Relational Database Design

Relational Database Design

- The theory of relation database design is an attempt to choose good relation schemas.
  - That is, to measure formally why one set of groupings of attributes into relation schemas is better than another.
- Two level to discuss goodness of relation schemas

1. Logical or Conceptual
   - How users interpret the relation schemas and the meaning of attributes .
   - Having good relation schemas at this level enables users to understand clearly the meaning of the data in the relation and hence, to formulate their queries correctly.
2. Implementation or storage
   - How the tuples in a base relation are stored and udated.

Guidelines for Relational Database Design

- We first discuss informal guidelines for good relational design
  - Semantics of the Relation Attributes
  - Reducing the redundant values in tuples
  - Reducing the null values in tuples
  - Disallowing the possibility of generating spurious tuples.
- 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)

1.1 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 sets and relationship sets into single relation.

- 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

Figure 14.1 Simplified version of the COMPANY relational database schema.

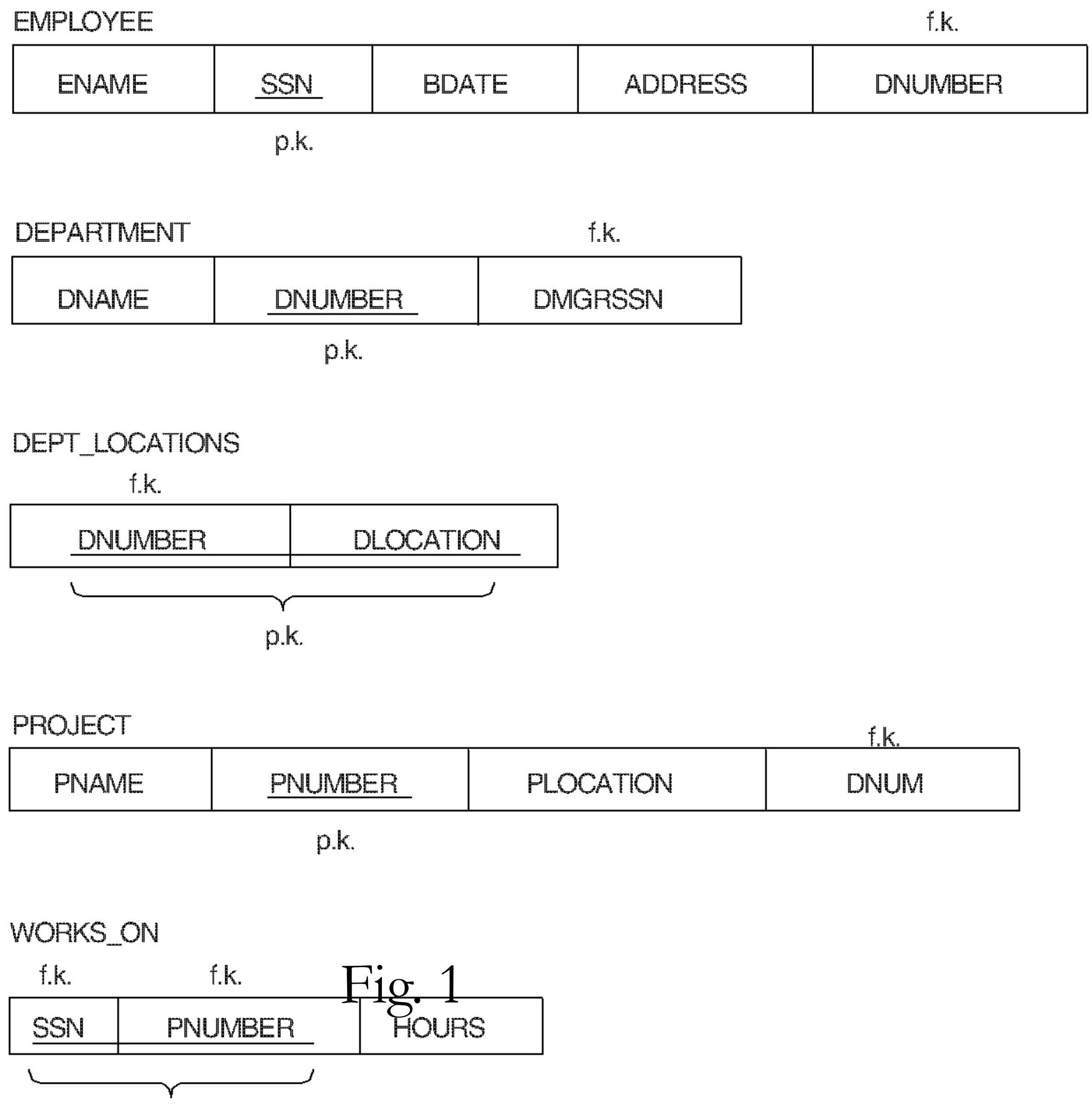
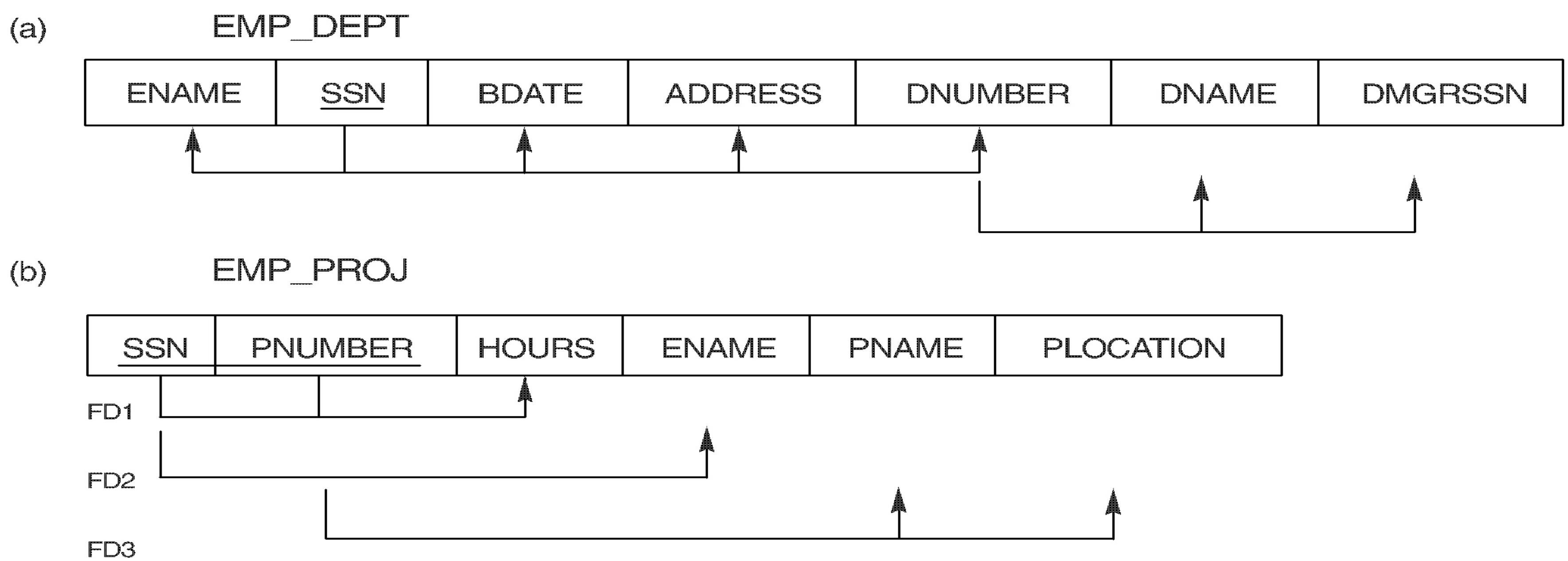


Fig. 1

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

Two relation schemas suffering from update anomalies



Example States for EMP_DEPT and EMP_PROJ

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

EXAMPLE OF AN UPDATE ANOMALY (1)

Insertion Anomalies:

Two different types of anomalies can happen

Consider relation EMP_DEPT table

Case 1. To insert a new employee tuple into EMP_DEPT, you must include either the attributes values for the department that employee works for or null values(if the employee does not work for a department yet. When entering a values for department, the department data must be consistent.

Case 2. It is difficult to insert a new department that has no employees in EMP_DEPT since *SSN* is a primary key and cannot be null and duplicated.

EXAMPLE OF AN UPDATE ANOMALY (2)

- <u>Deletion Anomalies</u>
  - Related to second case that is discussed in insertion anomalies.
  - If we delete from EMP_DEPT an employee tuple that is the last employee working for a particular department. Then information concerning department is also lost.
- <u>Modification Anomalies</u>
  - In EMP_DEPT, if we change the value of one of the attributes of a particular department- say the manage of department number 5, then we must update the tuples of all employees who work in that department. Otherwise, the database will be inconsistent.

Guideline to Redundant Information in Tuples and Update Anomalies

- **GUIDELINE 2:** Design the base relation schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them clearly and make sure that the program that update the database will operate correctly.

1.3 Null Values in Tuples

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

- May end up with many nulls resulting into waste of storage.
- Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - The attribute does not apply to the tuple.
  - The attribute value for the tuple is unknown.
  - The value is known but absent, that is it has not been recorded yet.

  - For example, if only 10 percent of employees have individual offices, creating EMP_OFFICES(SSN, OFFICE_NUMBER) which includes tuples for only the employee with individual office number.

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.

Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys.

Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations.

FIGURE 10.5
Particularly poor design for the EMP_PROJ
(a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 10.4 onto the relations EMP_LOCS and EMP_PROJ1.

(a)

EMP_LOCS

ENAME	PLOCATION

p.k.

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION

p.k.

(b)

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

FIGURE 10.5 (continued)

Particularly poor design for the EMP_PROJ relation of Figure 10.3b. (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 10.4 onto the relations EMP_LOCS and EMP_PROJ1.

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
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

Result of applying NATURAL JOIN to the tuples above the dotted lines in EMP_PROJ1 and EMP_LOCS

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.

⋮

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.

Functional Dependencies

- 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
- A functional dependency is a constraint between two sets of attributes from the database.
- A functional dependency, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subset of relation schema R, then X can uniquely identify the value of Y.

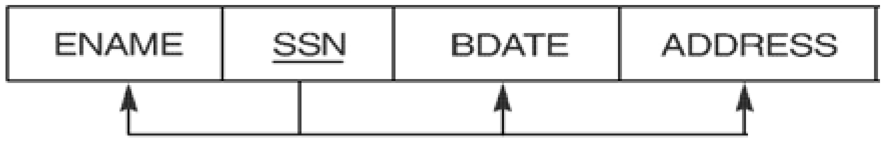
Functional Dependencies (2)

- 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)
- The left side of functional dependency is called determinant.
- X→Y can be pronounced as X determines Y or Y is functionally dependent on X.

Examples of FD constraints (1)

- social security number determines employee name

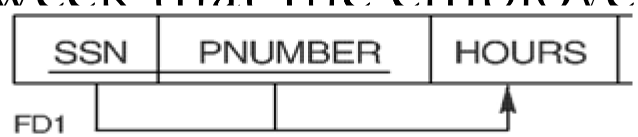
  SSN ->



- projec ne and locatic

  PNUMBER -> {PNAME, PLOCATION}

- employee ssn and project number determines the hours per week that the employee works on the project



  {SSN, PNUMBER} -> HOURS

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])

Inference Rules for 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 $X \supseteq Y$, then $X \rightarrow Y$

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

$\{X \rightarrow Y\} \models \{XZ \rightarrow YZ\}$

(XZ stands for X U Z)

IR3. (**Transitive**) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

$\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

Additional Inference Rules

<u>Some **additional inference rules** that are useful:</u>

(**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

Functional Dependency

- An FD where the right hand side is contained with the left hand is called a **trival FD**.
- Formally, a functional dependency X→Y is trival if X ⊇ Y, otherwise non-trival.

    A -> A always holds (Self-Dependency)

    abc -> a also always hold (Subset-dependency)

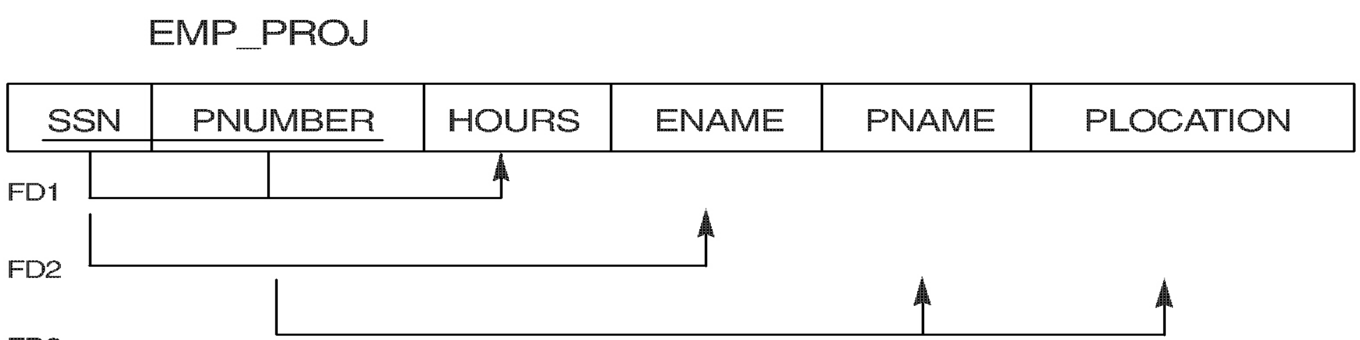
- If there is at least one element on the RHS that is not contained in the LHS, it is called **non-trival**, and if non of the elements of the RHS are contained in LHS, it is complely **non-trival FD**.

Full and partial functional dependency

- A functional dependency is a full functional dependency $X \rightarrow Y$ if removal of any attributes A from X means that the dependency does not hold any more.

  i.e. For any attribute $A \in X$, $(X-\{A\})$ does not functionally determine Y.
- A functional dependency is a partial functional dependency $X \rightarrow Y$ if some attribute A, $A \in X$ can be removed and the dependency still holds.
- {SSN,Pnumber}→Hours (full FD)
- {SSN, Pnumber, Hours, Ename, Pname, Plocation} (partial FD)

(b)



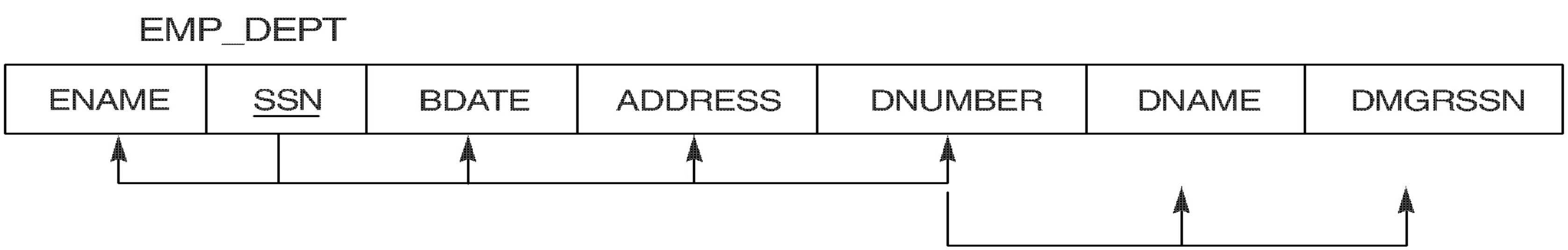
Transitive dependency

- A functional dependency X→Y in a relation schema R is a transitive dependency, if there is a set of attributes Z that is neither candidate key nor a subset of any key of R and both X→Z and Z→Y holds.

- Eg. SSN→DMGSSN can be inferred from following functional dependencies.

  SSN→DNUMBER
  DNUMBER→DMGSSN

(a)



Lossless Decomposition

A decomposition {R1, R2,…, Rn} of a relation R is called a lossless decomposition for R if the natural join of R1, R2,…, Rn produces exactly the relation R.

A decomposition is lossless if we can recover:

$$
\begin{tikzcd}
 & R(A, B, C) \arrow[dl] \arrow[dr] & \\
R1(A, B) \arrow[dr] & & R2(A, C) \arrow[dl] \\
 & R'(A, B, C) &
\end{tikzcd}
$$

Decompose

Recover

Thus, R' = R

Lossless Decomposition

- Let us consider a relation schema R which is divided into two R1 and R2 schemas. The decomposition is lossless if and only if

  $R1 \cap R2 \rightarrow R1$

  or

  $R1 \cap R2 \rightarrow R2$

Example : Problem with Decomposition

R

<u>Model Name</u>	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

R1

<u>Model Name</u>	Category
a11	Canon
s20	Nikon
a70	Canon

R2

Price	Category
100	Canon
200	Nikon
150	Canon

Example : Problem with Decomposition

R1 ⋈ R2

Model Name	Price	Category
a11	100	Canon
a11	150	Canon
s20	200	Nikon
a70	100	Canon
a70	150	Canon

R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

Spurious Tuples Repeat

There are two important properties of decompositions:

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

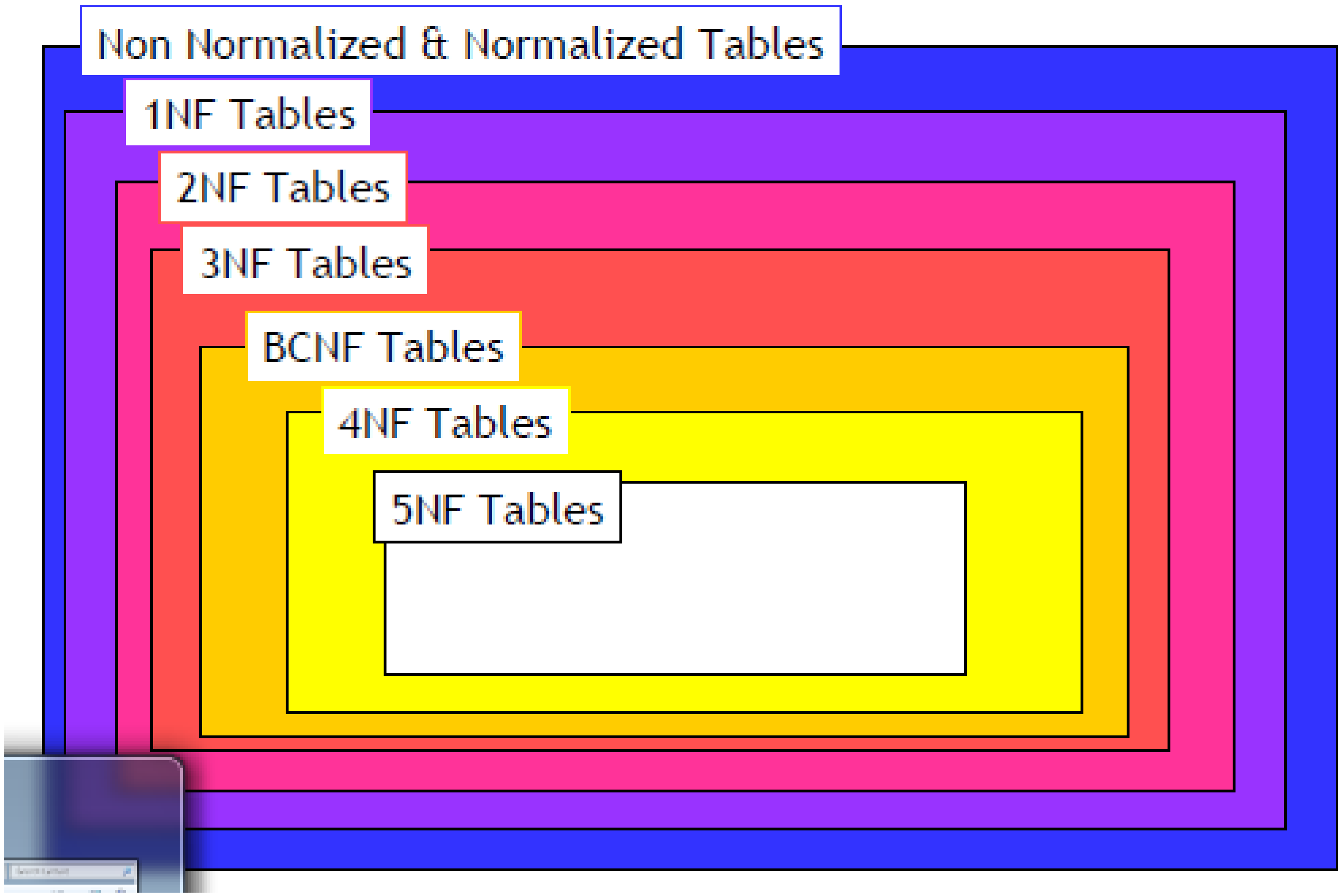
(b) preservation of the functional dependencies.

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

Normalization

- Normalization: Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.
- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.
- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less suffering from update anomalies.

Normalization of Tables



First Normal Form (1NF)

- It states that the domain of an attribute must include only atomic (simple and indivisible) values and value of any attribute in a tuple must be a single value from the domain of that attribute.

- That means, 1NF disallows a set of values, tuples of values or combination of both as an attribute value for a single tuple.

Non-normalized tables

- **Non normalized table: non first normal form**

EXAM (<u>ENO</u>, EXAMINER, SUBJECT, STUDENT (MATNR, NAME, ...))

1	HÄRDER	DBS	1234	May
			5678	Clinton
			9000	Smith
2	EULER	Math	5678	Clinton
			007	Coy

- → contains "attributes" that are tables as well
- Example 2:

<u>Sid</u>	Name	telephone	address
1	Peter	4110001 4342232	KTM
2	Hay	4110111 4110112	KTM
3	Smith	5110113	Pokhara

1NF contd...

- **RULES:**
- For multivalued attributes or/with combinations of composite attributes built up new tables
- Copy down the key

Second Normal Form (2NF)

- Based on concept of full functional dependency
- 1NF causes still a lot of anomalies, because different entity sets can be stored in a table and because of redundancy within a table
- 2NF avoids some of the anomalies by avoiding not fully functionally (partial) dependent attributes

⇒**Separate different entity sets into different tables**

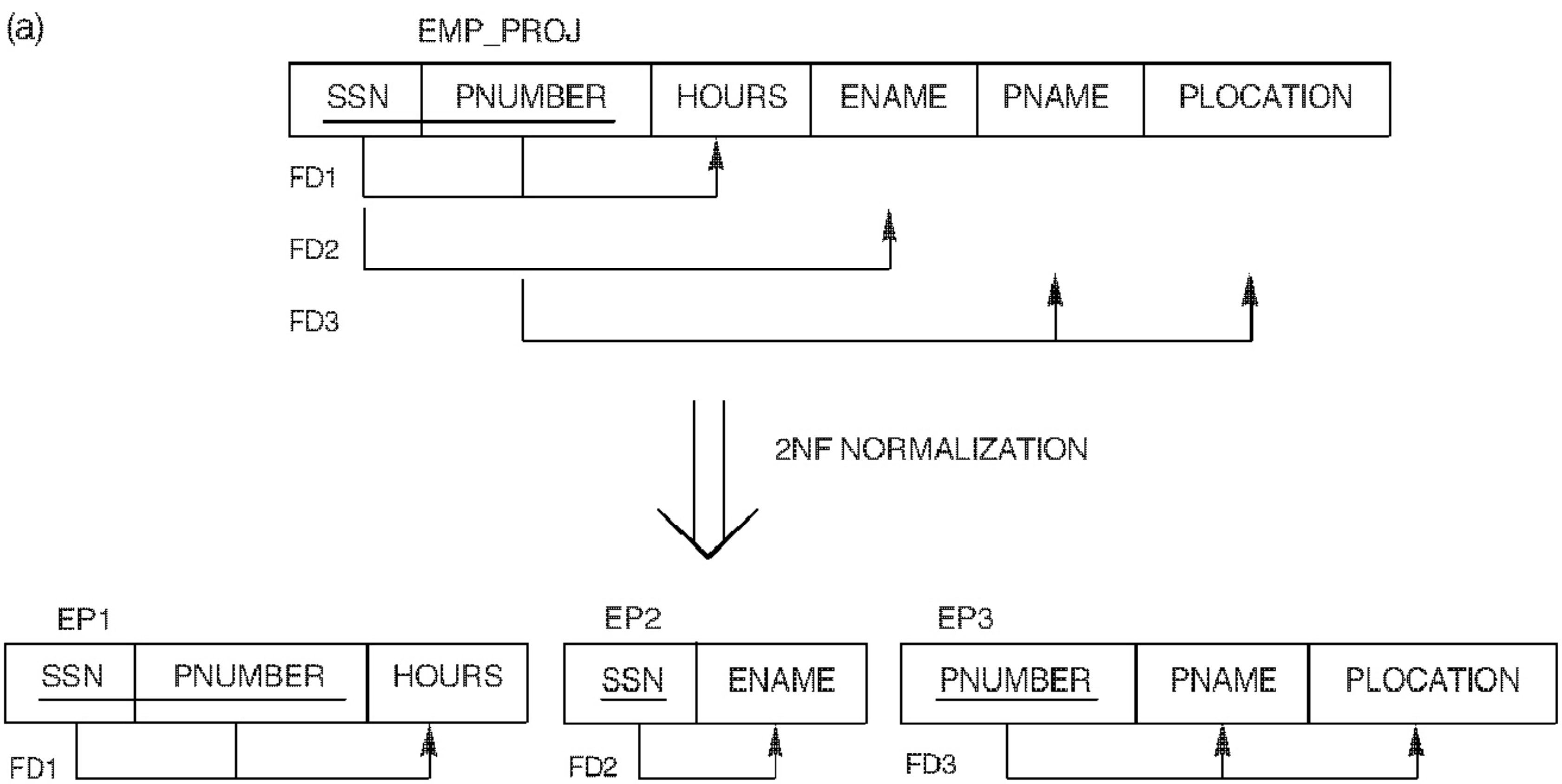
2NF definition

- Second normal form (2NF)
- A relation is in 2NF if and only if
  - It is in 1NF
  - There is no partial dependency on the primary key

    Or in other words

    There is only full functional dependency.

2NF example


(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

Third Normal Form (3NF)

- A relation is in 3NF if and only if
  - it is in 2NF and
  - Non key attributes are non transitively dependent on the primary key.

  OR

  There is no transitive dependency between the primary key and non key attributes.

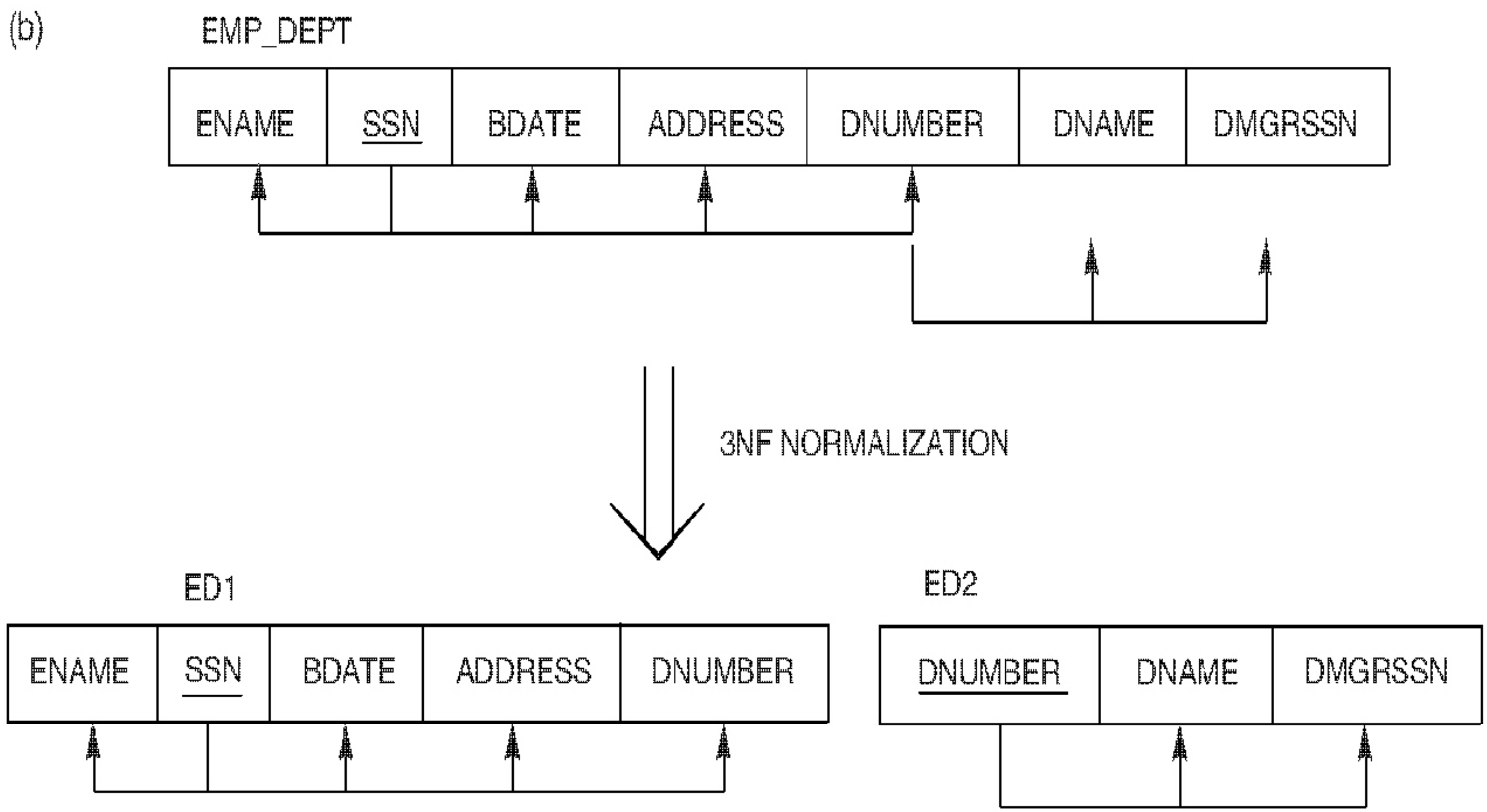
  R(<u>A</u>,B,C) that means A→B and B→C

  <u>Converting into relations in 2NF</u>

  R1(<u>A</u>,B)

  R2(<u>B</u>,C)
- All columns must relate directly to the primary key

3NF example



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

Consider following database schema i.e. in 1NF, Convert to 3NF

R($\underline{A}$, $\underline{B}$, $\underline{C}$, D, E, F, G, H, I, J, K, L, M)

1. Convert the tables into 2NF

2. Convert the tables into 3NF

Consider following database schema i.e. in 1NF, Convert to 3NF

R(A, B, C, D, E, F, G, H, I, J, K, L, M)

1. Convert the tables into 2NF

R1(A, B, C, F, G, H, I, J, K, L) ← R1 in 2NF

√ R2(B, M)

√ R3(C, D, E)

R2 and R3 in 2NF and 3NF

2. Convert the tables into 3NF

√ R11(A, B, C, F, G, I, K, L)

√ R12(G, H)

√ R13(I, J)

Boyce Codd Normal Form (BCNF)

- It states that determinant of a functional dependency is a candidate key not only a primary key.

- The normalization technique is useful for only those relations that have all the following properties.
  1. The relation has two or more candidate keys such that
  2. The candidate keys are composite, and
  3. They overlapped (i.e. has at least one attribute in common.)

BCNF example

The candidate keys are {S#,P#} and {SNAME,P#}

S#	**SNAME**	**P#**	**Qty**

Relation schema
in 3NF
but not in BCNF

S#	**SNAME**	**P#**	**Qty**
S1	Smith	P1	300
S1	Smith	P2	200
S1	Smith	P3	400
S1	Smith	P4	200
…	…	..	..

BCNF example (Solution 1)

Relation schemas in BCNF

S#	SNAME

S#	P#	Qty

S#	SNAME
S1	Smith
…	…

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
…	..	..

BCNF example (Solution 2)

Relation schemas in BCNF

S#	SNAME

SNAME	P#	Qty

S#	SNAME
S1	Smith
...	...

SNAME	P#	Qty
Smith	P1	300
Smith	P2	200
Smith	P3	400
Smith	P4	200
...	..	..

Consider following relation schema i.e. in 1NF, Convert to BCNF

R(A, B, C, D, E, F, G, H, I, J, K, L, M)

1. Convert the tables into 2NF

√ R1(A, B, E) ← In 2NF, 3NF and BCNF

R2(A, B, C, D, F, G, H, I, J, K, L, M) ← In 2NF only

2. Convert tables into 3NF

√ R21(J, K)

√ R22(L, M)

In 3NF and BCNF

R23(A, B, C, D, F, G, H, I, J, , L) ← In 3NF only

3. Convert the tables into BCN

√ R231(B,I)

√ R232(C,G)

√ R233(A, B, C, D, F, H, J, , L)

In BCNF

Remember

- 1NF – ~~atomic~~ value(no group value) for domain of an attribute.
- 2NF – No partial dependency on primary key.

  $R(\underline{A}, \underline{B}, C) \xrightarrow{\text{2NF}} R1(\underline{A}, \underline{B})$ & $R2(\underline{B}, C)$

- 3NF – No Transitive dependency

  $R(\underline{A}, \underline{B}, C, D) \xrightarrow{\text{3NF}} R1(\underline{A}, \underline{B}, C)$ & $R2(\underline{C}, D)$

- BCNF – No overlapping candidate keys

  $R(\underline{A}, \underline{B}, C, D)$, the candidate keys are {A, B} & {A,C}

  $\xrightarrow{\text{BCNF}} R1(\underline{A}, \underline{B}, D)$ & $R2(\underline{B}, C)$