CS 422 - DBMS

Lesson 5 - Chapter 14

Normalization

Infinite Correlation in the Unified Field

WHOLENESS OF THE LESSON

The ideal database design will put each "fact" in only one place and will correctly maintain all relationships amongst the data. Once the data is efficiently organized, we can find answers to our queries with the least amount of efforts. Science & Technology of Consciousness: The principle of least action is the basic design principle used by Nature.

Normalization

- Normalization is a database design technique to organize large amounts of data efficiently so that a fact is stored at one place only.
- Normalization helps to reduce data redundancy.
- **Benefits of Minimizing Data Redundancy**
  - Updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
  - Reduction in the file storage space required by the base relations thus minimizing costs.
  - Database will be easier for the user to access and maintain.
  - Avoids problems like update anomalies.

How Normalization Supports Database Design

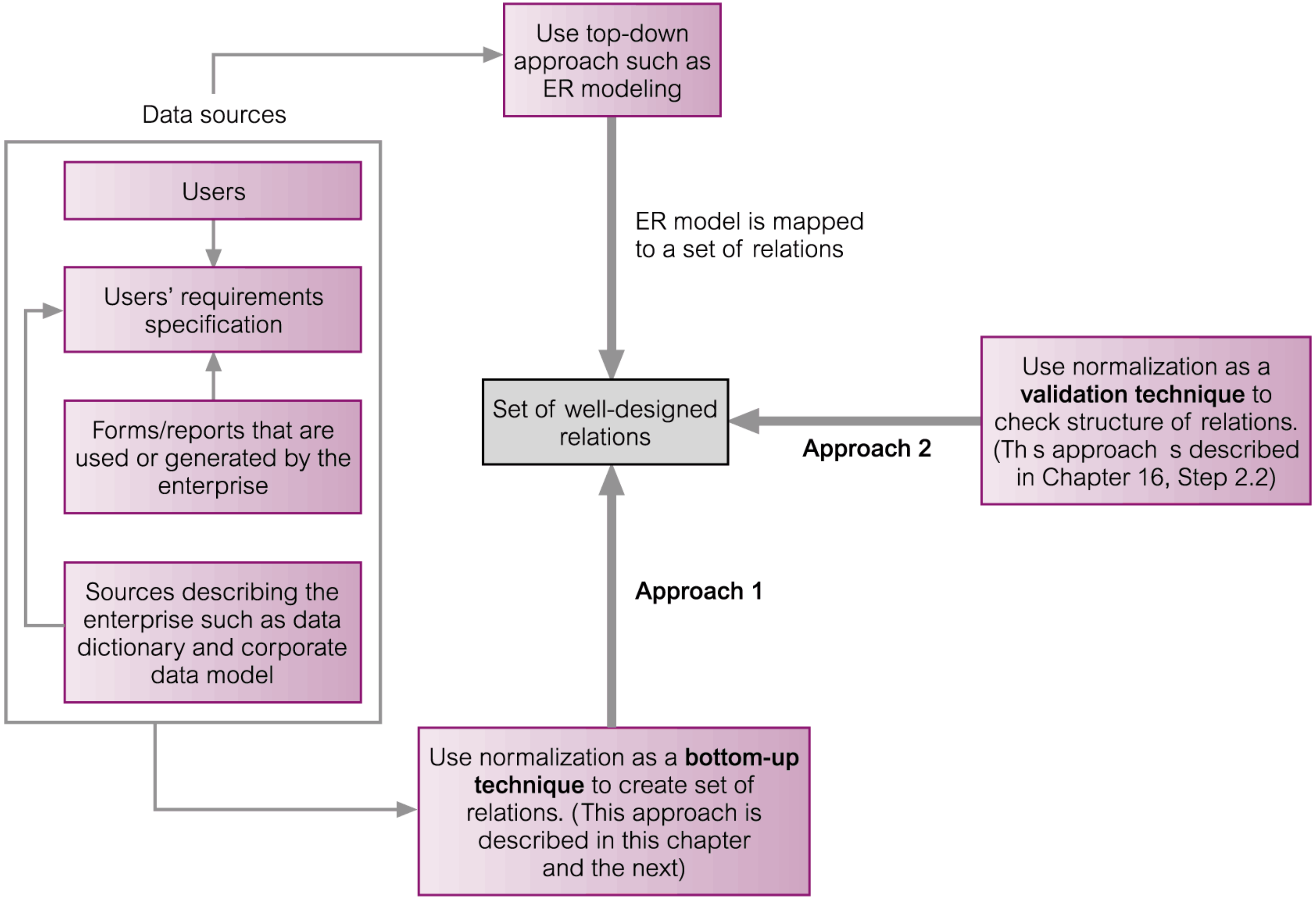


Figure 13.1 How normalization can be used to support database design.

Purpose of Normalization

- Purpose of Normalization is to produce a set of suitable relations that support the data requirements of an enterprise.
- **Characteristics of a suitable set of relations include:**
  - The *minimal* number of attributes necessary to support the data requirements of the enterprise;
  - Attributes with a close logical relationship are found in the same relation;
  - *Minimal* redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

Data Redundancy Problems

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.
- In contrast, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation, to represent where each member of staff is located.

Data Redundancy

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Data Redundancy and Update Anomalies

- Relations that contain redundant information may potentially suffer from update anomalies.

- Types of update anomalies include
  - **Insertion**
  - **Deletion**
  - **Modification**

Update Anomalies - Insertion

- To insert details of new staff into the StaffBranch relation:
  - For the staff located at branch number B007, we must enter the correct details of this branch so that these details will be consistent with other tuples in this relation.
- To insert details of a new branch which has not assigned any staff yet:
  - Violates entity integrity as staffNo being a PK cannot be null.
  - So cannot insert a new branch unless there are staff in that branch!

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Update Anomalies - Insertion

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337
5	Ekon	CSE	Mr. X	53337

We have to enter branch, hod and office_tel for every student we add to the table

Update Anomalies - Deletion

- If we delete a tuple from the StaffBranch relation that represents the last member of staff located at a branch, the details about that branch are also lost from the database.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Update Anomalies - Modification

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	~~Mr. X~~ Mr. Y	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

When Mr. X leaves the department and is replaced by Mr. Y we have to update all records

Easy to forget a record

What if we miss one? What if someone adds a new value while we are performing this update? What if someone overwrites the value, or does something else to one of the rows, which means our update didn't work?

Update Anomalies - Modification

- Updating an address of a branch in the *StaffBranch* relation may leave the database in an inconsistent state if the update is not done properly to all the tuples with that branch number.

- So to avoid all these anomalies, it's required to decompose the StaffBranch table into 2 separate tables *Staff* and *Branch* with the process of Normalization.

Normalized Database

ID	Department Name
1	Customer Support
2	Finance
3	Sales

ID	Location Name
1	Chicago
2	Boston
3	North Chicago
4	Portland

Employee ID	Name	Department	Location
1	John Smith	2	1
2	Mary Taylor	1	2
3	Rebecca Jones	3	1
4	Tony Adams	2	3
5	Sarah Johnson	3	4

- Easy to update “Finance” to say “Accounting”
  - Other tables are not involved
- When we delete employee Mary Taylor from the system, we still have a record of the Customer Support department.

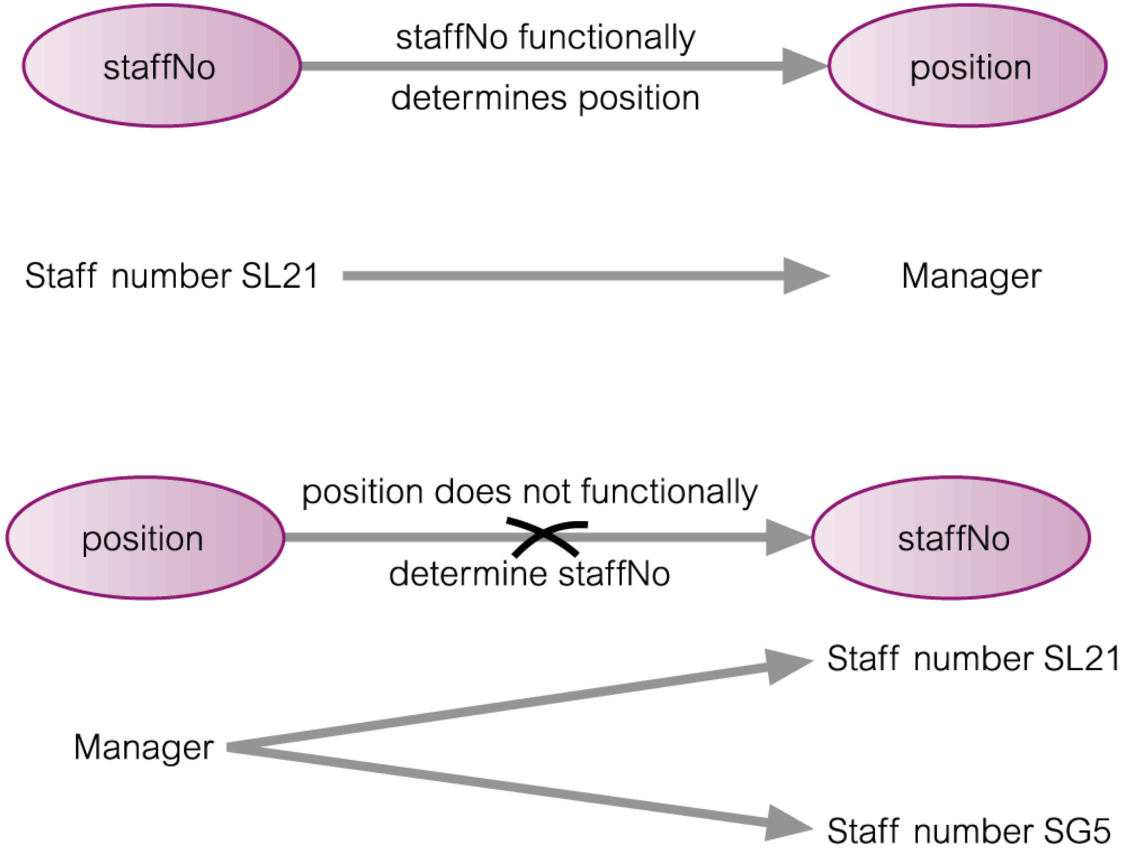
Functional Dependencies

- Functional dependency describes relationship between attributes where knowing the value of one attribute (or a set of attributes) is enough to find out the value of another attribute (or set of attributes) in the same table.
- For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with exactly one value of B in R.
- In other words, a functional dependency $A \rightarrow B$ in a relation exists if two tuples having the same value for A also has the same value for B (i.e A uniquely determines B).

An Example of Functional Dependency

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London



The **determinant** of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.

Example Functional Dependency that holds true for all Time

- Based on sample data, the following functional dependencies appear to hold.

  staffNo → sName

  sName → staffNo

- However, the only functional dependency that remains true *for all possible values* for the staffNo and sName attributes of the Staff relation is:

  staffNo → sName

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Identifying Functional Dependencies

- Identifying all functional dependencies between a set of attributes is relatively simple if the meaning of each attribute and the relationships between the attributes are well understood.
- This information should be provided by the enterprise in the form of discussions with users and/or documentation such as the users' requirements specification.
- However, if the users are unavailable for consultation and/or the documentation is incomplete then depending on the database application it may be necessary for the database designer to use their common sense and/or experience to provide the missing information.

Another Example of FD

Patient		
ssn	firstName	lastName
235-14-7854	Sandra	Smith
192-48-0924	John	Moore
821-13-2108	Laura	Turner
874-72-0093	John	Moore

ssn $\rightarrow$ firstName
ssn $\rightarrow$ lastName
ssn $\rightarrow$ firstName, lastName

Characteristics of Functional Dependencies

- There is a *one-to-one relationship* between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency.
- Holds for *all* time.
- The determinant has the *minimal* number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side.
    - **This requirement is called *full functional dependency.***

Example - Identify FDs for the StaffBranch Relation

- Assume that position held and branch determine a member of staff's salary.
- There is only one branch at a particular address.

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Example - Identify FDs for the StaffBranch Relation contd..

- The FDs for the StaffBranch relation are:

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Identifying Primary Key for a Relation using FDs

- Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.
- An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.

Example - Identify Primary Key for StaffBranch Relation

- To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation OR that functionally determines all other attributes.
- All attributes that are not part of a candidate key should be functionally dependent on the key.
- The only candidate key and therefore primary key for StaffBranch relation, is **staffNo**, as all other attributes of the relation are functionally dependent on staffNo.

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

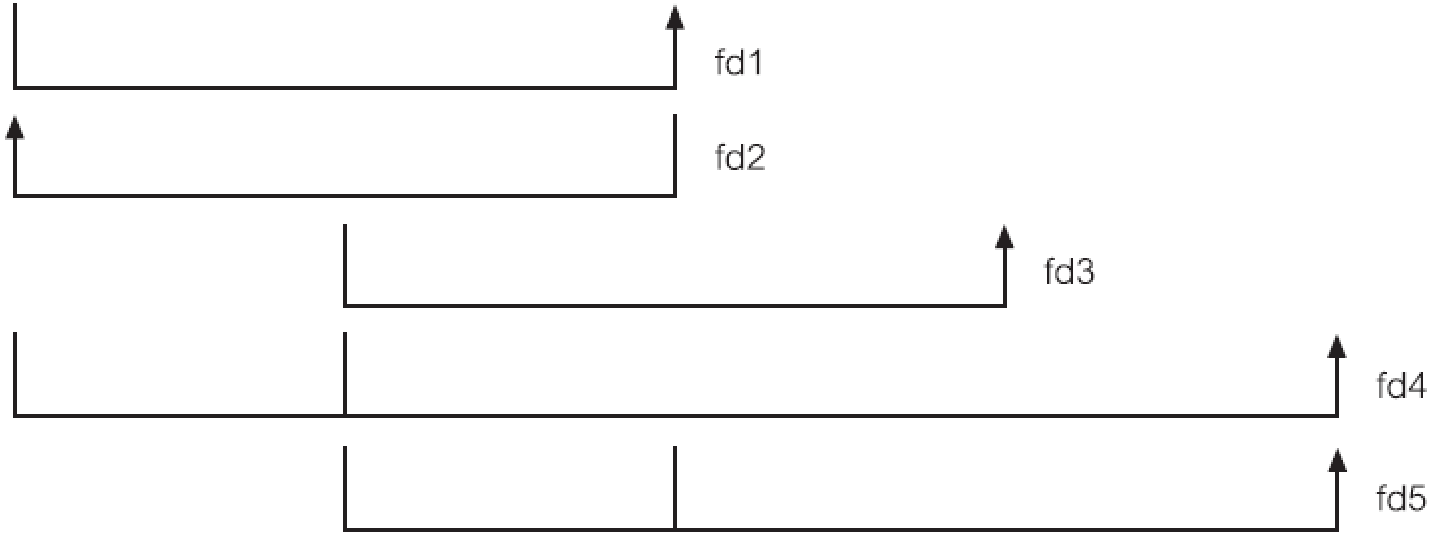
Example - Using sample data to identify functional dependencies

A	B	C	D	E
a	b	g	w	q
e	b	k	w	p
a	d	g	w	t
e	d	k	w	q
a	f	g	s	t
e	f	k	s	t

- Important to establish that sample data values shown in the relation are representative of *all possible values* that can be held by attributes A, B, C, D, and E. Assume true despite the relatively small amount of data shown in this relation.

Example - Using sample data to identify functional dependencies contd..

A	B	C	D	E
a	b	z	w	q
e	b	r	w	p
a	d	z	w	t
e	d	r	w	q
a	f	z	s	t
e	f	r	s	t



Example - Using sample data to identify functional dependencies contd..

- Functional dependencies between attributes A to E in the Sample relation.

$A \rightarrow C$	(fd1)
$C \rightarrow A$	(fd2)
$B \rightarrow D$	(fd3)
$(A, B) \rightarrow E$	(fd4)
$(B, C) \rightarrow E$	(fd5)

Example - Identifying Primary Key for Sample Relation

- Sample relation has five functional dependencies.
- The determinants in the Sample relation are A, B, C, (A, B) and (B, C). However, the only determinants that functionally determine all the other attributes of the relation are (A,B) & (B,C).
  - The attributes that make up the determinant (A, B) can determine all the other attributes in the relation either separately as A or B or together as (A, B).
- (A, B) is identified as the primary key for this relation.

Types of Functional Dependencies

- **Full functional dependency**
- **Partial dependency**
- **Transitive dependency**

Full Functional Dependency

- Determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side.
- A Functional Dependency $A \rightarrow B$ is a *Full Functional Dependency* if removal of any attribute from A results in the dependency no longer existing.

Example of Full Functional Dependency

STOCKS (Symbol, Company, Headquarters, Date, ClosePrice)

Symbol	Company	Headquarters	Date	ClosePrice
MSFT	Microsoft	Redmond, WA	09/07/2023	23.96
MSFT	Microsoft	Redmond, WA	09/08/2023	23.93
MSFT	Microsoft	Redmond, WA	09/09/2023	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2023	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2023	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2023	24.33

- **(Symbol, Date) → ClosePrice** FFD
- **(Symbol, Date) → Company** **X FFD**

Partial Dependency

- A functional dependency A→B is a partial dependency if there is some attribute that can be removed from A and yet the dependency still holds.
- For the Normalization process, we'll consider A is PK.
- Note : Partial dependencies could exist in a relation only when there is a composite PK.

Example of Partial Dependency

STOCKS (<u>Symbol</u>, Company, Headquarters, <u>Date</u>, ClosePrice)

<u>Symbol</u>	Company	Headquarters	<u>Date</u>	ClosePrice
MSFT	Microsoft	Redmond, WA	09/07/2023	23.96
MSFT	Microsoft	Redmond, WA	09/08/2023	23.93
MSFT	Microsoft	Redmond, WA	09/09/2023	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2023	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2023	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2023	24.33

- **(Symbol, Date) → (Company, Headquarters)**

Partial Dependency

Because:

- Symbol → (Company, Headquarters)

Another Example of Partial Dependency

Teacher only depends on subject_id

student_id	subject_id	marks	teacher
10	1	70	Java Teacher
10	2	75	C++ Teacher
11	1	80	Java Teacher

Primary key

Transitive Dependencies

- Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if $A \rightarrow B$ and $B \rightarrow C$, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).
- In this situation, $B \rightarrow C$ is called as Transitive Dependency.
- Important to recognize a transitive dependency because its existence in a relation can potentially cause update anomalies.

Example of Transitive Dependency

- Consider the following FDs in the StaffBranch relation.

  - **staffNo → sName, position, salary, branchNo, bAddress**
  - **branchNo → bAddress**

- Transitive dependency, **branchNo → bAddress** exists on staffNo via branchNo.

MAIN POINT

A functional dependency (FD) describes a permanent semantic relationship between sets of attributes in a relation schema that all relation instances of that schema must adhere to. Science & Technology of Consciousness: Due to this permanence a functional dependency is functioning as a law of Nature. Vedic Science teaches respect for the laws of Nature and provides a simple technique of TM to bring action into accord with all the laws of Nature.

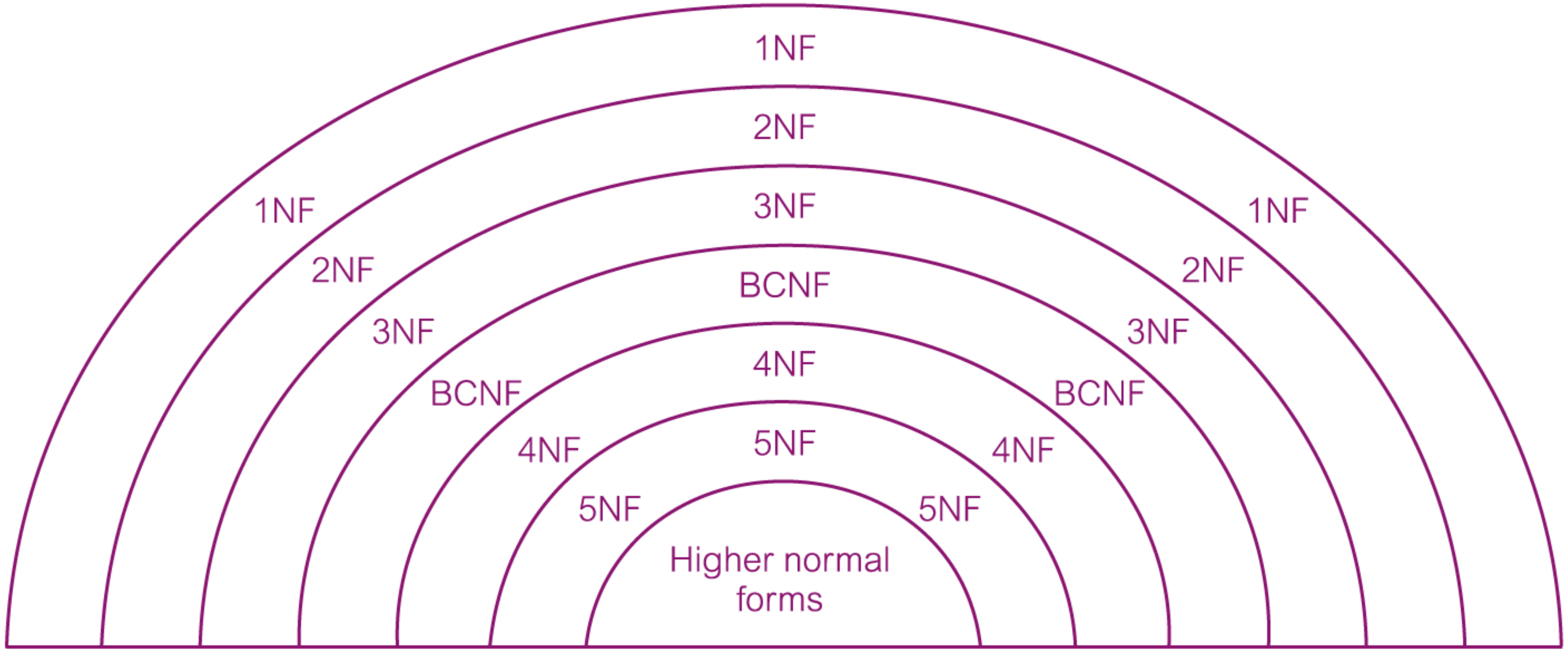
The Process of Normalization

- Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.

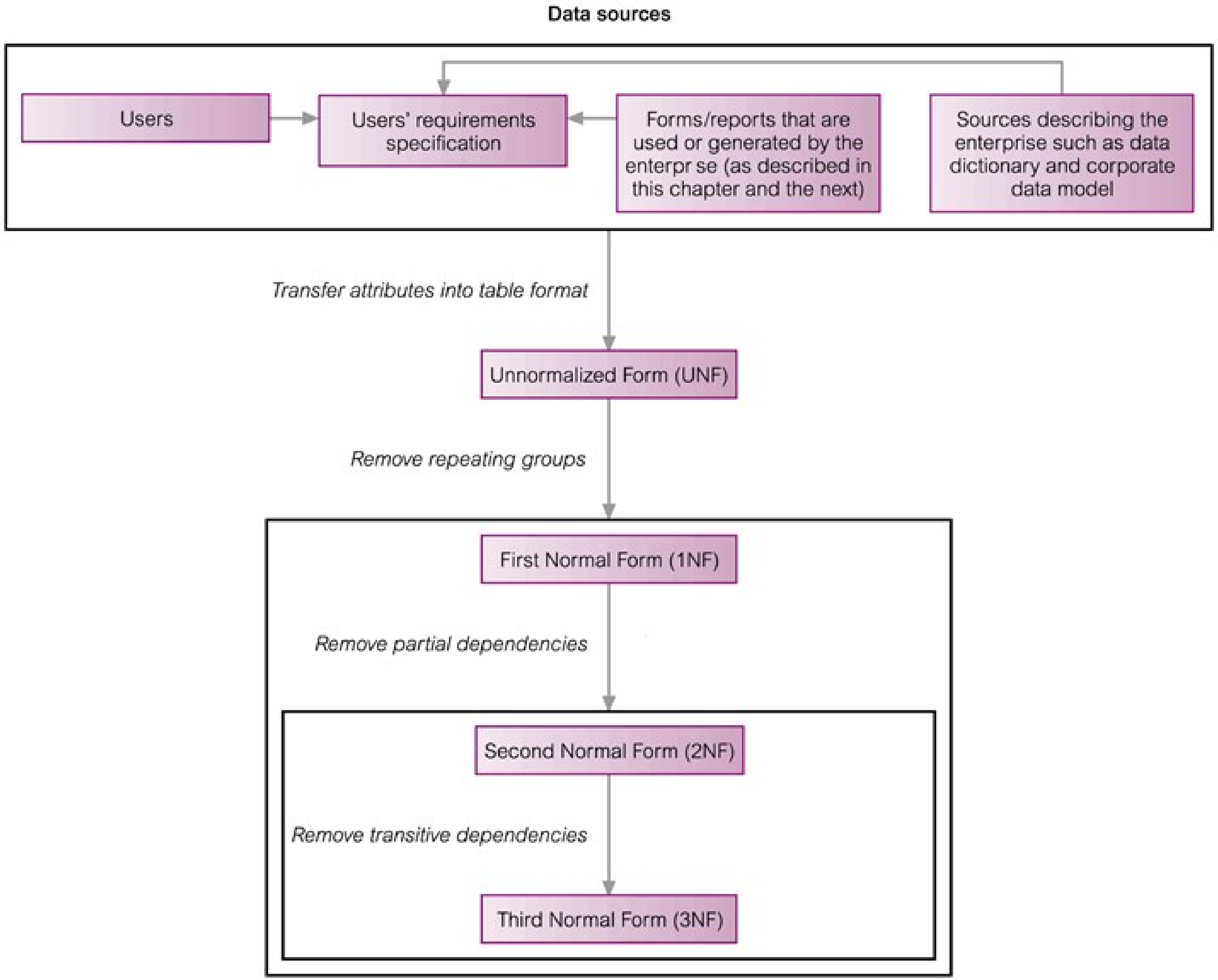
- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.

The Process of Normalization

- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.



The Process of Normalization



Unnormalized Form (UNF)

- To create an Unnormalized table:
  - Transform the data from the information source (e.g. a standard data entry form) into table format with columns and rows.
- A table is said to be Unnormalized if
  - It contains one or more repeating groups.
    - A repeating group is an attribute, or group of attributes, within a table that occurs with multiple values for a single occurrence of the nominated key attribute(s) for that table.
      - In this context, the term “key” refers to the attribute(s) that uniquely identify each row within the Unnormalized table.

Unnormalized Form (UNF) Examples

- Repeating groups are present.

SID	NAME	COURSES	BUILDING #	ADDRESS
101	John Doe	CS422	B002	22 North Veda
102	Ane Doe	CS422, CS465	B003	100 South Liberty Ave
103	Bob Rich	CS465	B003	100 South Liberty Ave
104	Frank Peter	CS422, CS446, CS465	B004	22 North Veda

First Normal Form (1NF)

A relation is said to be in 1NF only if the intersection of each row and column contains one and only one value.

No repeating groups

SID	NAME	COURSES	BUILDING #	ADDRESS
101	John Doe	CS422	B002	22 North Veda
102	Ane Doe	CS422	B003	100 South Liberty Ave
102	Ane Doe	CS465	B003	100 South Liberty Ave
103	Bob Rich	CS465	B003	100 South Liberty Ave
104	Frank Peter	CS422	B004	22 North Veda
104	Frank Peter	CS446	B004	22 North Veda
104	Frank Peter	CS465	B004	22 North Veda

UNF to 1NF conversion

- Nominate an attribute or group of attributes to act as the key for the unnormalized table.
- Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).
- Remove the repeating group by either

1. **Adding more rows (‘flattening’ the table).**
   - *Duplicating the non-repeating data*
   - *Adds more redundancy into the original UNF table.*

2. **Or by placing the repeating data along with a copy of the original key attribute(s) into a separate relation.**
   - *This approach is mainly used when the UNF table contains more then one repeating groups or repeating groups within repeating groups.*
   - *This approach moves the original UNF table further along the normalization process than approach 1.*

Second Normal Form (2NF)

- Based on the concept of full functional dependency.
- A relation is said to be in 2NF if it is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.
- All non-key fields depend on all components of the primary key.
  - **Guaranteed when PK is a single field.**

2NF = 1NF + No partial dependencies

1NF to 2NF conversion

1. Identify the functional dependencies in the relation.

2. Identify the primary key for the 1NF relation.

3. If partial dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant.

Second Normal Form (2NF)

- (SID, COURSES) -> BUILDING#, ADDRESS
- SID -> NAME
- BUILDING# -> ADDRESS

PK

PD

SID	NAME	COURSES	BUILDING #	ADDRESS
101	John Doe	CS422	B002	22 North Veda
102	Ane Doe	CS422	B003	100 South Liberty Ave
102	Ane Doe	CS465	B003	100 South Liberty Ave
103	Bob Rich	CS465	B003	100 South Liberty Ave
104	Frank Peter	CS422	B004	22 North Veda
104	Frank Peter	CS446	B004	22 North Veda
104	Frank Peter	CS465	B004	22 North Veda

Original 1NF Table

SID	COURSES	BUILDING #	ADDRESS
101	CS422	B002	22 North Veda
102	CS422	B003	100 South Liberty Ave
102	CS465	B003	100 South Liberty Ave
103	CS465	B003	100 South Liberty Ave
104	CS422	B004	22 North Veda
104	CS446	B004	22 North Veda
104	CS465	B004	22 North Veda

SID	NAME
101	John Doe
102	Ane Doe
103	Bob Rich
104	Frank Peter

2NF Tables

Another Example of 1NF to 2NF
1NF table is given

STOCKS (Company, Symbol, Headquarters, Date, Close_Price)

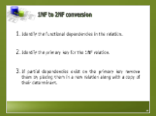
Symbol	Company	Headquarters	Date	Close Price
MSFT	Microsoft	Redmond, WA	09/07/2013	23.96
MSFT	Microsoft	Redmond, WA	09/08/2013	23.93
MSFT	Microsoft	Redmond, WA	09/09/2013	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2013	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2013	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2013	24.33

Example of 1NF to 2NF contd..

<u>Symbol</u>	Company	Headquarters	<u>Date</u>	Close Price
MSFT	Microsoft	Redmond, WA	09/07/2013	23.96
MSFT	Microsoft	Redmond, WA	09/08/2013	23.93
MSFT	Microsoft	Redmond, WA	09/09/2013	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2013	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2013	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2013	24.33

FD1: (Symbol, Date) $\rightarrow$ Company, Headquarters, Close Price

FD2: Symbol $\rightarrow$ (Company, Headquarters) => PD



Example of 1NF to 2NF

Decomposed 2NF tables after removing the PD

Company

Symbol	Company	Headquarters
MSFT	Microsoft	Redmond, WA
ORCL	Oracle	Redwood Shores, CA

Stock_Prices

Symbol	Date	Close Price
MSFT	09/07/2013	23.96
MSFT	09/08/2013	23.93
MSFT	09/09/2013	24.01
ORCL	09/07/2013	24.27
ORCL	09/08/2013	24.14
ORCL	09/09/2013	24.33

Third Normal Form (3NF)

- Based on the concept of transitive dependency.
- A relation is said to be in 3NF if it is in 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.
- No non-key field depends upon another.
  - **All non-key fields depend only on the PK.**

3NF = 2NF + No transitive dependencies

2NF to 3NF

- Identify the primary key in the 2NF relations.
- Identify functional dependencies in the relations.
- If transitive dependencies exist on the primary key then remove them by placing them in a new relation along with a copy of their determinant.

Third Normal Form (3NF)

- (SID, COURSES) -> BUILDING#, ADDRESS
- BUILDING# -> ADDRESS

PK

TD

SID	COURSES	BUILDING #	ADDRESS
101	CS422	B002	22 North Veda
102	CS422	B003	100 South Liberty Ave
102	CS465	B003	100 South Liberty Ave
103	CS465	B003	100 South Liberty Ave
104	CS422	B004	22 North Veda
104	CS446	B004	22 North Veda
104	CS465	B004	22 North Veda

2NF Table

SID	COURSES	BUILDING #
101	CS422	B002
102	CS422	B003
102	CS465	B003
103	CS465	B003
104	CS422	B004
104	CS446	B004
104	CS465	B004

BUILDING #	ADDRESS
B002	22 North Veda
B003	100 South Liberty Ave
B004	22 North Veda

SID	NAME
101	John Doe
102	Ane Doe
103	Bob Rich
104	Frank Peter

Final 3NF Tables

Book Example

- A collection of (simplified) *DreamHome* leases:

Book Example contd..

ClientRental unnormalized table:

ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	50	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-June-12	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

Book Example contd..

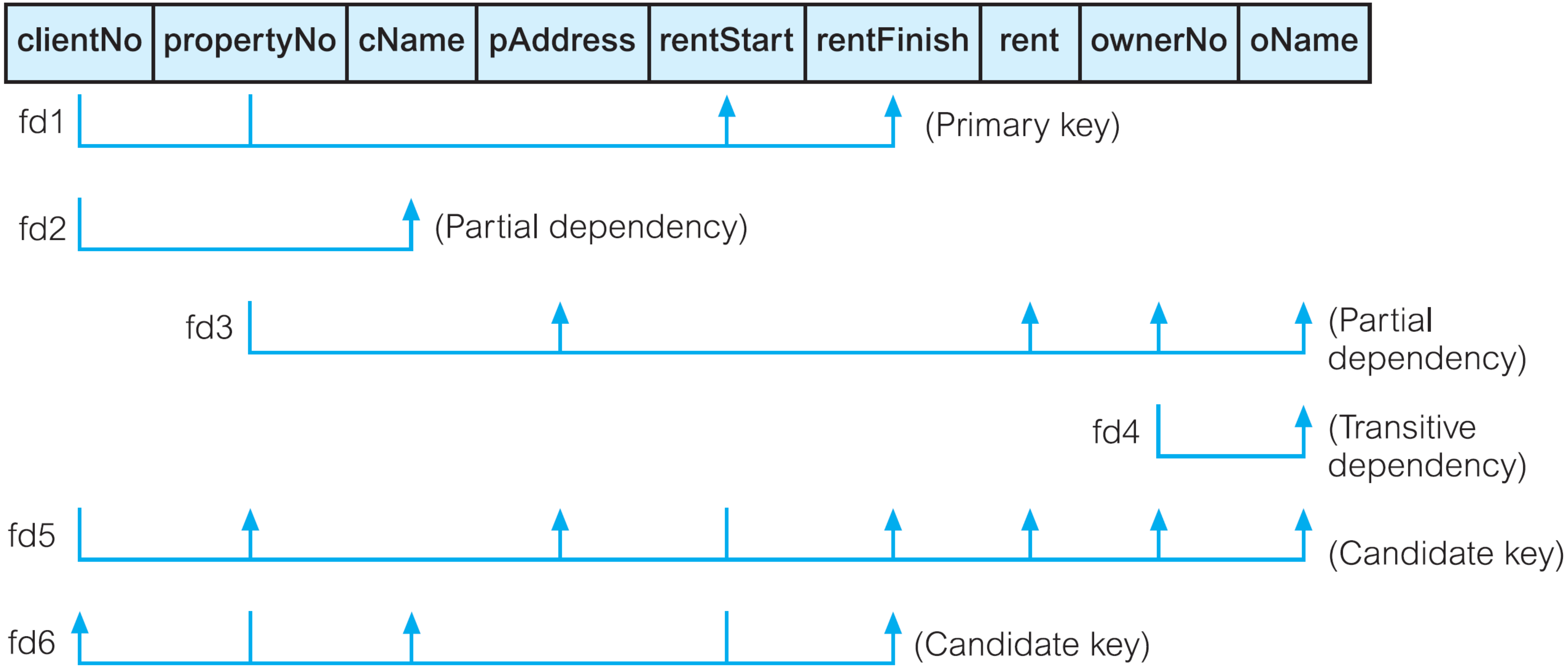
1NF ClientRental relation

ClientRental (clientNo, propertyNo, cName, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

Book Example contd..

Functional dependencies of the ClientRental relation



Book Example contd..

Functional dependencies of the ClientRental relation

fd1	**clientNo, propertyNo** → rentStart, rentFinish	(Primary key)
fd2	**clientNo** → cName	(Partial dependency)
fd3	**propertyNo** → pAddress, rent, ownerNo, oName	(Partial dependency)
fd4	**ownerNo** → oName	(Transitive dependency)
fd5	**clientNo, rentStart** → propertyNo, pAddress, rentFinish, rent, ownerNo, oName	(Candidate key)
fd6	**propertyNo, rentStart** → clientNo, cName, rentFinish	(Candidate key)

Book Example contd..

- Primary key is (clientNo, propertyNo).
- **Partial dependency:**
  clientNo → cName
  propertyNo → pAdress, rent, ownerNo, oName
- Remove partial dependencies by splitting ClientRental relation into three relations.

Book Example contd..

2NF relations derived from the ClientRental relation

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

Client (clientNo, cName)

Rental (clientNo, propertyNo, rentStart, rentFinish)

PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)

Book Example contd..

- The functional dependencies for the *Client*, *Rental*, and *PropertyOwner* relations, derived earlier are as follows:

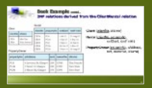
Client		
fd2	clientNo → cName	(Primary key)
Rental		
fd1	clientNo, propertyNo → rentStart, rentFinish	(Primary key)
fd5	clientNo, rentStart → propertyNo, rentFinish	(Candidate key)
fd6	propertyNo, rentStart → clientNo, rentFinish	(Candidate key)
PropertyOwner		
fd3	propertyNo → pAddress, rent, ownerNo, oName	(Primary key)
fd4	ownerNo → oName	(Transitive dependency)

2NF to 3NF Book Example contd..

- In the 2NF relations, primary key of *PropertyOwner* relation is propertyNo.
- There is a transitive dependency from propertyNo to oName via ownerNo (i.e. there is a functional dependency ownerNo → oName).
- Remove this transitive dependency by splitting *PropertyOwner* relation into two relations *Owner* and *PropertyForRent*.

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw



Book Example contd..

3NF relations derived from *PropertyOwner* relation

PropertyForRent (propertyNo, pAddress, rent, ownerNo)

Owner (ownerNo, oName)

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

Book Example contd..

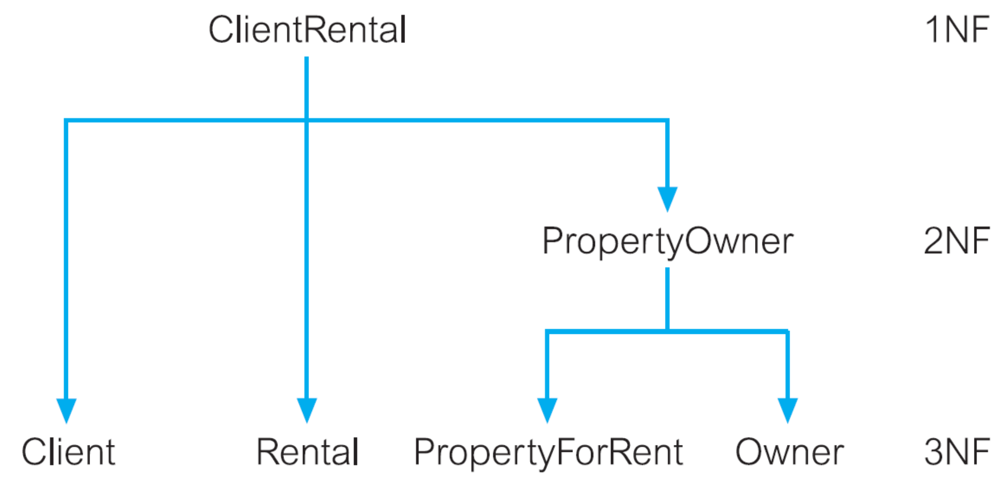
Resulting 3NF relations

Client (clientNo, cName)

Rental (clientNo, propertyNo, rentStart, rentFinish)

PropertyForRent (propertyNo, pAddress, rent, ownerNo)

Owner (ownerNo, oName)



Normal Forms Defined Informally

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