Module 5

Relational–Database Design

Contents

- Pitfalls in Relational-Database designs
- Concept of normalization
- Function Dependencies
- First Normal Form
- 2NF
- 3NF
- BCNF

Module 5

Relational–Database Design

Lecture 1

- Pitfalls in Relational-Database designs
- Concept of normalization

Design Guidelines for Relational Algebra

- Making sure that the semantics of the attributes is clear in the schema.
- Reducing the redundant information in tuples.
- Reducing the null values in tuples.
- Facilitate the checking of updates for violation of database integrity constraints.
- Disallowing the possibilities of generating spurious tuples.

Goal of Good Relational Schema Design

- To avoid anomalies (inconsistencies) and redundancy
- Eliminate data dependency within the relation

Problems with Bad Design

- Early computers were slow and had limited storage capacity
- Redundant or repeating data slowed operations and took up too much precious storage space
- Poor design increased chance of data errors, lost information
- Inability to represent certain information.

- Example
- Consider the relation schema:

Lending-schema = (branch-name, branch-city, assets, customer-name, loan-number, amount)

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Problems:

Redundancy: Data for branch-name, branch-city, assets are repeated for each loan that a branch makes

. Wastes space

. Complicates updating

Null values .

- Cannot store information about a branch if no loans exist .
- Can use null values, but they are difficult to handle.

Combine Schemas?

Suppose we combine *instructor* and *department* into *inst_dept*

- *Department(dept_name, building, budget)*
- *Instructor(ID, name, dept_name, salary)*
- Result is possible repetition of information

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

What About Smaller Schemas?

- Suppose we had started with *inst_dept*. How would we know to split up (**decompose**) it into *instructor* and *department*?
- Write a rule "if there were a schema (*dept_name*, *building*, *budget*), then *dept_name* would be a candidate key"
- Denote as a **functional dependency**:

  $$dept\_name \rightarrow building, budget$$

- In *inst_dept*, because *dept_name* is not a candidate key, the building and budget of a department may have to be repeated.
  - This indicates the need to decompose *inst_dept*

Contnd..

- Not all decompositions are good. Suppose we decompose

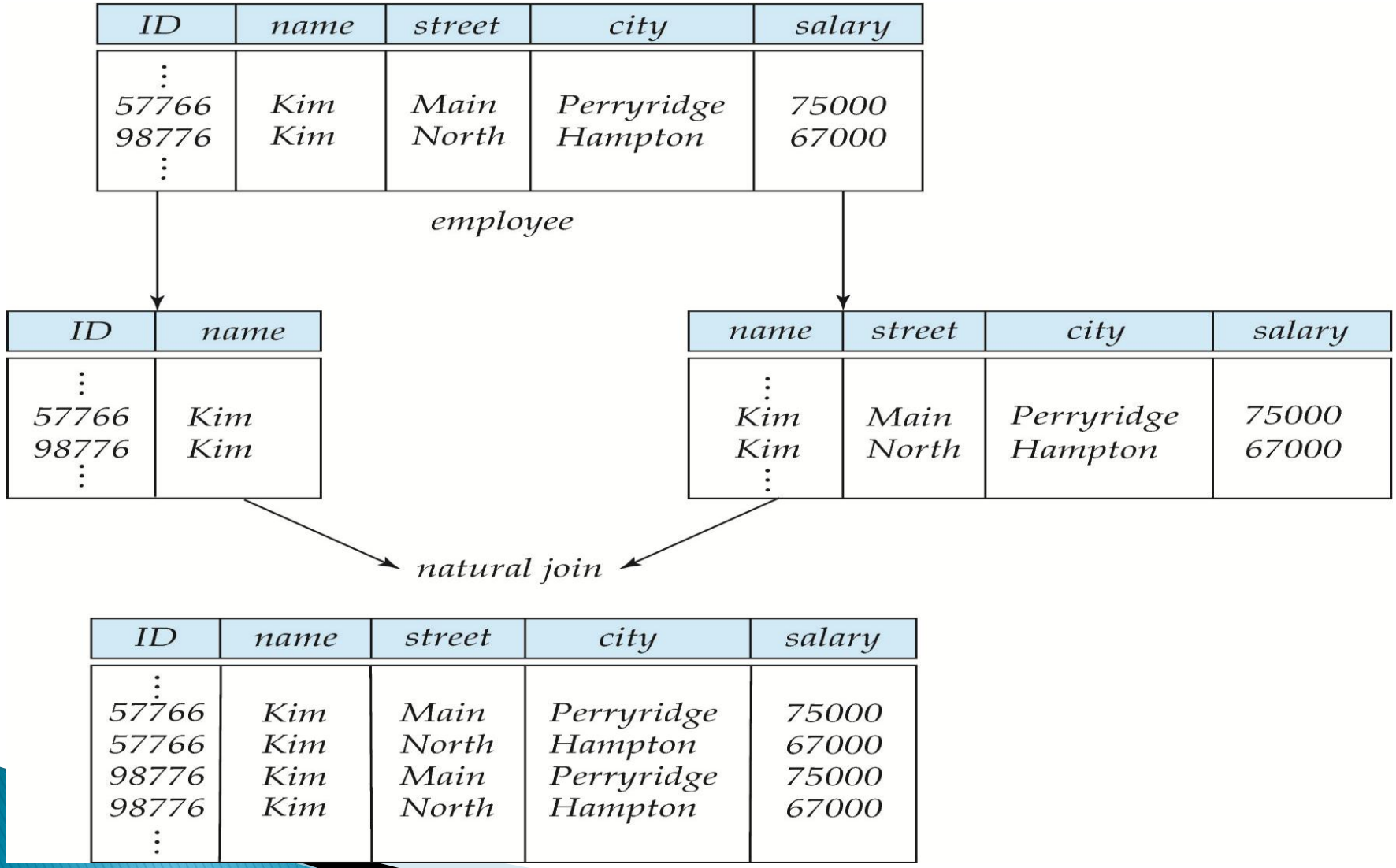
  employee(ID, name, street, city, salary) into

  employee1 (*ID, name*)

  employee2 (*name, street, city, salary*)

- The next slide shows how we lose information -- we cannot reconstruct the original *employee* relation -- and so, this is a **lossy decomposition**.

A Lossy Decomposition



Problems Without Normalization

- If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anomalies are very frequent if database is not normalized.

- **Student**

rollno	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

In the table above, we have data of 4 **Computer Sci**. students. As we can see, data for the fields branch, **hod(Head of Department)** and **office_tel** is repeated for the students who are in the same branch in the college, this is **Data Redundancy**.

- **Insertion Anomaly**

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.

- Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.
- These scenarios are nothing but **Insertion anomalies**.

Updation Anomaly

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

- **Deletion Anomaly**

In our **Student table,** two different information's are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

Modification Anomalies

Customers_Orders_Inventory

Customer	OrderNum	ItemNum	Item
General Tool	07456	2246	Pentium Computer
General Toll	08622	3145	HP Printer
General Tool Co.	08622	3967	17” monitor
Totally Toys	06755	2246	Pentium computer
TOTALLY TOYS	08134	3145	Hewlett-Packard Printer
XYZ Inc.	09010	0446	Dot Matrix Printer

A search for “General Tool Co.” would miss “General Tool” and “General Toll”. A case-sensitive search for “Totally Toys” would miss “TOTALLY TOYS”

Insertion Anomalies

Customers_Orders_Inventory

Customer	OrderNum	ItemNum	Item
General Tool	07456	2246	Pentium Computer
General Toll	08622	3145	HP Printer
General Tool Co.	08622	3967	17” monitor
Totally Toys	06755	2246	Pentium computer
TOTALLY TOYS	08134	3145	Hewlett-Packard Printer
XYZ Inc.	09010	0446	Dot Matrix Printer

How would you enter a new item into your inventory if no one had ordered it yet?

Deletion Anomalies

Customers_Orders_Inventory

Customer	OrderNum	ItemNum	Item
General Tool	07456	2246	Pentium Computer
General Toll	08622	3145	HP Printer
General Tool Co.	08622	3967	17” monitor
Totally Toys	06755	2246	Pentium computer
TOTALLY TOYS	08134	3145	Hewlett-Packard Printer
XYZ Inc.	09010	0446	Dot Matrix Printer

If you wanted to stop selling “dot matrix printer” and remove it from your inventory, you would have to delete the order and customer info for “XYZ Inc.”

The Fix

Order_Items

OrderNum	ItemNum
06755	2246
07456	2246
08134	3145
08622	3145
08622	3967
09010	0446

Orders

CustomerNum	OrderNum
7822	09010
8755	06755
8755	08134
9123	07456
9123	08622

Customers

CustomerNum	Customer
7822	XYZ Inc.
8755	Totally Toys
9123	General Tool Co.

Products

ItemNum	Item
0446	Dot Matrix Printer
2246	Pentium Computer
3145	Hewlett-Packard printer
3967	17” monitor

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

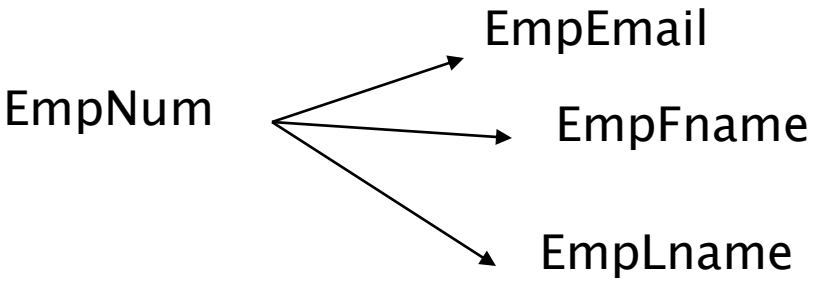
EmpNum → EmpLname

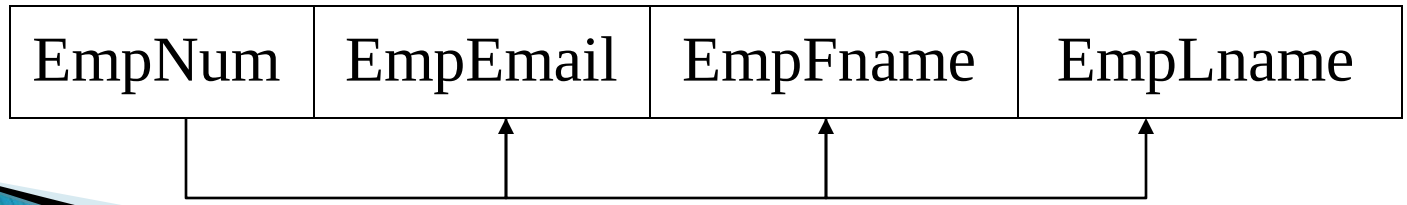
must exist.

Functional Dependencies

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

3 different ways you might see FDs depicted





Determinant

Functional Dependency

EmpNum → EmpEmail

Attribute on the LHS is known as the ***determinant***

- EmpNum is a determinant of EmpEmail

Transitive dependency

Transitive dependency

Consider attributes A, B, and C, and where

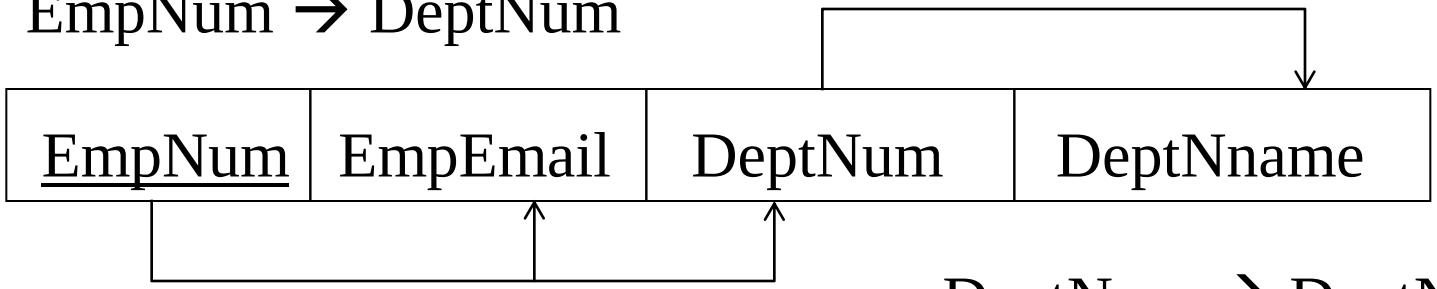
A → B and B → C.

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

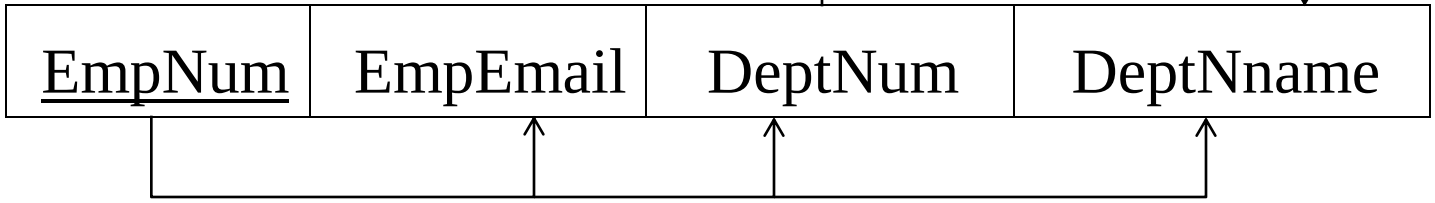
We say that C is transitively dependent on A through B.

Transitive dependency

EmpNum → DeptNum



DeptNum → DeptName

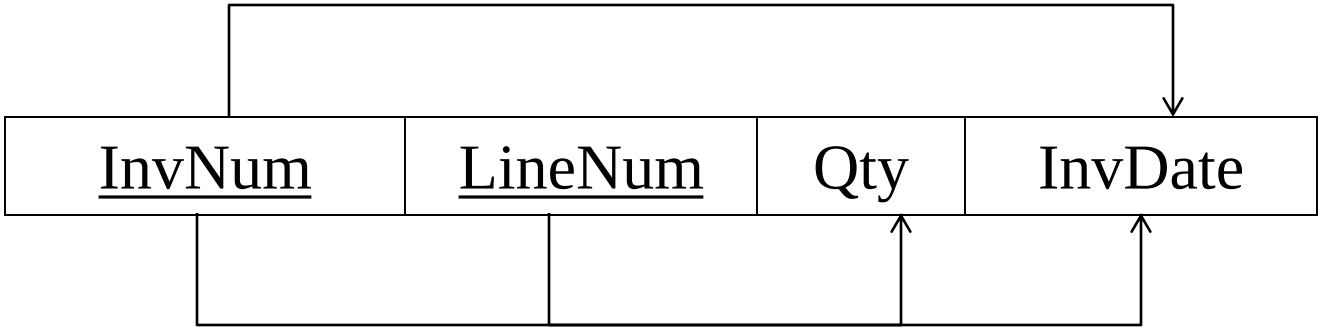


DeptName is *transitively dependent* on EmpNum via DeptNum

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

e.g. (Stud-id, Co_id) ->Grade; FFD

(stud_id)-> Stu_ name, Stu_contact PFD

Module 5

Relational–Database Design

Lecture 2

- Concept of normalization
- Function Dependencies
- First Normal Form
- Second Normal Form

Normalization Why?

- All relations are not equal
- Tables not normalized experience issues known as modification problems
  - Insertion problems
    - Difficulties inserting data into a relation
  - Modification problems
    - Difficulties modifying data into a relation
  - Deletion problems
    - Difficulties deleting data from a relation

Normalization

- *Normalization* is a process of analyzing a relation to ensure that it is *well formed*
- More specifically, if a relation is normalized (well formed), rows can be inserted, deleted, or modified without creating update anomalies.

Normalization

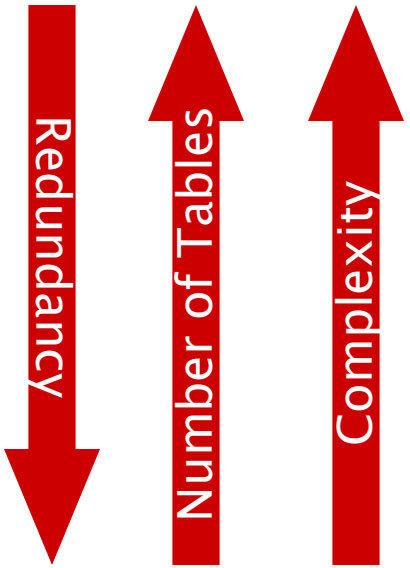
- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Goals of Normalization

- Eliminate redundant data
- Decide whether a particular relation R is in "good" form.
- In the case that a relation R is not in "good" form, decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Normalization is based on:
  - **functional dependencies**
  - **multivalued dependencies**

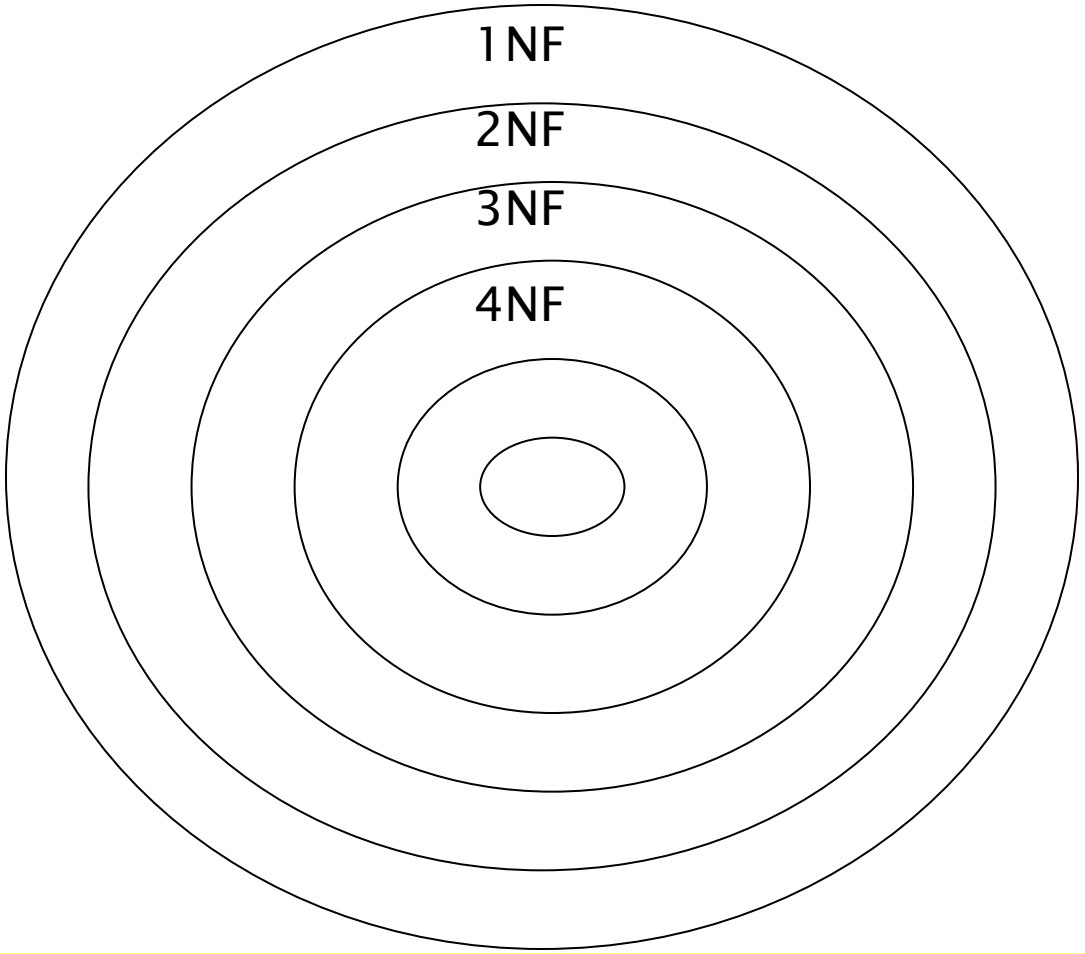
Levels of Normalization

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)



Most databases should be 3NF or BCNF in order to avoid the database anomalies.

Levels of Normalization



Each higher level is a subset of the lower level

First Normal Form

- Historically, it is designed to disallow
  - composite attributes
  - multivalued attributes
  - Or the combination of both

- All the values need to be **atomic**

First Normal Form

- Domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **first normal form** if the domains of all attributes of R are atomic
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data

First Normal Form (Cont'd)

- Atomicity is actually a property of how the elements of the domain are used.
  - Example: Strings would normally be considered indivisible
  - Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
  - If the first two characters are extracted to find the department, the domain of roll numbers is not atomic.

First Normal Form

- Steps:

1. Place all items that appear in the repeating group in a new table
2. Designate a primary key for each new table produced.
3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

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.

First Normal Form

Employee

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

EmployeeDegree

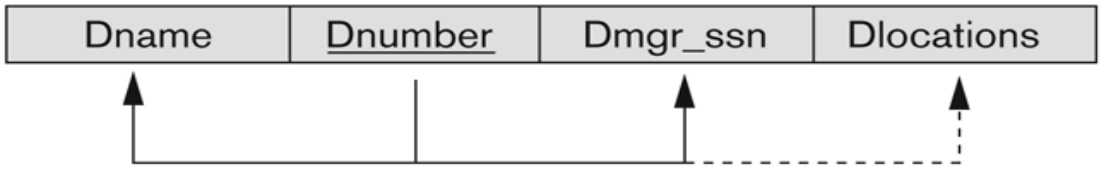
EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

An outer join between Employee and EmployeeDegree will produce the information we saw before

First Normal Form

(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.

Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements

- The database is in **first normal form**
- All **nonkey** attributes in the table must be fully functionally dependent on the entire primary key(That is, we don't have any partial functional dependency.)

Note: *Remember that we are dealing with non-key attributes*

Second Normal Form

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

- Steps:

1. If a data item is fully functionally dependent on only a part of the primary key, move that data item and that part of the primary key to a new table.
2. If other data items are functionally dependent on the same part of the key, place them in the new table also
3. Make the partial primary key copied from the original table the primary key for the new table. Place all items that appear in the repeating group in a new table

Second Normal Form

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

InvNum	LineNum	ProdNum	Qty	InvDate

InvNum, LineNum ⟶ ProdNum, Qty

InvNum ⟶ InvDate

There are two candidate keys

Invdate is the non-key attribute, and it is dependent on InvNum

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

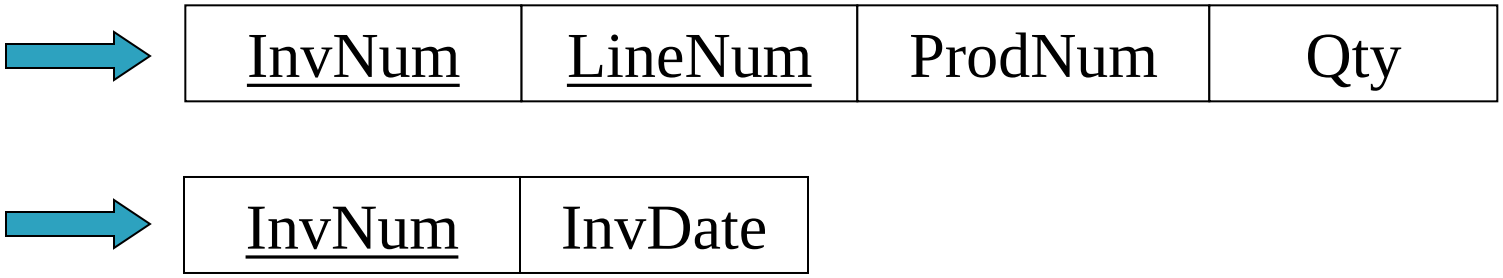
InvLine is only in **1NF**

Second Normal Form

InvLine

InvNum	LineNum	ProdNum	Qty	InvDate

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



Second Normal Form

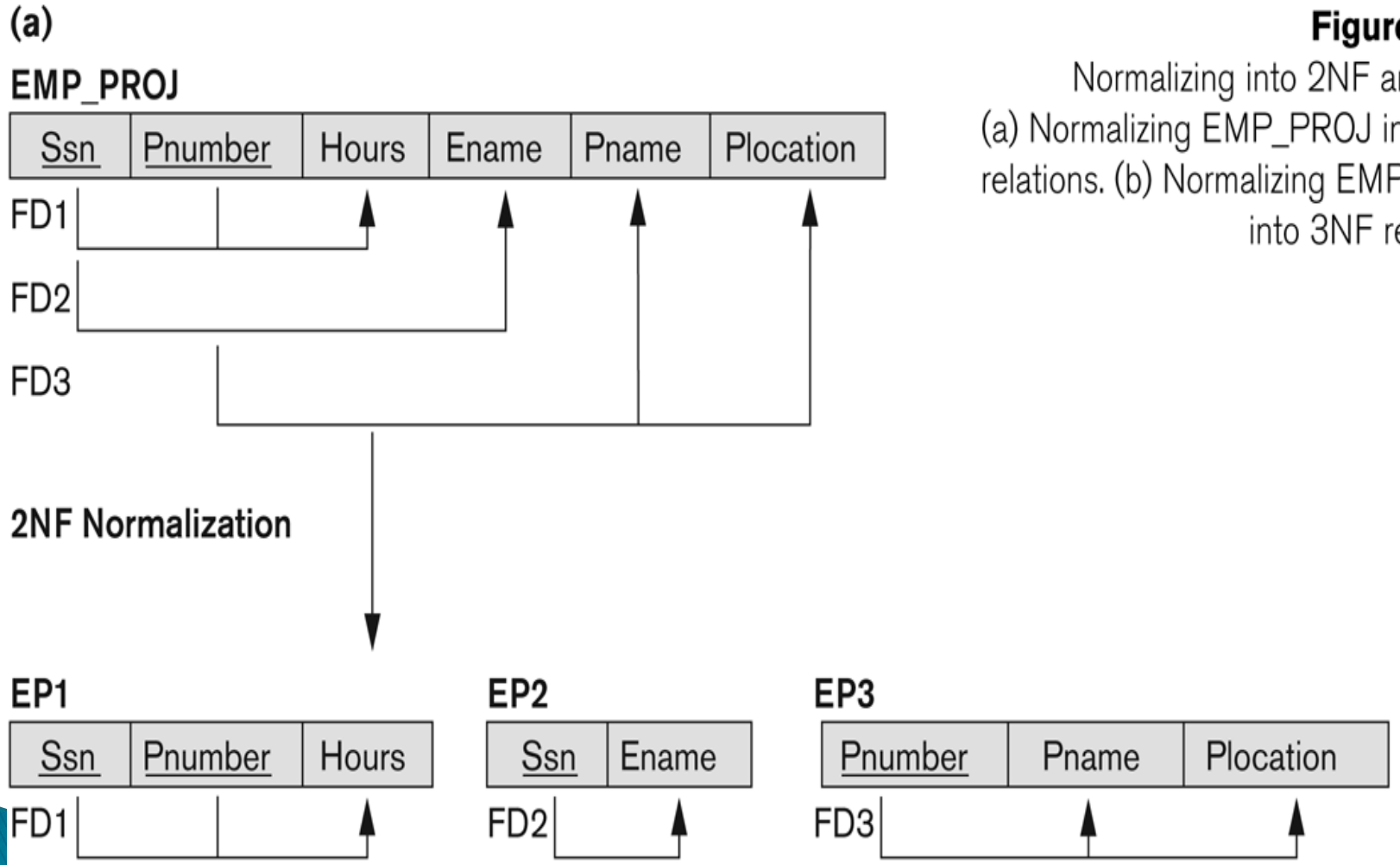


Figure 10.10
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Module 5

Relational–Database Design

Lecture 3

- Functional Dependencies
- Third Normal Form

Functional Dependencies

- *A* functional dependency is **trivial** if it is satisfied by all instances of a relation
  - Example:
    - $ID, name \rightarrow ID$
    - $name \rightarrow name$
  - In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

Closure of a Set of Functional Dependencies

- The set of all functional dependencies logically implied by F is the *closure* of F.
- We denote the *closure* of F by F^+.
- We can find all of F^+ by applying Armstrong's Axioms:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
  - if $\alpha \rightarrow \beta$, then $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(augmentation)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$
      and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - from $CG \rightarrow H$ *and* $CG \rightarrow I$: "union rule" can be inferred from
      - definition of functional dependencies, or
      - Augmentation of $CG \rightarrow I$ to infer $CG \rightarrow CGI$, augmentation of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

Closure of Functional Dependencies (Cont.)

- We can further simplify manual computation of F^+ by using the following **additional rules.**
  - If $\alpha \rightarrow \beta$ holds *a*nd $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\ \gamma$ holds **(union)**
  - If $\alpha \rightarrow \beta\ \gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds **(decomposition)**
  - If $\alpha \rightarrow \beta$ holds *a*nd $\gamma\ \beta \rightarrow \delta$ holds, then $\alpha\ \gamma \rightarrow \delta$ holds **(pseudotransitivity)**

  The above rules can be inferred from Armstrong's axioms.

Canonical Cover

- Sets of functional dependencies may have redundant dependencies that can be inferred from the others
  - Eg: $A \rightarrow C$ is redundant in: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$
  - Parts of a functional dependency may be redundant
    - E.g. on RHS: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow CD\}$ can be simplified to

      $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$
- Intuitively, a canonical cover of F is a "minimal" set of functional dependencies equivalent to F, with no redundant dependencies or having redundant parts of dependencies

Example of Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$
  - Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A is extraneous in $AB \rightarrow C$ because $B \rightarrow C$ logically implies $AB \rightarrow C$.
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow BC$ since $A \rightarrow BC$ is logically implied by $A \rightarrow B$ and $B \rightarrow$ C.
- The canonical cover is:

$$A \rightarrow B$$
$$B \rightarrow C$$

Extraneous Attributes

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.
  - Attribute A is extraneous in α if $A \in \alpha$ and F logically implies
    $$(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}.$$
  - Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.
- *Note:* implication in the opposite direction is trivial in each of the cases above, since a "stronger" functional dependency always implies a weaker one

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, AB \rightarrow C\}$
  - B is extraneous in $AB \rightarrow C$ because $A \rightarrow C$ logically implies $AB \rightarrow C$.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - C is extraneous in $AB \rightarrow CD$ since $A \rightarrow C$ can be inferred even after deleting C

Third Normal Form

- This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes. (transitivity)
- A relation is in **3NF** if the relation is in 2NF and all determinants of *non-key* attributes are candidate keys
  A relation in 3NF will not have any transitive dependencies of non-key attribute on a candidate key through another non-key attribute.
- For a table to be in 3NF, there are two requirements
  The table should be second normal form
  No attribute is transitively dependent on the primary key

Third Normal Form(3NF)

Third Normal Form is violated if:

- Second Normal Form is violated
- If there exists a non-key field(s) which is functionally dependent on another non-key field(s).

non-key ⟶ non-key

Note: A candidate key is not a non-key field.

Third Normal Form

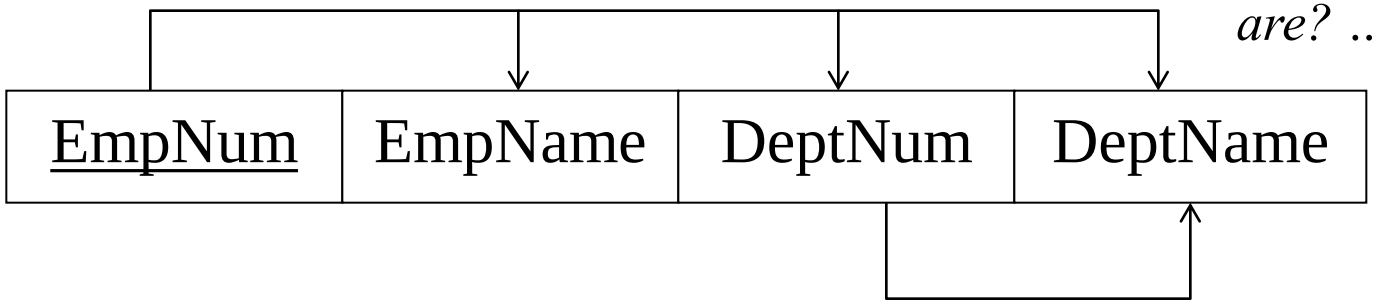
- Steps:

1. Move all items involved in transitive dependencies to a new entity.
2. Identify a primary key for the new entity.
3. Place the primary key for the new entity as a foreign key on the original entity.

Third Normal Form

Consider this **Employee** relation

Candidate keys are? …



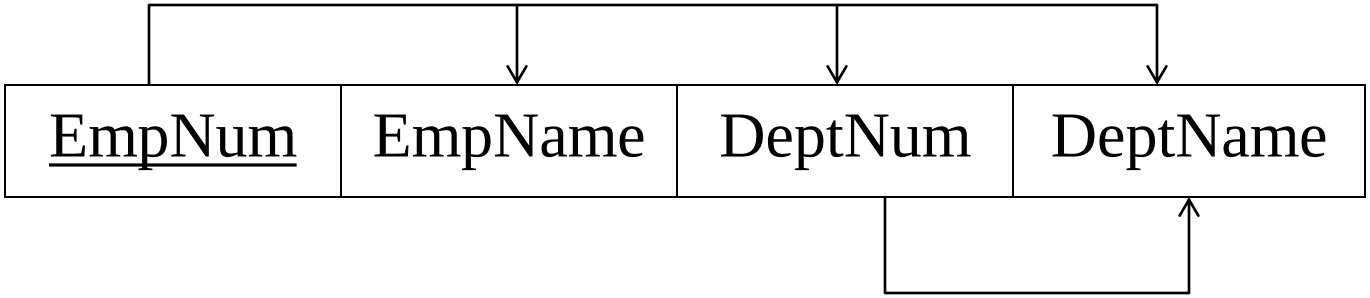
EmpName, DeptNum, and DeptName are non-key attributes.

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

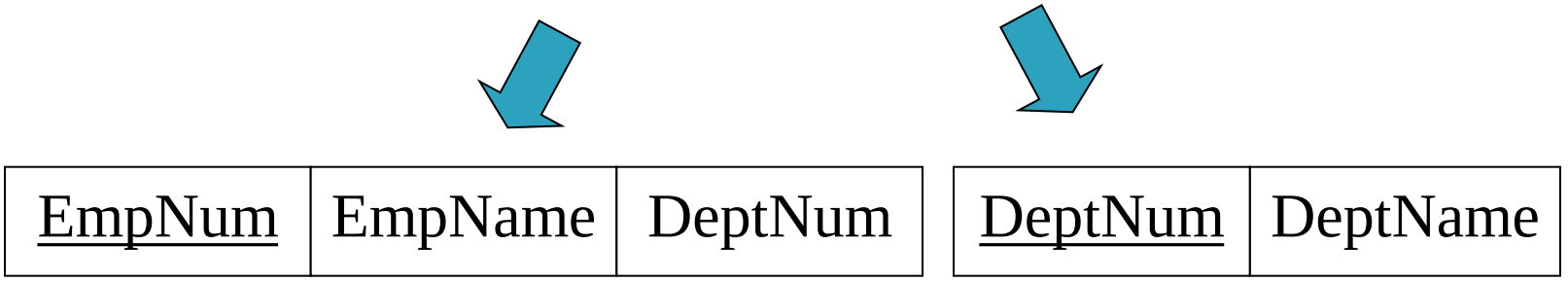
Is the relation in 3NF? … no

Is the relation in 2NF? … yes

Third Normal Form

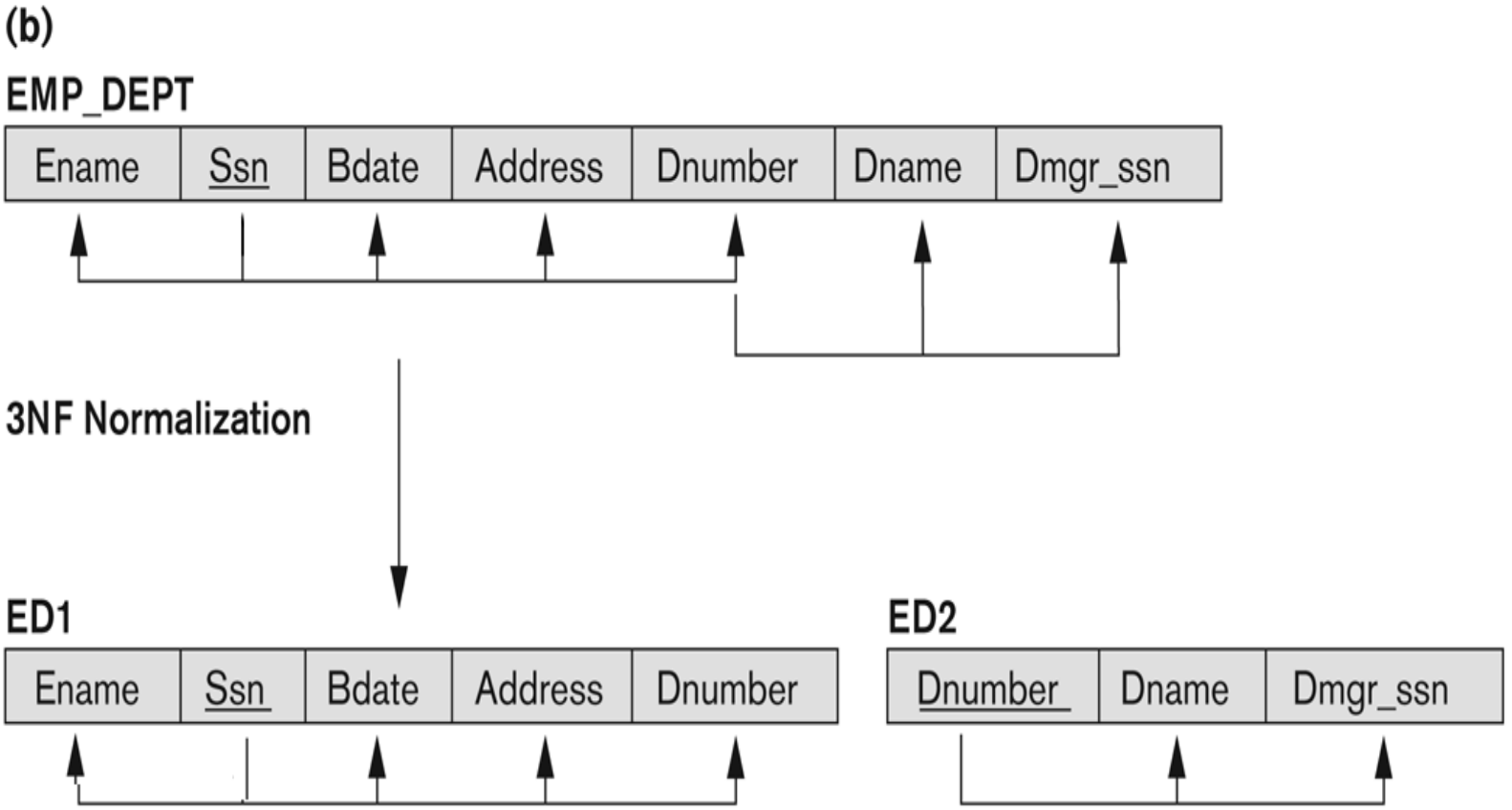


We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.



Verify these two relations are in 3NF.

Third Normal Form



3NF Example – Violation

Journal_Entry Table

JENO	DATE	DESCRIPTION
1	02-JAN-2003	Owner investment
2	03-JAN-2003	Borrowed money
3	03-JAN-2003	Purchased Supplies

Are there any non-key fields which functional determine another non-key field?

Are there any redundant facts?

Transactions Table

JENO	LINENO	ACCTNO	ACCTNAME	AMOUNT
1	1	100	Cash	20,000
1	2	310	Smith–Capital	20,000
2	1	100	Cash	30,000
2	2	220	Notes Payabl	30,000
3	1	120	Supplies	5,000
3	2	100	Cash	1,000
3	3	220	Notes Payabl	4,000

3NF Example – Violation
FD that indicates violation of 3NF

Anomalies if not corrected:

- **update** (if name of account 100 changes it must be changed in multiple places risking inconsistancy)
- **deletion** (can't delete JENO3 and its transactions without losing information about account 120)
- **insertion** (can't set up a new account, Jones-capital, for a new partner unless we first have a transaction involving that account.

Journal_Entry Table

JENO	DATE	DESCRIPTION
1	02-JAN-2003	Owner investment
2	03-JAN-2003	Borrowed money
3	03-JAN-2003	Purchased Supplies

JENO	LINENO	ACCTNO	ACCTNAME	AMOUNT
1	1	100	Cash	20,000
1	2	310	Smith-Capita	20,000
2	1	100	Cash	30,000
2	2	220	Notes Payabl	30,000
3	1	120	Supplies	5,000
3	2	100	Cash	1,000
3	3	220	Notes Payabl	4,000

3NF Example – Corrected

Accounts Table

ACCTNO	ACCTNAME
100	Cash
120	Supplies
220	Notes Payable
310	Smith-Capital

Journal_Entry Table

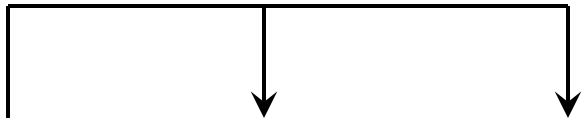
JENO	DATE	DESCRIPTION
1	02-JAN-2003	Owner investment
2	03-JAN-2003	Borrowed money
3	03-JAN-2003	Purchased Supplies

Transactions Table

JENO	LINENO	ACCTNO	AMOUNT
1	1	100	20,000
1	2	310	20,000
2	1	100	30,000
2	2	220	30,000
3	1	120	5,000
3	2	100	1,000
3	3	220	4,000

3NF Example – Corrected Final Dependencies

ACCTNO	ACCTNAME
100	Cash
120	Supplies
220	Notes Payable
310	Smith-Capital



JENO	DATE	DESCRIPTION
1	02-JAN-2003	Owner investment
2	03-JAN-2003	Borrowed money
3	03-JAN-2003	Purchased Supplies

JENO	LINENO	ACCTNO	AMOUNT
1	1	100	20,000
1	2	310	20,000
2	1	100	30,000
2	2	220	30,000
3	1	120	5,000
3	2	100	1,000
3	3	220	4,000

All non-key fields are FD on the PK and only the PK.

Module 5

Relational–Database Design

Lecture 4

- Boyce-Codd Normal Form- BCNF

Boyce-Codd Normal Form (BCNF)

BCNF is a refinement of the third normal form in which **it drops the restriction of a non-key attribute from the 3rd normal form.**

- a relation is in BCNF if it is in 3NF and if every determinant is a candidate key.
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Boyce-Codd Normal Form (BCNF)

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

- It should be in the **Third Normal Form**.
- And, for any dependency A → B, A should be a **super key**.
- It means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

BCNF Normal Form

Boyce-Codd Normal Form is violated if:

- Third Normal Form is violated
- If there exists a partial key which is functionally dependent on a non-key field(s).

  non-key $\longrightarrow$ partial-key

Decomposing a Schema into BCNF

Suppose we have a schema R and a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF.

We decompose R into:

- $(\alpha \cup \beta)$
- $(R - (\beta - \alpha))$

BCNF Example Semantics

- A student can have more than one major
- A student has a different advisor for each major.
- Each advisor advises for only one major.

BCNF Example – Violation

Student_Majors Table

SID	MAJOR	ADVISOR
1	PHYSICS	EINSTEIN
1	BIOLOGY	LIVINGSTON
2	PHYSICS	BOHR
2	COMPUTER SCIENCE	CODD
3	PHYSICS	EINSTEIN
4	BIOLOGY	LIVINGSTON
4	ACCOUNTING	PACIOLI
5	PHYSICS	EINSTEIN
6	PHYSICS	BOHR
6	BIOLOGY	DARWIN
7	COMPUTER SCIENCE	CODD
7	BIOLOGY	DARWIN

Does this relation violate third normal form?
Are there any redundant facts?

BCNF Example – Violation
FD that violates BCNF

It is important that you convince yourself that major does not FD advisor.

SID	MAJOR	ADVISOR
1	PHYSICS	EINSTEIN
1	BIOLOGY	LIVINGSTON
2	PHYSICS	BOHR
2	COMPUTER SCIENCE	CODD
3	PHYSICS	EINSTEIN
4	BIOLOGY	LIVINGSTON
4	ACCOUNTING	PACIOLI
5	PHYSICS	EINSTEIN
6	PHYSICS	BOHR
6	BIOLOGY	DARWIN
7	COMPUTER SCIENCE	CODD
7	BIOLOGY	DARWIN

BCNF Example – Corrected

Advisors Table

ADVISOR	MAJOR
BOHR	PHYSICS
CODD	COMPUTER SCIENCE
DARWIN	BIOLOGY
EINSTEIN	PHYSICS
LIVINGSTON	BIOLOGY
PACIOLI	ACCOUNTING

Note that the if the original key, counter-intuitively, in schema 1 had been defined as SID & ADVISOR this would have been a 2NF violation.

Student_Advisors Table

SID	ADVISOR
1	EINSTEIN
1	LIVINGSTON
2	BOHR
2	CODD
3	EINSTEIN
4	LIVINGSTON
4	PACIOLI
5	EINSTEIN
6	BOHR
6	DARWIN
7	CODD
7	DARWIN

BCNF EXAMPLE

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

BCNF EXAMPLE

- **Functional dependencies in the table above**:
  emp_id -> emp_nationality
  emp_dept -> {dept_type, dept_no_of_emp}
- **Candidate key**: {emp_id, emp_dept}
- The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

BCNF EXAMPLE

To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

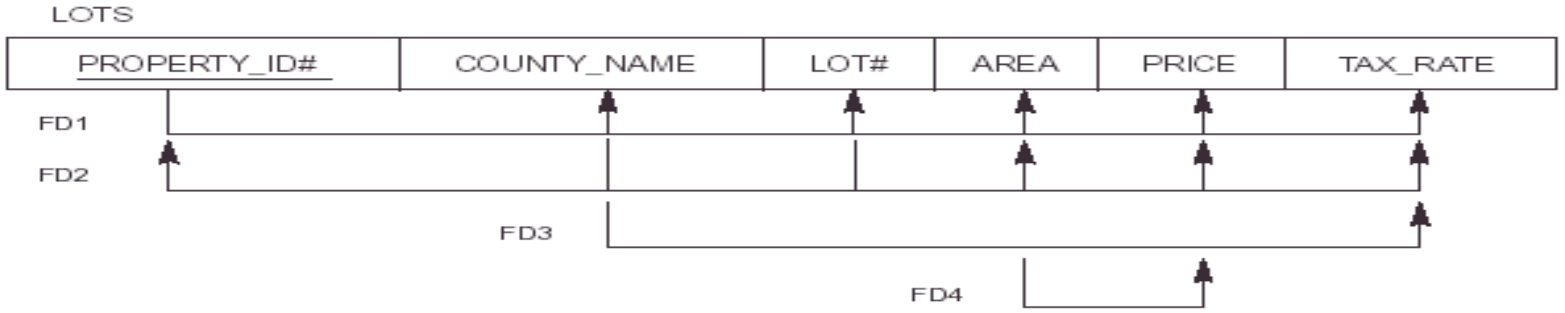
BCNF EXAMPLE

emp_dept_mapping table:

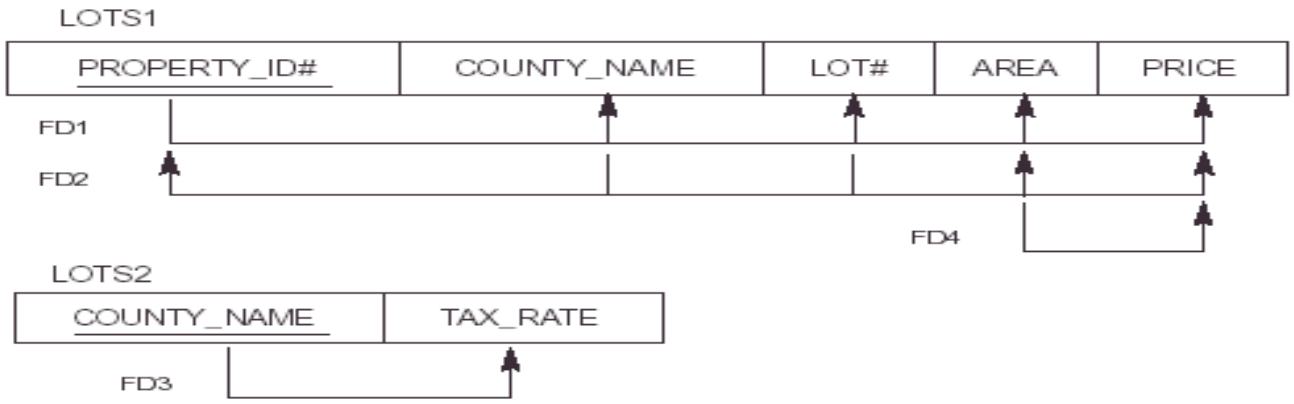
emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}
This is now in BCNF as in both the functional dependencies left side part is a key.

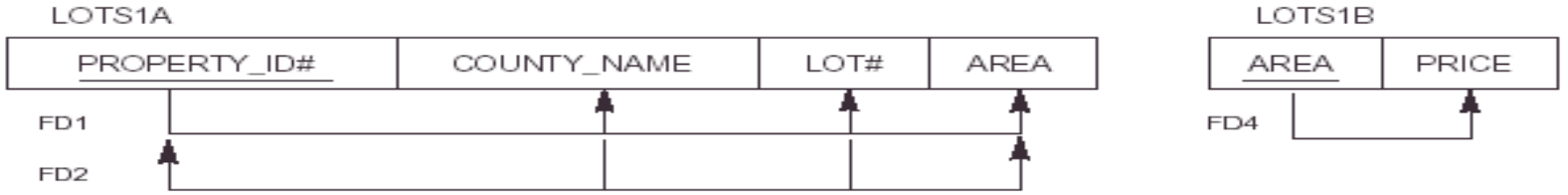
(a)


LOTS
PROPERTY_ID#
COUNTY_NAME
LOT#
AREA
PRICE
TAX_RATE
FD1
FD2
FD3
FD4

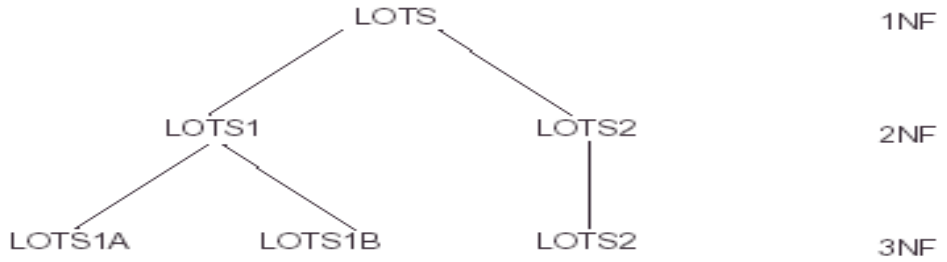
(b)


LOTS1
PROPERTY_ID#
COUNTY_NAME
LOT#
AREA
PRICE
FD1
FD2
FD4
LOTS2
COUNTY_NAME
TAX_RATE
FD3

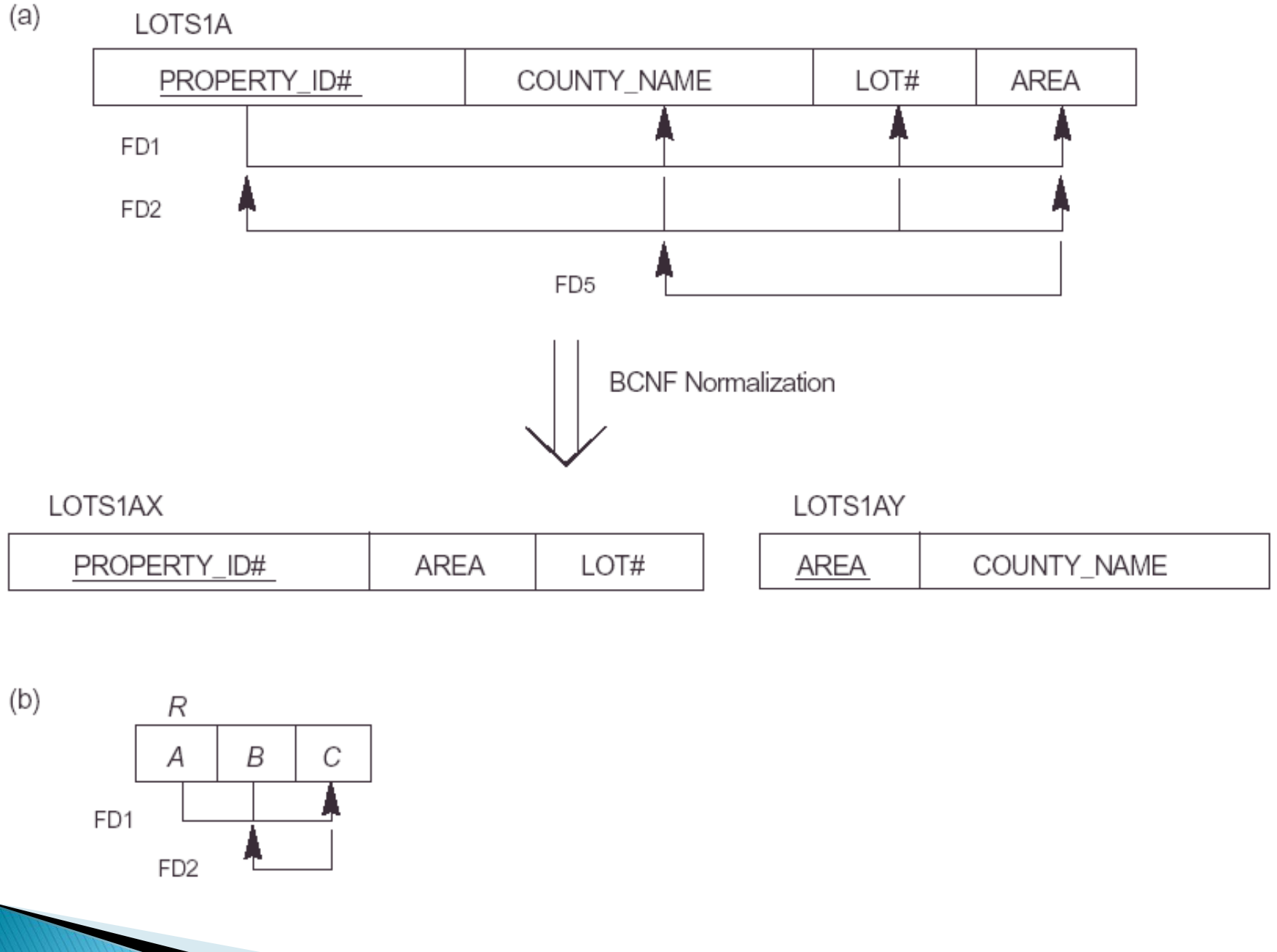
(c)


LOTS1A
PROPERTY_ID#
COUNTY_NAME
LOT#
AREA
FD1
FD2
LOTS1B
AREA
PRICE
FD4

(d)


LOTS
LOTS1
LOTS2
LOTS1A
LOTS1B
LOTS2
1NF
2NF
3NF

(a)


LOTS1A
PROPERTY_ID#
COUNTY_NAME
LOT#
AREA
FD1
FD2
FD5
BCNF Normalization
LOTS1AX
PROPERTY_ID#
AREA
LOT#
LOTS1AY
AREA
COUNTY_NAME

(b)

R
A
B
C
FD1
FD2

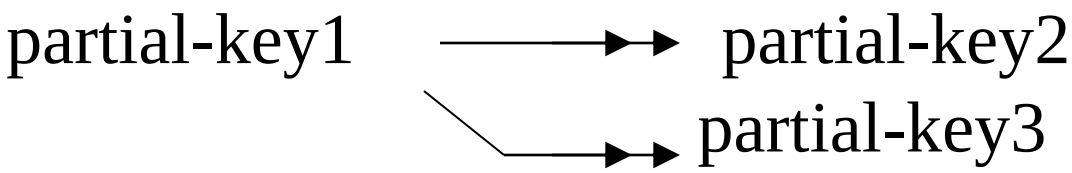
Fourth Normal Form (4NF)

- **Fourth normal form eliminates independent many-to-one relationships between columns.**
- **To be in Fourth Normal Form,**
  - **a relation must first be in Boyce-Codd Normal Form(BCNF).**
  - **a given relation may not contain more than one multi-valued attribute.**

Fourth Normal Form (4NF)

4th Normal Form is violated if:

- Boyce Codd Normal Form is violated
- If there exists a partial key which has multiple independent multi-valued functional dependencies to other partial keys.



Fourth Normal Form (4NF)

- Steps:

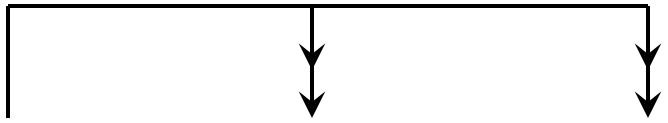
1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

4NF Example – Violation

Instruments_Languages

Name	Instrument	Language
Fred	Piano	French
Fred	Flute	Italian
Fred	Flute	Spanish
Jane	Piano	French
Jane	Oboe	French
Sam	Piano	French
Sam	Oboe	Spanish
Sam	Flute	Spanish

4NF Example – Violation



Name	Instrument	Language
Fred	Piano	French
Fred	Flute	Italian
Fred	Flute	Spanish
Jane	Piano	French
Jane	Oboe	French
Sam	Piano	French
Sam	Oboe	Spanish
Sam	Flute	Spanish

Does this relation violate 1st, 2nd, 3rd, or BCNF?
Are there any redundant facts?

4NF Example – Correction

LanguagesSpoken

Name	Language
Fred	French
Fred	Italian
Fred	Spanish
Jane	French
Sam	French
Sam	Spanish

InstrumentsPlayed

Name	Instrument
Fred	Piano
Fred	Flute
Jane	Piano
Jane	Oboe
Sam	Piano
Sam	Oboe
Sam	Flute

Fourth Normal Form (4NF)

Example (Not in 4NF)

Scheme → {Employee, Skill, ForeignLanguage}

1. **Primary Key → {Employee, Skill, Language }**
2. **Each employee can speak multiple languages**
3. **Each employee can have multiple skills**
4. **Thus violates 4NF**

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

4NF - Decomposition

Example (Convert to 4NF)

Old Scheme → {Employee, Skill, ForeignLanguage}

New Scheme → {Employee, Skill}

New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish