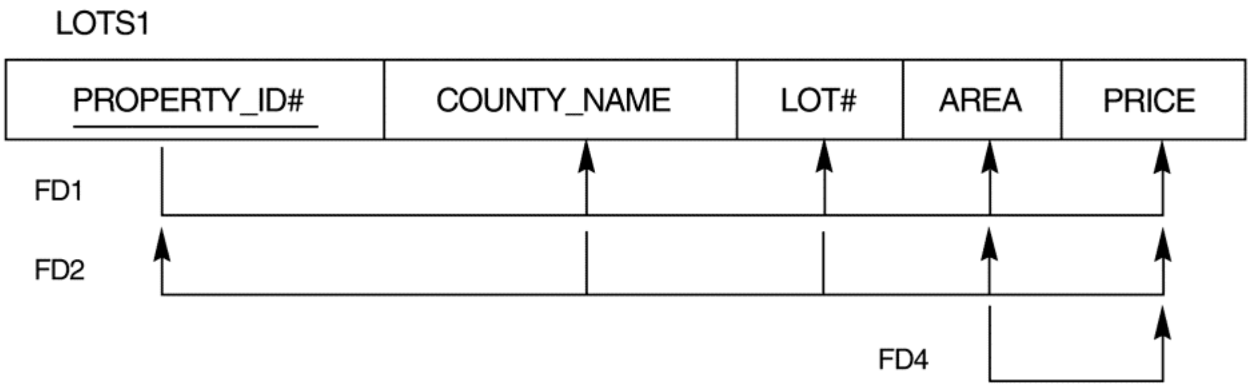
{COUNTY_NAME, LOT#} -> TAX_RATE: FFD?



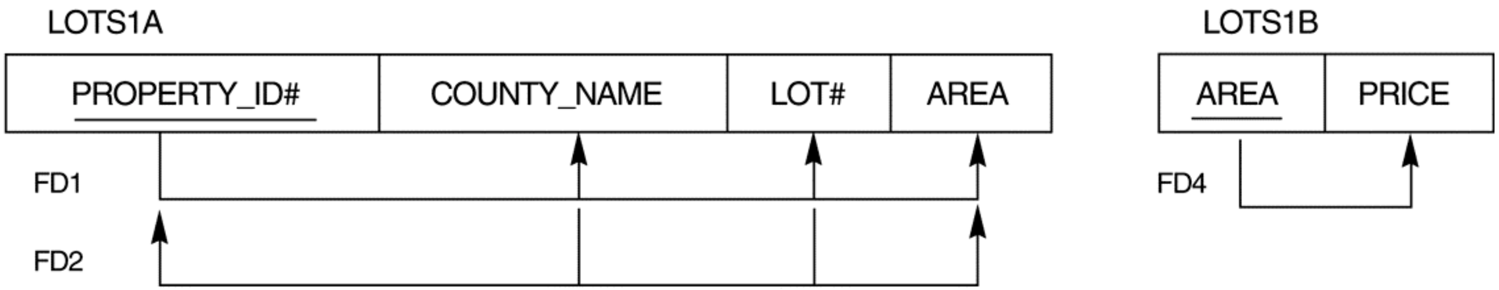
general definition of 3NF

- def) a relation schema *R* is in 3NF satisfies the following property
  - whenever a nontrivial functional dependency $X \rightarrow A$ holds in *R*, either (a) *X* is a **superkey** of *R*, or (b) *A* is a **prime attribute** of *R*
- an FD $X \rightarrow A$
  - violating (b) => *A* is a nonprime attribute $\wedge$
  - violating (a) => *X* is not a superset of any key of *R*
    - => *X* is either **nonprime** or a **proper subset** of a key of *R*
    - *X* is nonprime => transitive dependency (i.e., $\exists$ a key *Y*, s.t. $Y \rightarrow X \rightarrow A$)
    - *X* is a proper subset of a key => partial dependency (i.e., $\exists$ a partial dependency "$Z(\supset X) \rightarrow A$" due to the existence of "$X \rightarrow A$")
- therefore, a relation schema *R* is in 3NF if for **every nonprime** attribute *A* of *R*
  - it is **non-transitively dependent** on <u>**every**</u> key of *R*, and
  - it is **fully functionally dependent** on <u>**every**</u> key of *R*

example



- FD4: AREA -> PRICE
  - AREA is not a superkey and PRICE is not a prime attribute
  - that is, from FD1 and FD2, we know that PRICE is transitively dependent on each of the candidate keys (PROPERTY_ID#, {COUNTY_NAME, LOT#}) via the nonprime attribute AREA



Boyce-Codd normal form (BCNF)

- a relation schema R is in BCNF if whenever a nontrivial functional dependency X -> A holds in R, then X is a **superkey** of R
- stricter than 3NF: every relation in BCNF is also in 3NF, but a relation in 3NF is not necessarily in BCNF
- example
  - FD5
    - {COUNTY_NAME, LOT#} is a candidate key
    - AREA is not a superkey => violates BCNF
    - COUNTY_NAME is a prime attribute => satisfies 3NF