Functional Dependencies and Normalization for Relational Databases

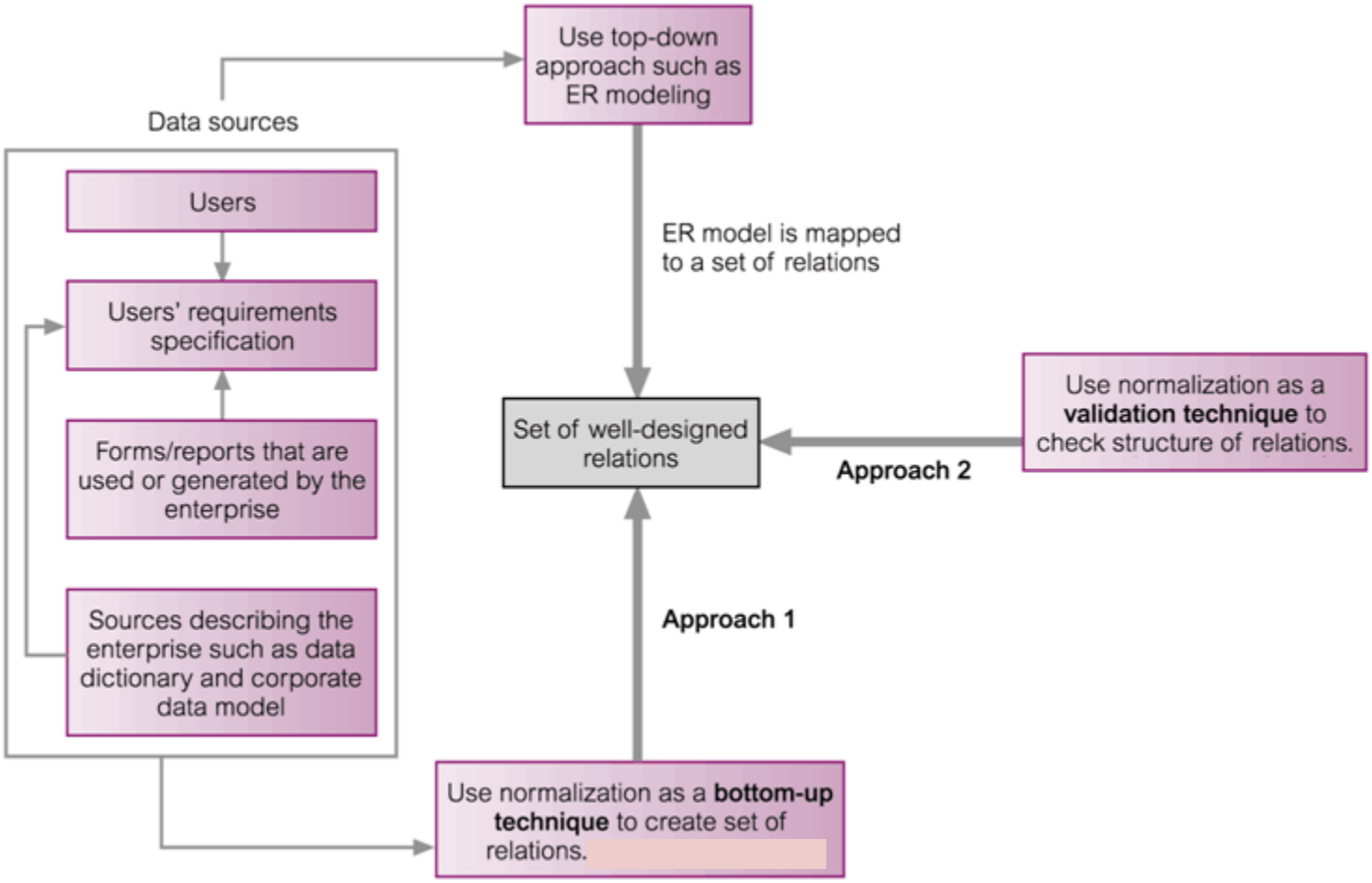
Purpose of Normalization

- Normalization is a technique for producing a set of suitable relations that support the data requirements of an enterprise.

Purpose of Normalization

- The benefits of using a database that has a suitable set of relations is that the database will be:
  - easier for the user to access and maintain the data;
  - take up minimal storage space on the computer.

How Normalization Supports Database Design



Data Redundancy and Update Anomalies

- Major aim of relational database design is to group attributes into relations to minimize data redundancy.

Normalization

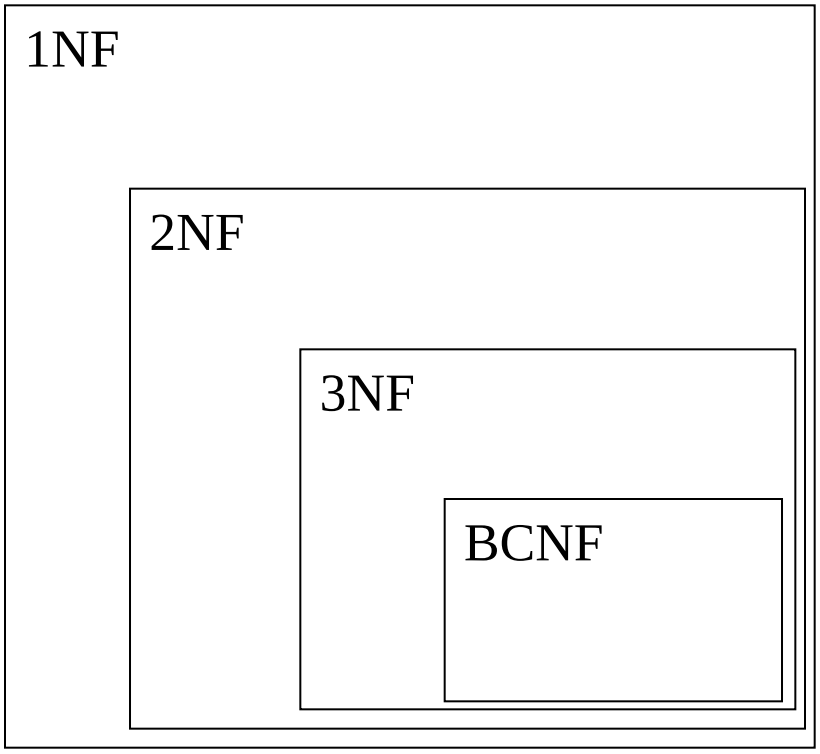
Normalization

Normal forms: first, second, third normal forms, Boyce-Codd Normal Form, fourth and fifth normal forms
1NF, 2NF, 3NF, BCNF, …

Normalization is a process that "improves" a database design by generating relations that are of higher normal forms.

The *objective* of normalization:
"*to create relations where every dependency is on the key, the whole key, and nothing but the key*".

Normalization



a relation in BCNF, is also in 3NF

a relation in 3NF is also in 2NF

a relation in 2NF is also in 1NF

Normalization - Example

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

EMP_DEPT

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

EMP_PROJ

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

Redundancy causes trouble

SSN	Name	Phone Number
123-32-1099	Fred	(201) 555-1234
123-32-1099	Fred	(206) 572-4312
909-43-4444	Joe	(908) 464-0028
909-43-4444	Joe	(212) 555-4000
234-56-7890	Jocelyn	(212) 555-4000

update anomaly = update one copy of Fred's SSN but not the other

deletion anomaly = delete all Fred's phones, lose his SSN as a side effect

insertion anomaly = try to insert a user without any phone number yet

Common sense will tell you how to fix this schema

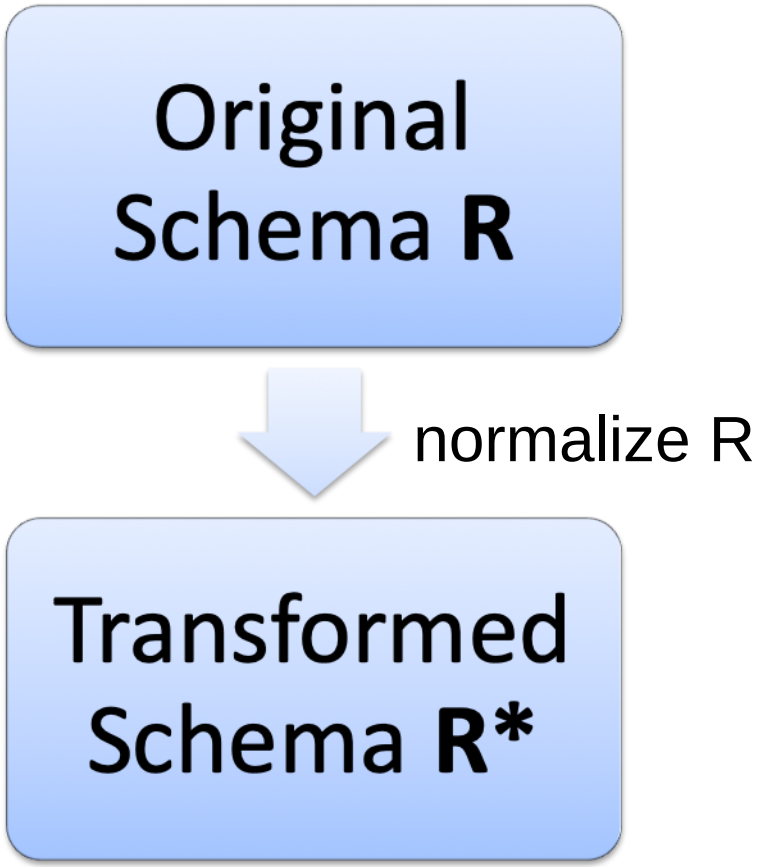
SSN	Name
123-32-1099	Fred
909-43-4444	Joe

SSN	Phone Number
123-32-1099	(201) 555-1234
123-32-1099	(206) 572-4312
909-43-4444	(908) 464-0028
909-43-4444	(212) 555-4000

No more update, delete or insert anomalies.

What if common sense is not enough?

Normalization theory tells you what to do.



R* must:

- Preserve the information of R
- Have minimal redundancy

This theory formalizes the concept of redundancy

Functional Dependencies

Functional Dependencies

We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same. We illustrate this as:

A → B

Example: Suppose we keep track of employee email addresses, and we only track one email address for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number:

employee number → email address

Functional Dependencies

EmpNum	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

If EmpNum is the PK then the FDs:

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

must exist.

But, actually, PK's are inferred from FDs!

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 Y *subset-of* X, then X → Y

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

(Notation: XZ stands for X U Z)

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

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

Inference Rules for 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

Example:

FD: $a \rightarrow b$; $c \rightarrow \{d,e\}$; $\{a,c\} \rightarrow \{f\}$

$\{a\}^+ = \{a,b\}$

$\{c\}^+ = \{c,d,e\}$

$\{a,c\}^+ = \{a,c,f,b,d,e\}$

Definitions

- 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.

Algorithm for Finding the Key

Finding a Key *K* for *R* Given a set *F* of Functional Dependencies

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

1. Set $K := R$.
2. For each attribute *A* in *K*

{ compute $(K - A)^+$ with respect to *F*;

If $(K - A)^+$ contains all the attributes in *R*,

then set $K := K - \{A\}$; }

Transitive dependency

Transitive dependency

Consider attributes A, B, and C, and where

A → B and B → C.

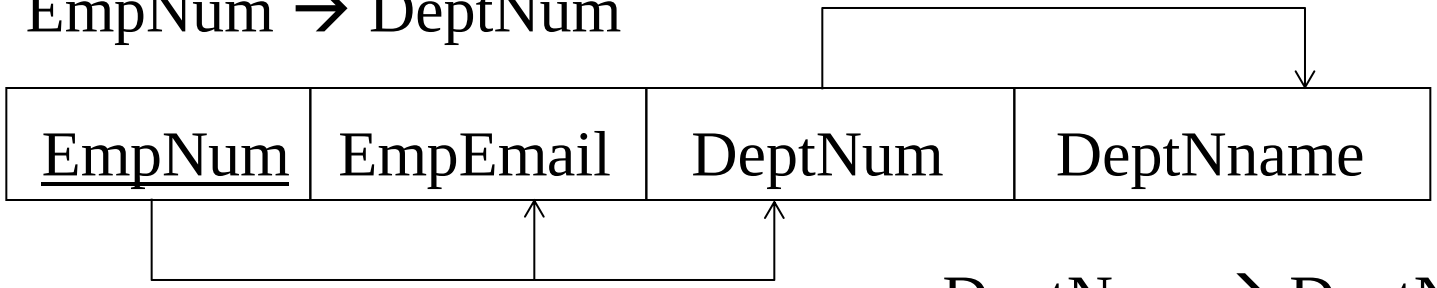
Functional dependencies are transitive, which means that we also have the functional dependency

A → C

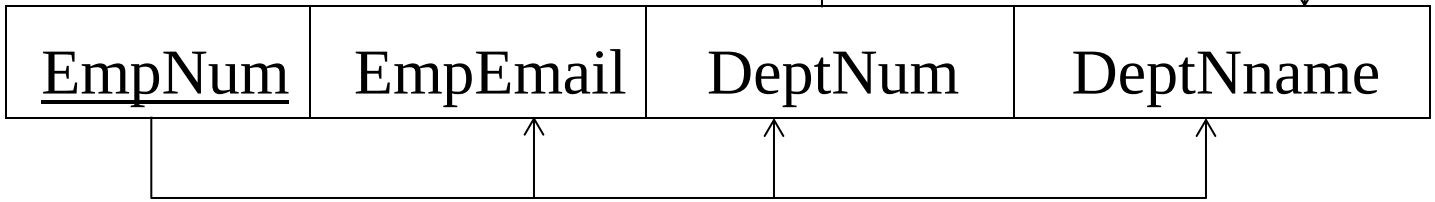
We say that C is transitively dependent on A through B.

Transitive dependency

EmpNum → DeptNum



DeptNum → DeptName

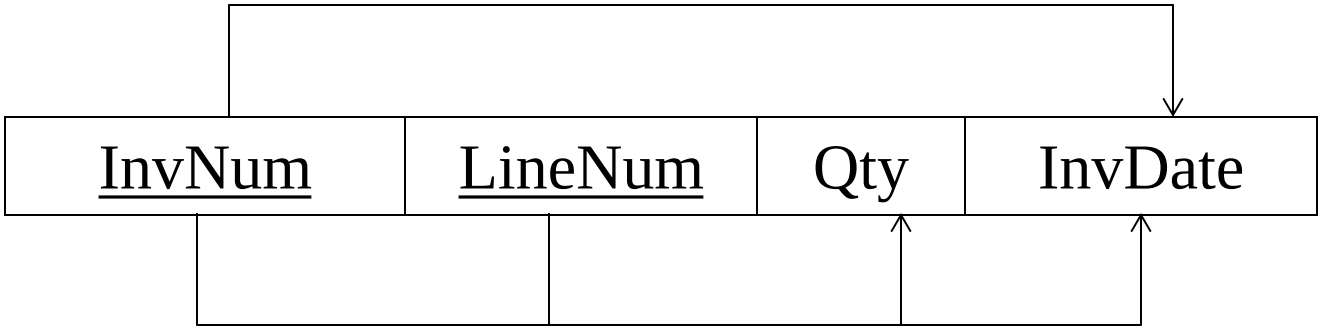


DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum → DeptName

Partial dependency

A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

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

Spurious Tuples (2)

There are two important properties of decompositions:

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

(b) preservation of the functional dependencies.

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

Spurious Tuples (example)

EName	Ploc
a	Ank
a	Ist
b	Ank
b	Esk

Ssn	Pno	Hours	Pname	Ploc
11	1	3	x	Ank
11	2	4	y	Ist
12	1	1	x	Ank
12	3	10	z	Esk

Ssn	Pno	Hours	Pname	Ploc	Ename
11	1	3	x	Ank	a
11	1	3	x	Ank	b
11	2	4	y	Ist	a
12	1	1	x	Ank	a
12	1	1	x	Ank	b
12	3	10	z	Esk	b

First Normal Form

First Normal Form

We say a relation is in **1NF** if all values stored in the relation are single-valued and atomic.

1NF places restrictions on the structure of relations. Values must be simple.

First Normal Form

The following in **not** in 1NF

EmpNum	**EmpPhone**	**EmpDegrees**
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

EmpDegrees is a multi-valued field:

employee 679 has two degrees: *BSc* and *MSc*

employee 333 has three degrees: *BA*, *BSc*, *PhD*

First Normal Form

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

To obtain 1NF relations we must, without loss of information, replace the above with two relations - see next slide

First Normal Form

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

Second Normal Form

Second Normal Form

A relation is in **2NF** if it is in 1NF, and every non-key attribute is fully dependent on each candidate key. (That is, we don't have any partial functional dependency.)

- 2NF (and 3NF) both involve the concepts of key and non-key attributes.
- A *key attribute* is any attribute that is part of a key; any attribute that is not a key attribute, is a *non-key attribute*.
- A relation in 2NF will not have any partial dependencies

Second Normal Form

Consider this **InvLine** table (in 1NF)**:**

InvNum	LineNum	ProdNum	Qty	InvDate

InvNum, LineNum ⟶ ProdNum, Qty

InvNum ⟶ InvDate

InvLine is **not 2NF** since there is a partial dependency of InvDate on InvNum

InvLine is only in **1NF**

Second Normal Form

InvLine

InvNum	LineNum	ProdNum	Qty	InvDate

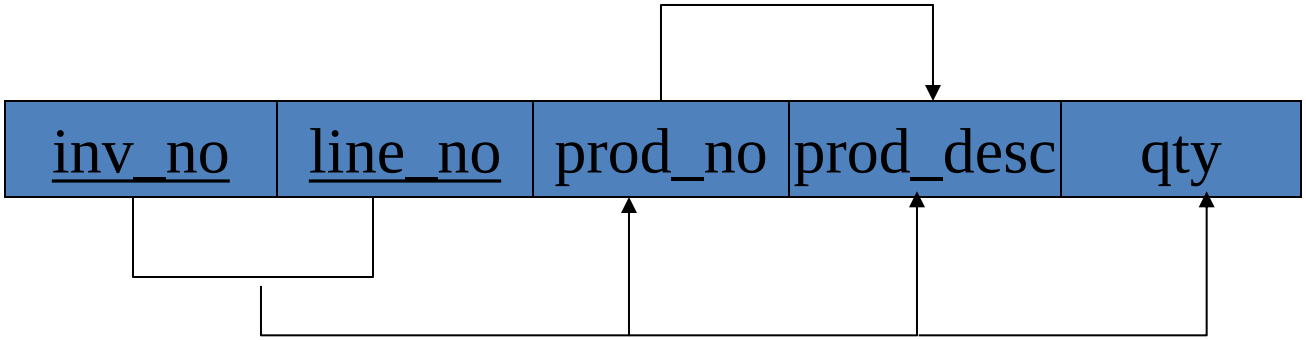
The above relation has redundancies: the invoice date is repeated on each invoice line.

We can *improve* the database by decomposing the relation into two relations:

InvNum	LineNum	ProdNum	Qty

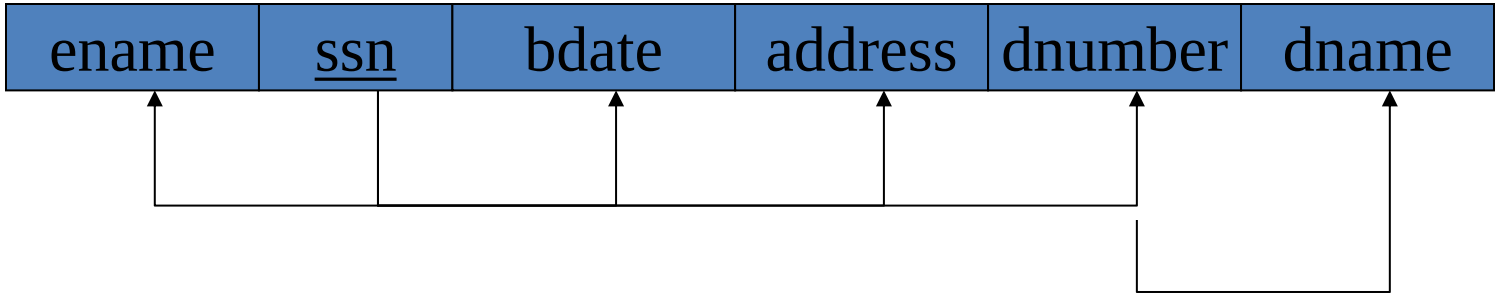
InvNum	InvDate

Is the following relation in 2NF?



2NF, but not in 3NF

EmployeeDept



since dnumber is not a candidate key and we have:

dnumber $\rightarrow$ dname.

Third Normal Form

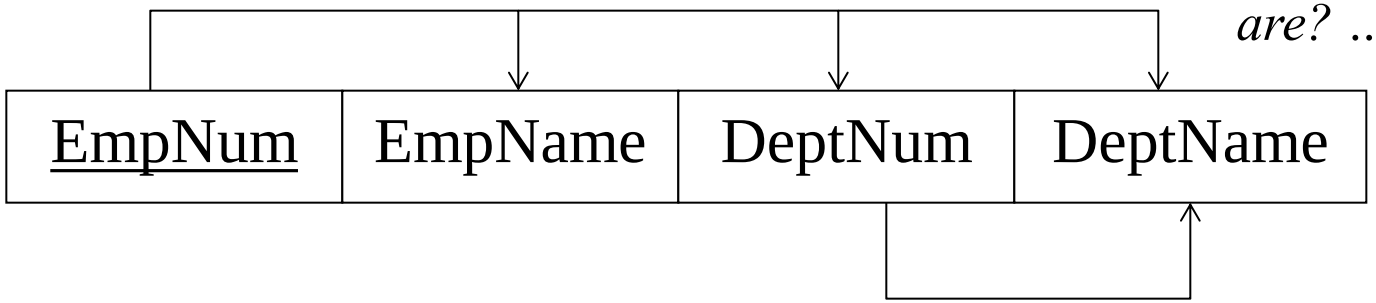
Third Normal Form

- A relation is in **3NF** if the relation is in 2NF and all determinants of *non-key* attributes are candidate keys
  That is, for any functional dependency: X $\rightarrow$ Y, where Y is a non-key attribute (or a set of non-key attributes), X is a candidate key.
- A relation in 3NF will not have any transitive dependencies of non-key attribute on a candidate key through another non-key attribute.

Third Normal Form

Consider this **Employee** relation

Candidate keys are? …



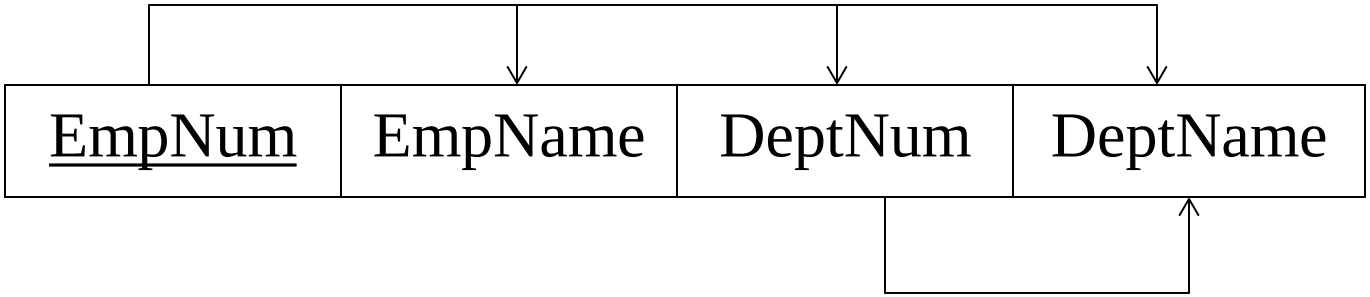
EmpName, DeptNum, and DeptName are non-key attributes.

DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

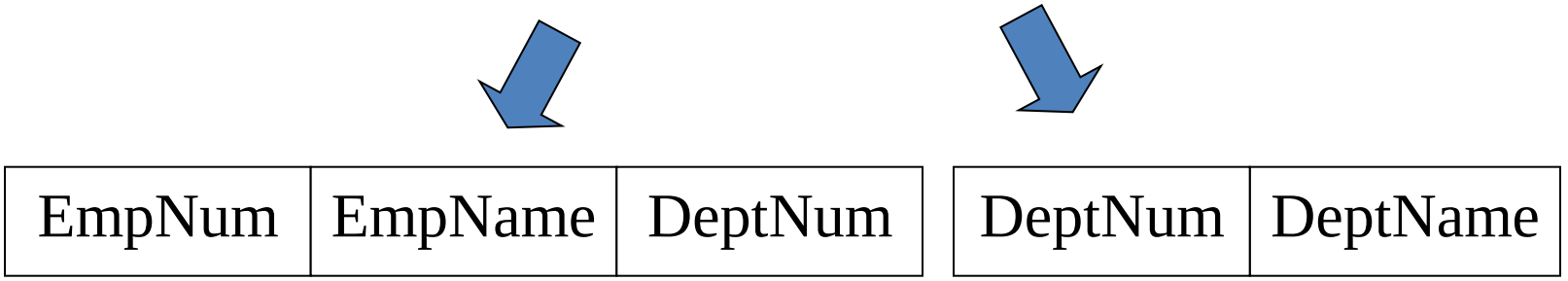
Is the relation in 3NF? … no

Is the relation in 2NF? … yes

Third Normal Form



We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.



Verify these two relations are in 3NF.

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

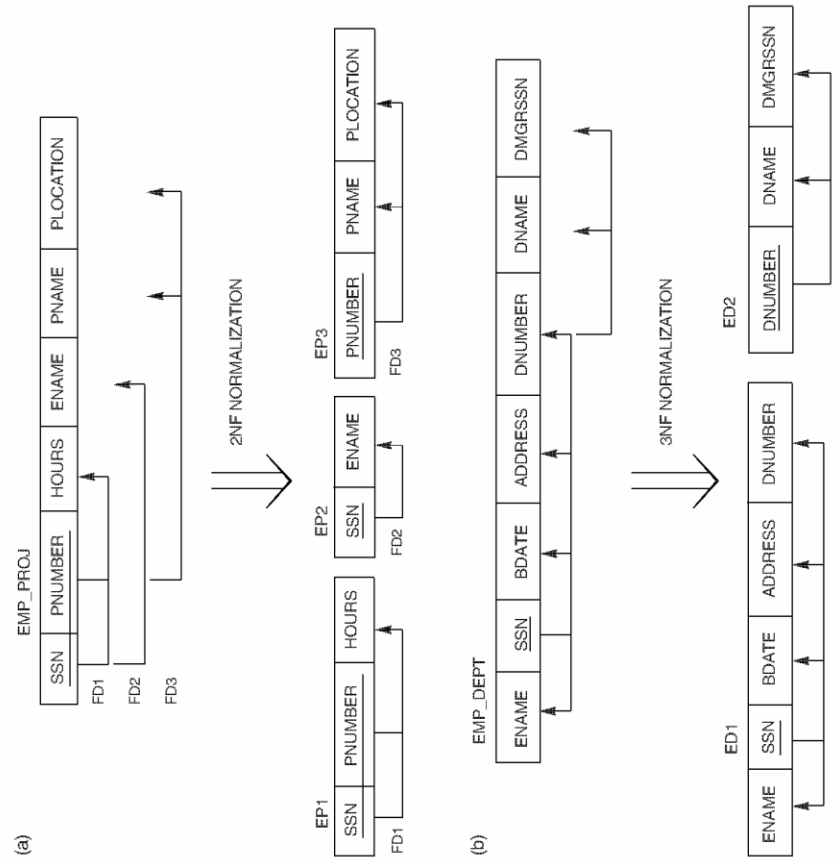


Figure 14.11 Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies fd1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.