Normalization

Objectives

- The Purpose Of Normalization.
- Data Redundancy And Update Anomalies.
- Functional Dependency
- The Process Of Normalization
- The most commonly used Normal Forms, First Normal Form (1NF), Second Normal Form (2NF), And Third Normal Form (3NF).

Purpose of Normalization

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

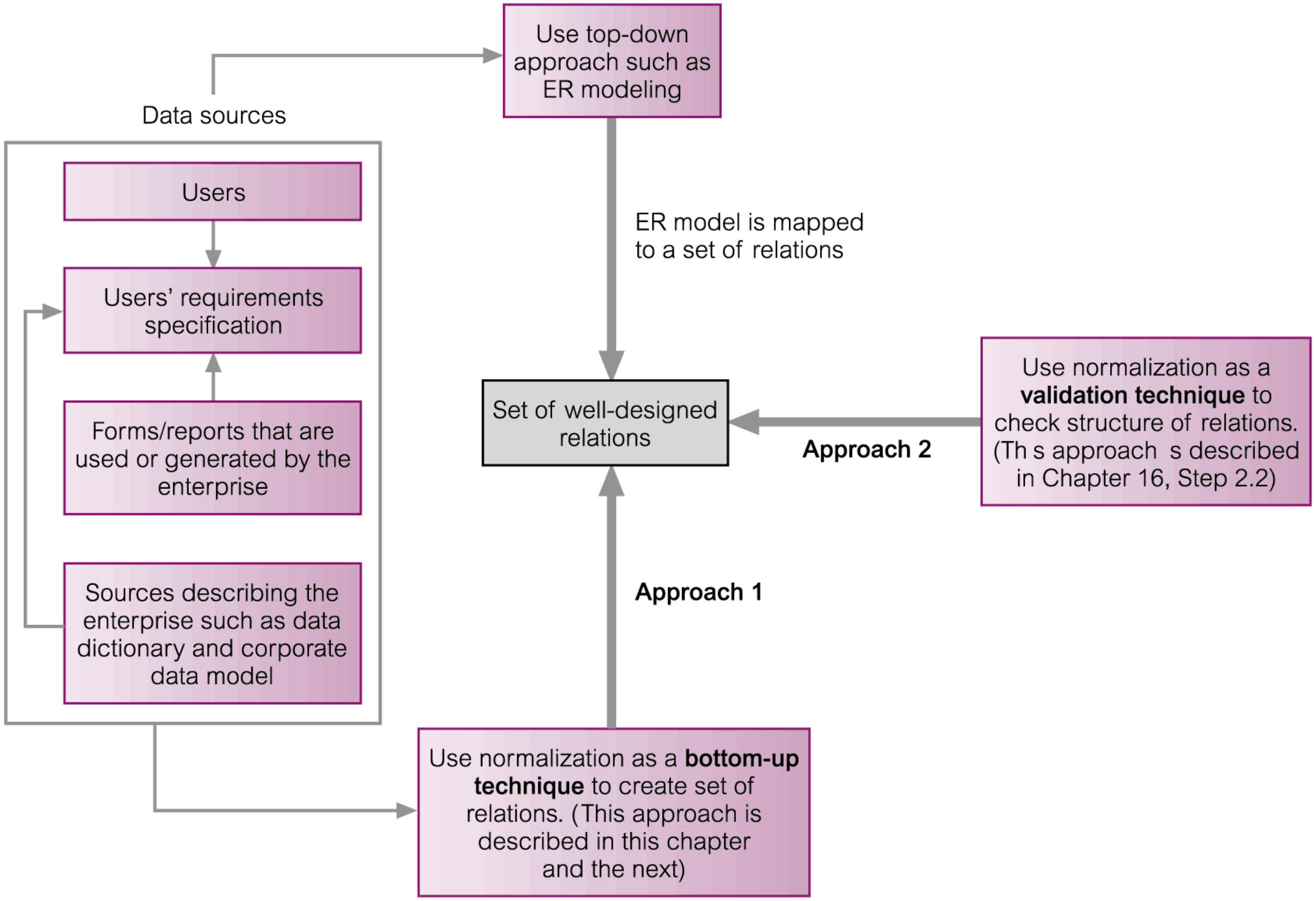
Purpose of Normalization

- 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

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 sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise
Sources describing the enterprise such as data dictionary and corporate data model
Use top-down approach such as ER modeling
ER model is mapped to a set of relations
Set of well-designed relations
Use normalization as a **validation technique** to check structure of relations. (Th s approach s described in Chapter 16, Step 2.2)
Approach 2
Approach 1
Use normalization as a **bottom-up technique** to create set of relations. (This approach is described in this chapter and the next)

Data Redundancy and Update Anomalies

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

Data Redundancy and Update Anomalies Example

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

Data Redundancy and Update Anomalies

- Relations that contain redundant information may potentially suffer from update anomalies.
- Types of update anomalies include
  - Insertion
  - Deletion
  - Modification

Lossless-join and Dependency Preservation Properties

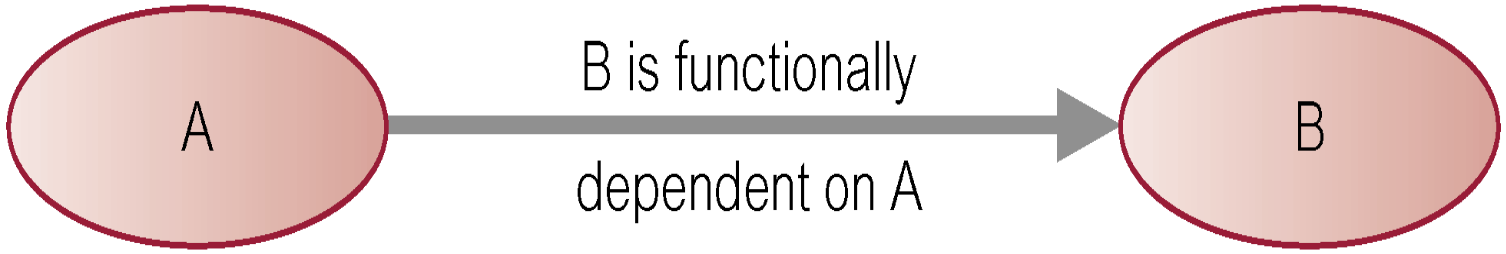
- Two important properties of decomposition.
  - Lossless-join property enables us to find any instance of the original relation from corresponding instances in the smaller relations.
  - Dependency preservation property enables us to enforce a constraint on the original relation by enforcing some constraint on each of the smaller relations.

Functional Dependencies

- Functional dependency describes relationship between attributes.
- For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted A   B), if each value of A in R is associated with exactly one value of B in R.

Characteristics of Functional Dependencies

- Property of the meaning or semantics of the attributes in a relation.
- Diagrammatic representation.



- 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 for all Time

- Consider the values shown in staffNo and sName attributes of the Staff relation (see Slide 8).
- Based on sample data, the following functional dependencies appear to hold.

staffNo → sName

sName → staffNo

Example Functional Dependency that holds for all Time

- 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

Characteristics of Functional Dependencies

- Determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side.
- This requirement is called full functional dependency.

Characteristics of Functional Dependencies

- functional dependency indicates that if A and B are attributes of a relation, B is fully functionally dependent on A, if B is functionally dependent on A, but not on any proper subset of A.

Example Full Functional Dependency

- Exists in the Staff relation (see Slide 8).

  staffNo, sName → branchNo

- True - each value of (staffNo, sName) is associated with a single value of branchNo.

- However, branchNo is also functionally dependent on a subset of (staffNo, sName), namely staffNo. Example above is a *partial dependency.*

Characteristics of Functional Dependencies

- Main characteristics of functional dependencies used in normalization:
  - 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.

Transitive Dependencies

- Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).

Example Transitive Dependency

- Consider functional dependencies in the StaffBranch relation (see Slide 8).

  staffNo → sName, position, salary, branchNo,
  bAddress

  branchNo → bAddress

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

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.

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.

Identifying Functional Dependencies

- 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.

Example - Identifying a set of functional dependencies for the StaffBranch relation

- Examine semantics of attributes in StaffBranch relation (see Slide 8).
- Assume that position held and branch determine a member of staff’s salary.

Example - Identifying a set of functional dependencies for the StaffBranch relation

- With sufficient information available, identify the functional dependencies for the StaffBranch relation as:

  staffNo → sName, position, salary, branchNo, bAddress

  branchNo → bAddress

  bAddress → branchNo

  branchNo, position → salary

  bAddress, position → salary

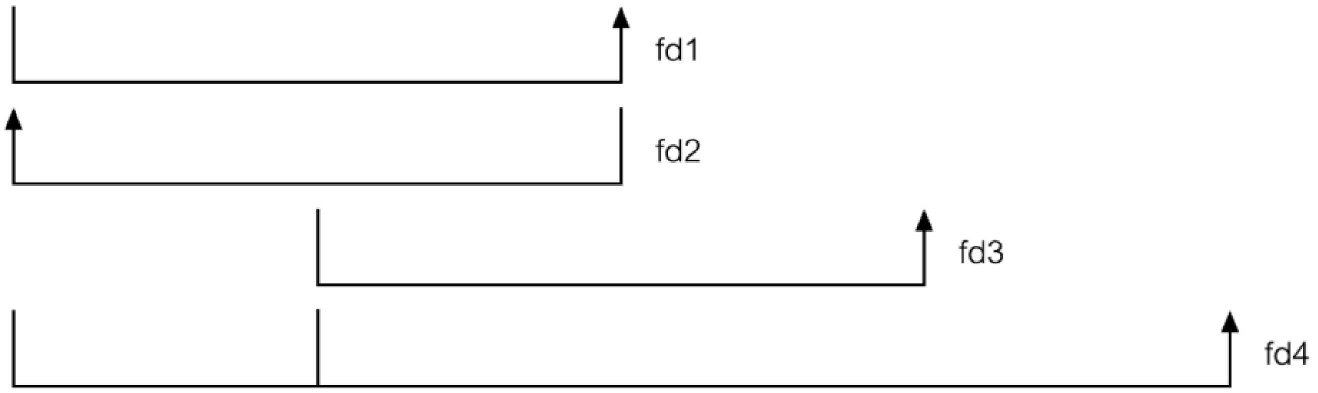
Example - Using sample data to identify functional dependencies.

- Consider the data for attributes denoted A, B, C, D, and E in the Sample relation (see Slide 27).
- Important to establish that sample data values shown in 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.

Sample Relation

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.

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

A C	(fd1)
C A	(fd2)
B D	(fd3)
A, B E	(fd4)

Identifying the Primary Key for a Relation using Functional Dependencies

- 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

- StaffBranch relation has five functional dependencies (see Slide 25).
- The determinants are staffNo, branchNo, bAddress, (branchNo, position), and (bAddress, position).
- To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation.

Example - Identify Primary Key for StaffBranch Relation

- 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.

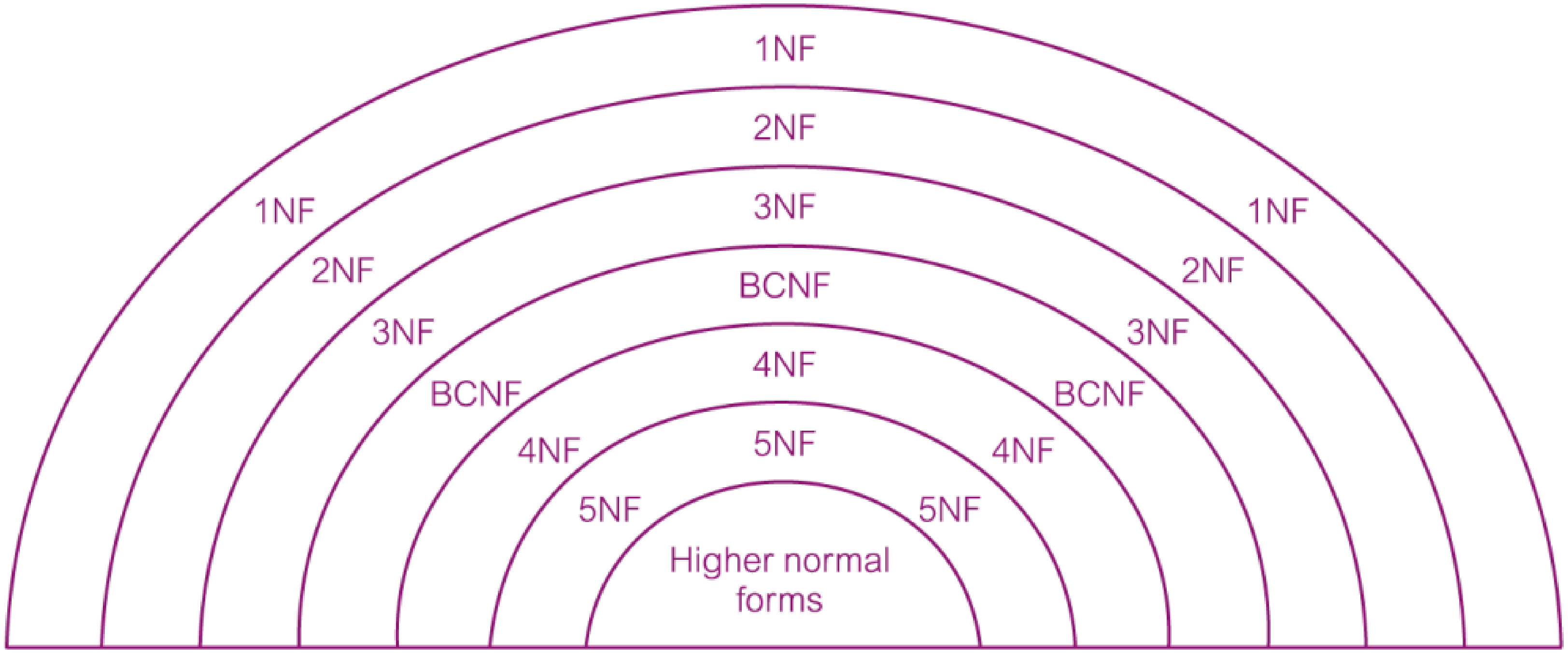
Example - Identifying Primary Key for Sample Relation

- Sample relation has four functional dependencies (see Slide 28).
- The determinants in the Sample relation are A, B, C, and (A, B). However, the only determinant that functionally determines all the other attributes of the relation is (A, B).
- (A, B) is identified as the primary key for this relation.

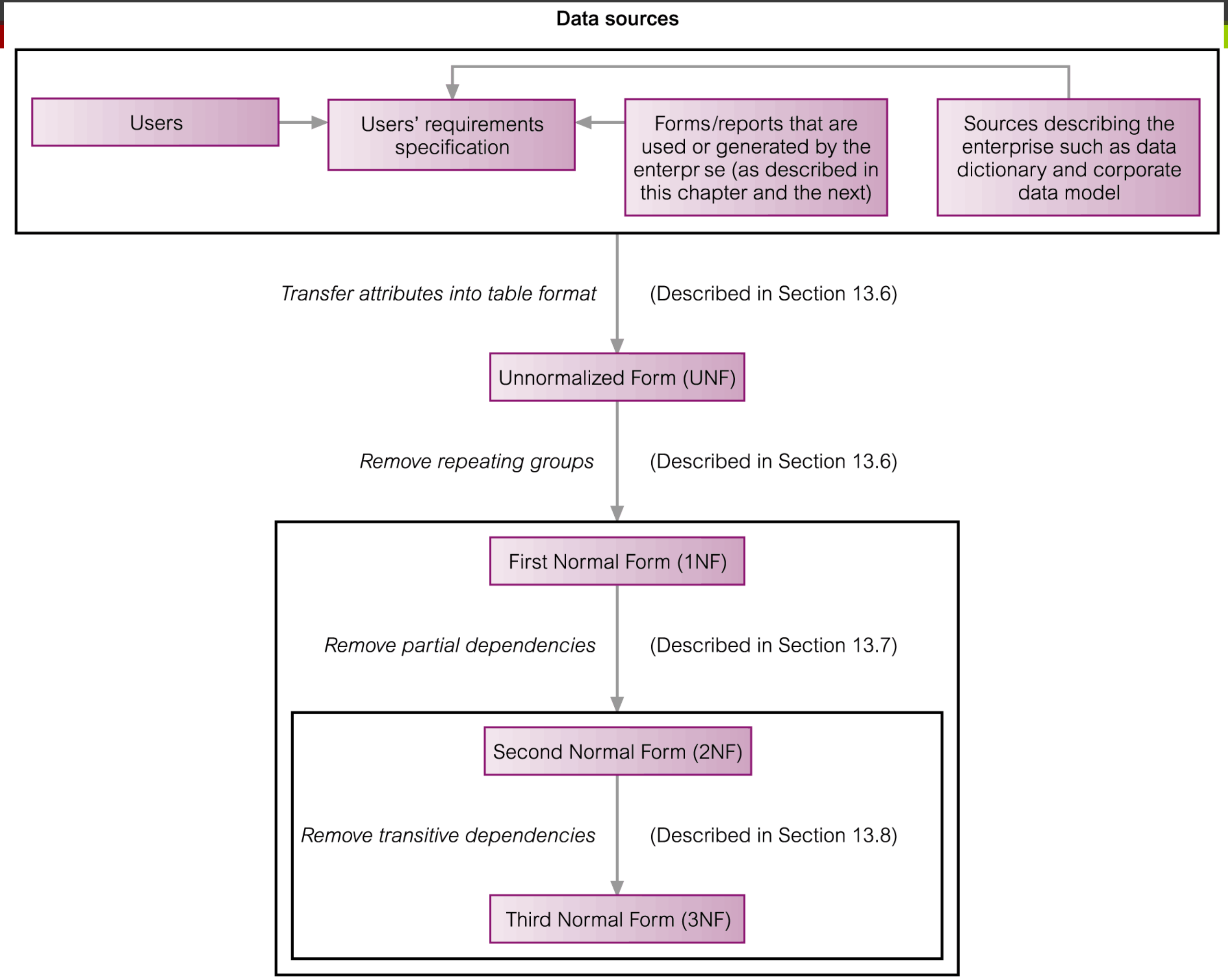
The Process of Normalization

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

The Process of Normalization



The Process of Normalization

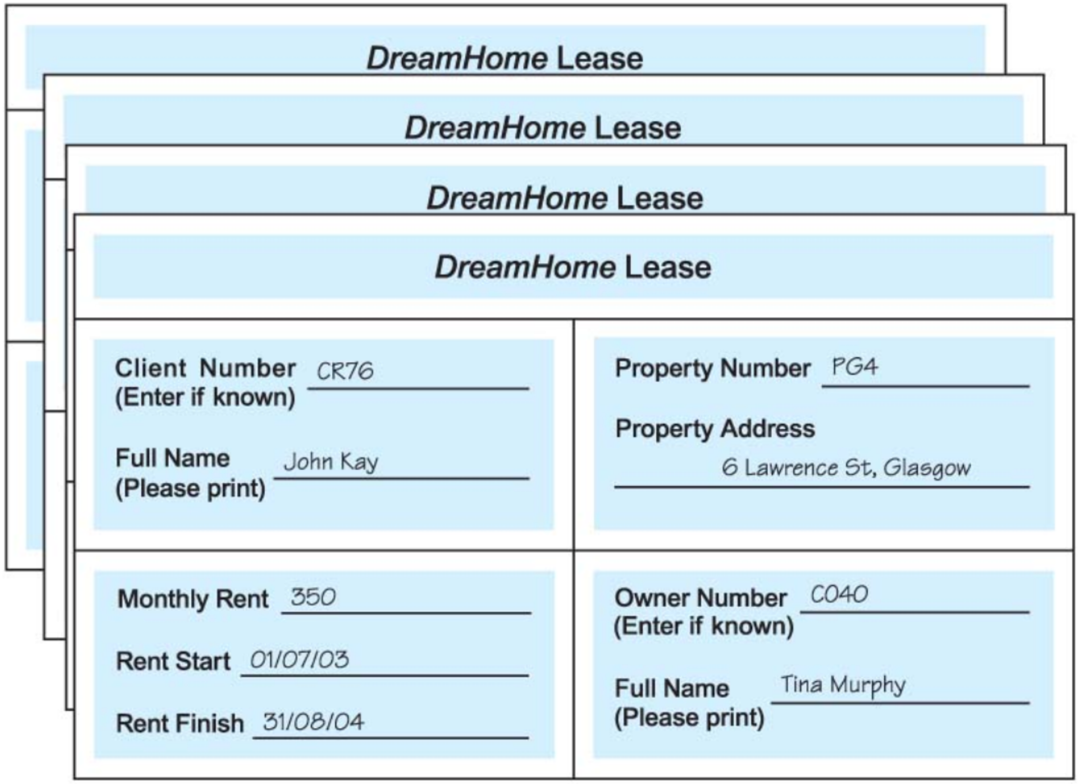

Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterpr se (as described in this chapter and the next)
Sources describing the enterprise such as data dictionary and corporate data model
Transfer attributes into table format
(Described in Section 13.6)
Unnormalized Form (UNF)
Remove repeating groups
(Described in Section 13.6)
First Normal Form (1NF)
Remove partial dependencies
(Described in Section 13.7)
Second Normal Form (2NF)
Remove transitive dependencies
(Described in Section 13.8)
Third Normal Form (3NF)

Unnormalized Form (UNF)

- A table that contains one or more repeating groups.

- To create an unnormalized table
  - Transform the data from the information source (e.g. form) into table format with columns and rows.

Example



DreamHome Lease

DreamHome Lease

DreamHome Lease

DreamHome Lease

Client Number (Enter if known) CR76

Full Name (Please print) John Kay

Property Number PG4

Property Address 6 Lawrence St, Glasgow

Monthly Rent 350

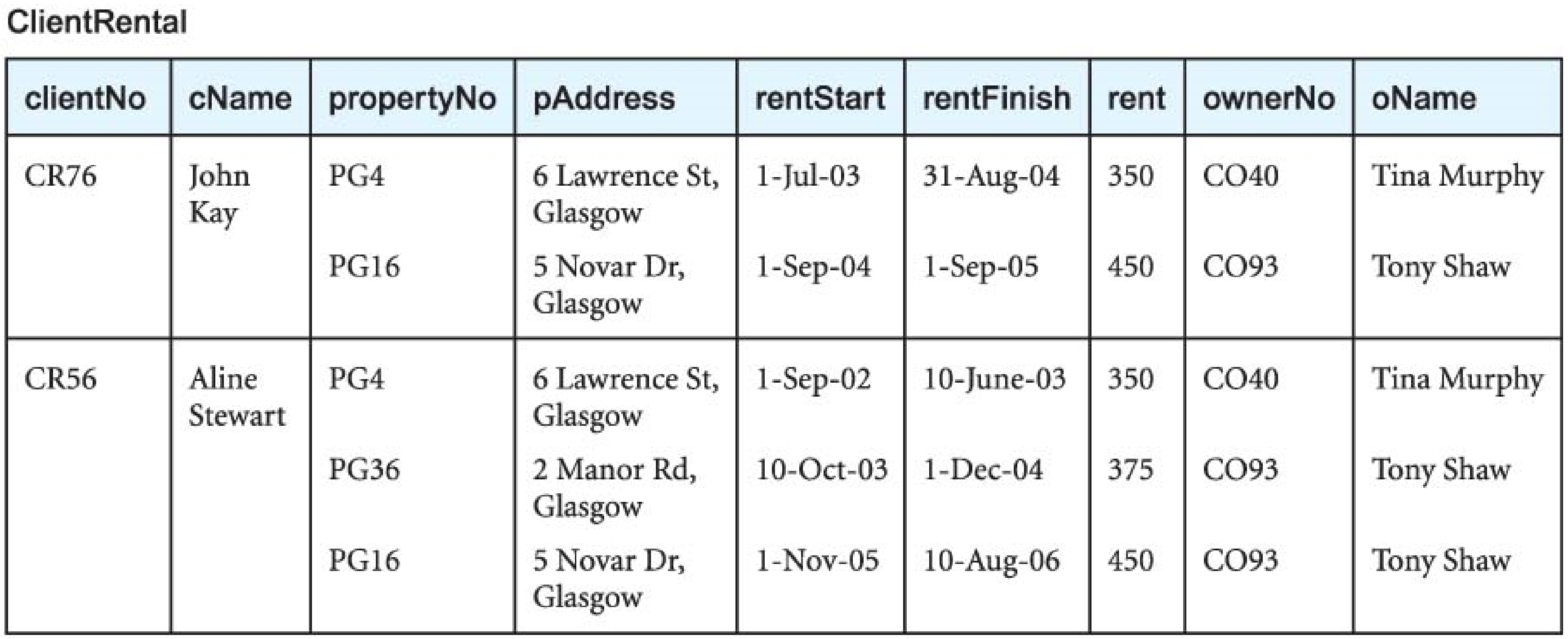
Rent Start 01/07/03

Rent Finish 31/08/04

Owner Number (Enter if known) CO40

Full Name (Please print) Tina Murphy

Example



ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-03	31-Aug-04	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-04	1-Sep-05	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-02	10-June-03	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-03	1-Dec-04	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-05	10-Aug-06	450	CO93	Tony Shaw

ClientRental unnormalized table.

First Normal Form (1NF)

- A relation in which the intersection of each row and column contains one and only one value.

UNF to 1NF

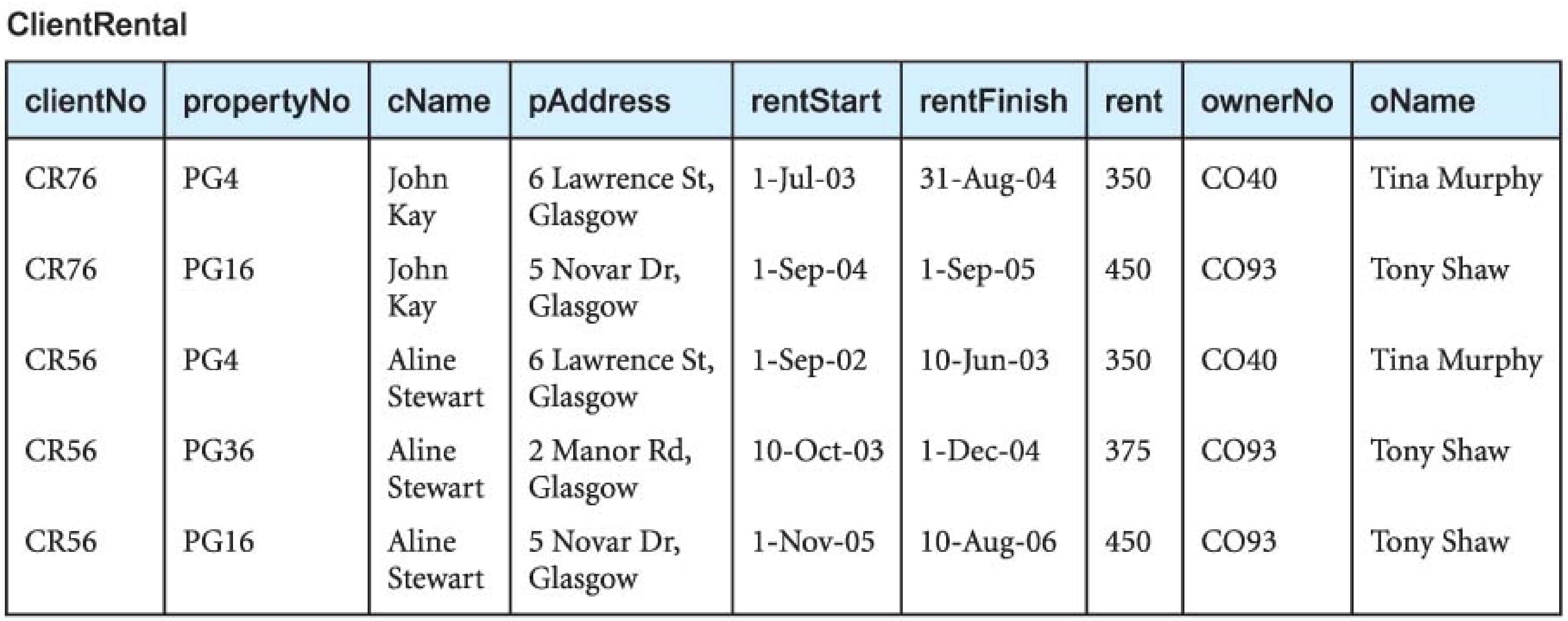
- Nominate an attribute or group of attributes to act as the key for the unnormalized table.

- Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).

UNF to 1NF

- Remove the repeating group by
  - Entering appropriate data into the empty columns of rows containing the repeating data (‘flattening’ the table).
  - Or by
  - Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

Example



ClientRental

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-03	31-Aug-04	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-04	1-Sep-05	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-02	10-Jun-03	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-03	1-Dec-04	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-05	10-Aug-06	450	CO93	Tony Shaw

First Normal Form ClientRental relation

Second Normal Form (2NF)

- Partial FDs:
  - A FD, A   B is a partial FD, if some attribute of A can be removed and the FD still holds
  - Formally, there is some proper subset of A,
  - $C \subset A$, such that $C \rightarrow B$

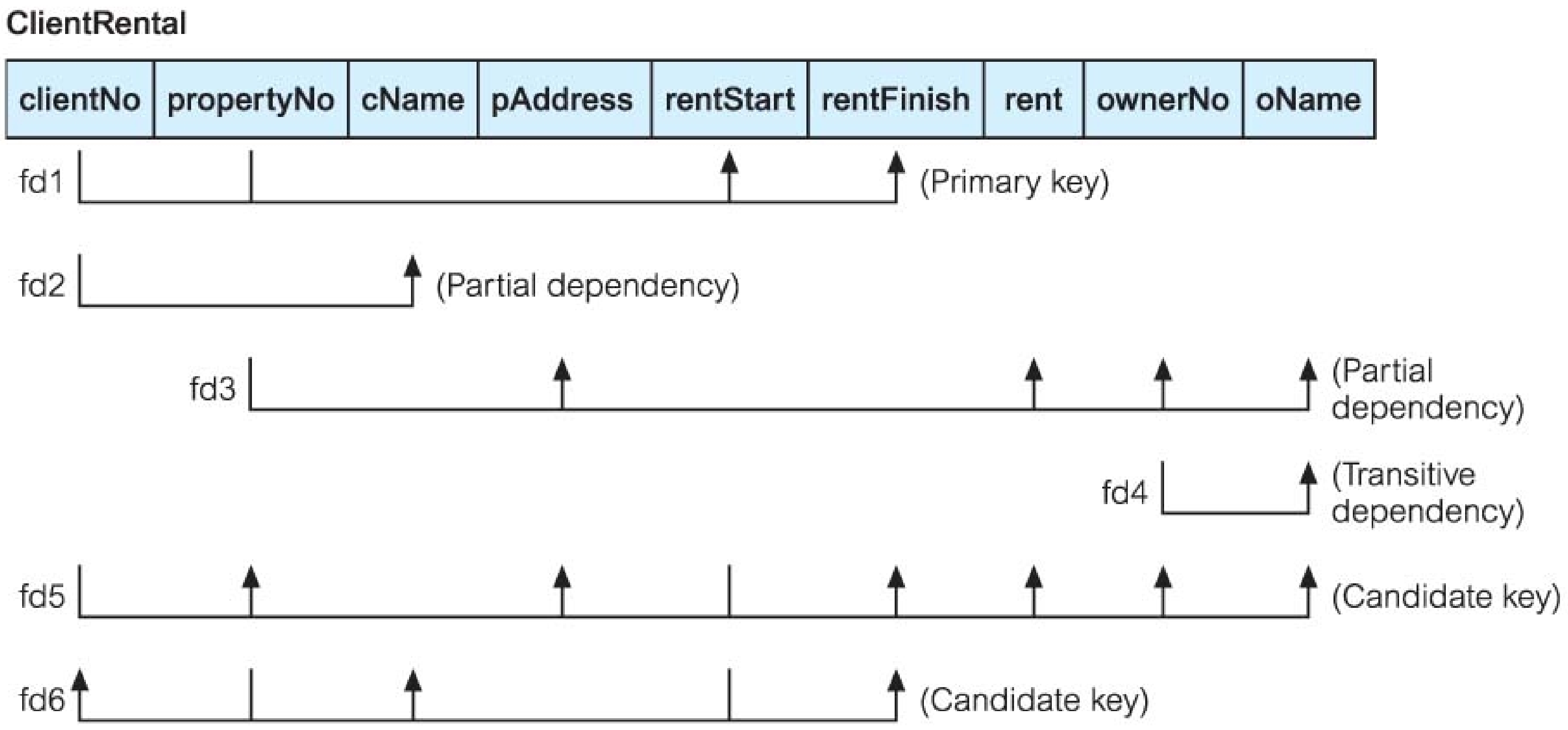
Second Normal Form (2NF)

- A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.
- or
- no non-key attribute is partially dependent on a candidate key

1NF to 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.

Example



Functional dependencies of the ClientRental relation.

Example

- the ClientRental relation has the following functional dependencies:
  - 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)

Example

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-03	31-Aug-04
CR76	PG16	1-Sep-04	1-Sep-05
CR56	PG4	1-Sep-02	10-Jun-03
CR56	PG36	10-Oct-03	1-Dec-04
CR56	PG16	1-Nov-05	10-Aug-06

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

Second Normal Form relations derived from the ClientRental relation.

Third Normal Form (3NF)

- Transitive Dependency is 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 through B. (Provided that A is not functionally dependent on B or C).

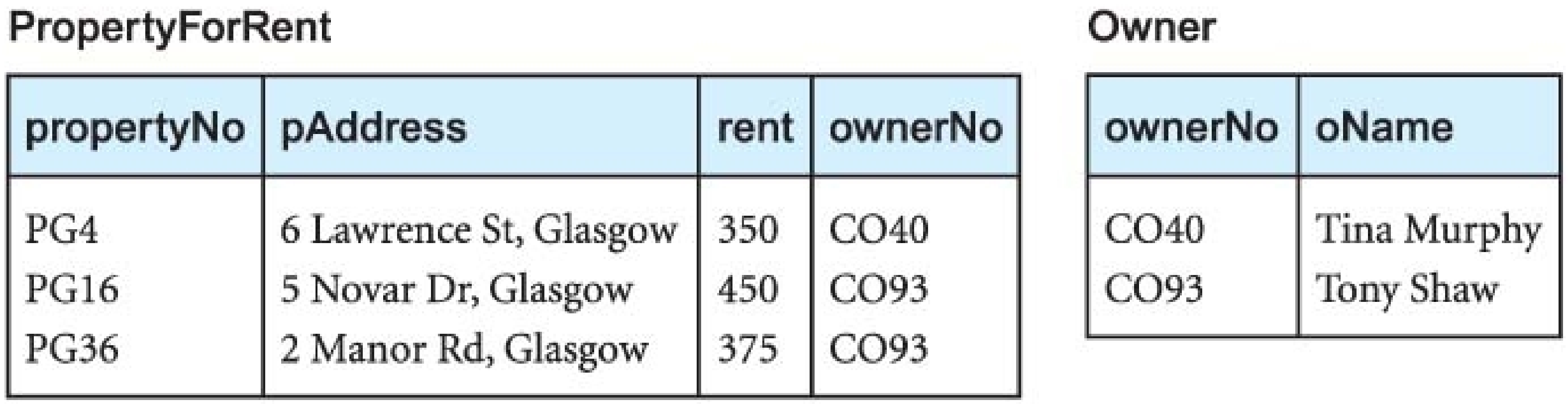
Third Normal Form (3NF)

- 2NF and in which no non-primary-key attribute is transitively dependent on the primary key or a candidate key

2NF to 3NF

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

Example



PropertyForRent

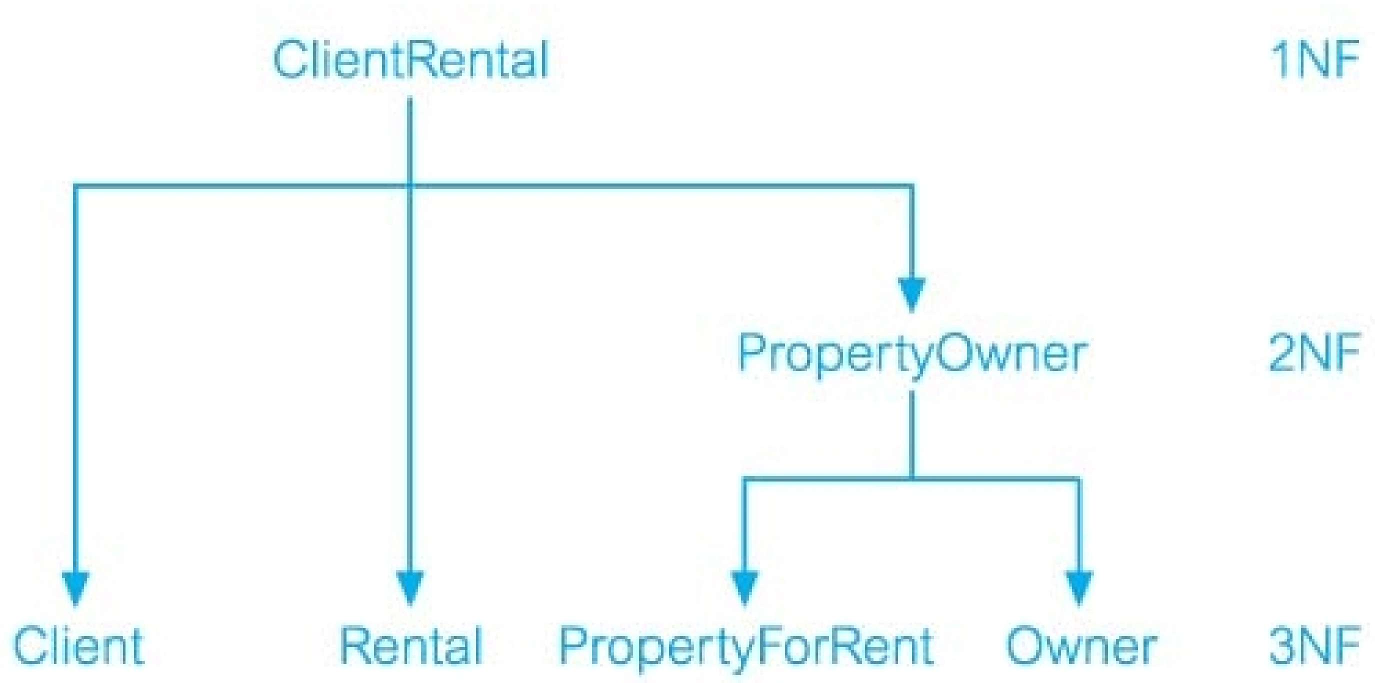
propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

Third Normal Form relations derived from the PropertyOwner relation.

Example



The decomposition of the ClientRental 1NF relation into 3NF relations.

Example

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-03	31-Aug-04
CR76	PG16	1-Sep-04	1-Sep-05
CR56	PG4	1-Sep-02	10-Jun-03
CR56	PG36	10-Oct-03	1-Dec-04
CR56	PG16	1-Nov-05	10-Aug-06

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

A summary of the 3NF relations derived from the ClientRental relation.

Boyce–Codd Normal Form (BCNF)

- Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with the general definition of 3NF.

- Boyce–Codd normal form (BCNF)
  - A relation is in BCNF if and only if every determinant is a candidate key.

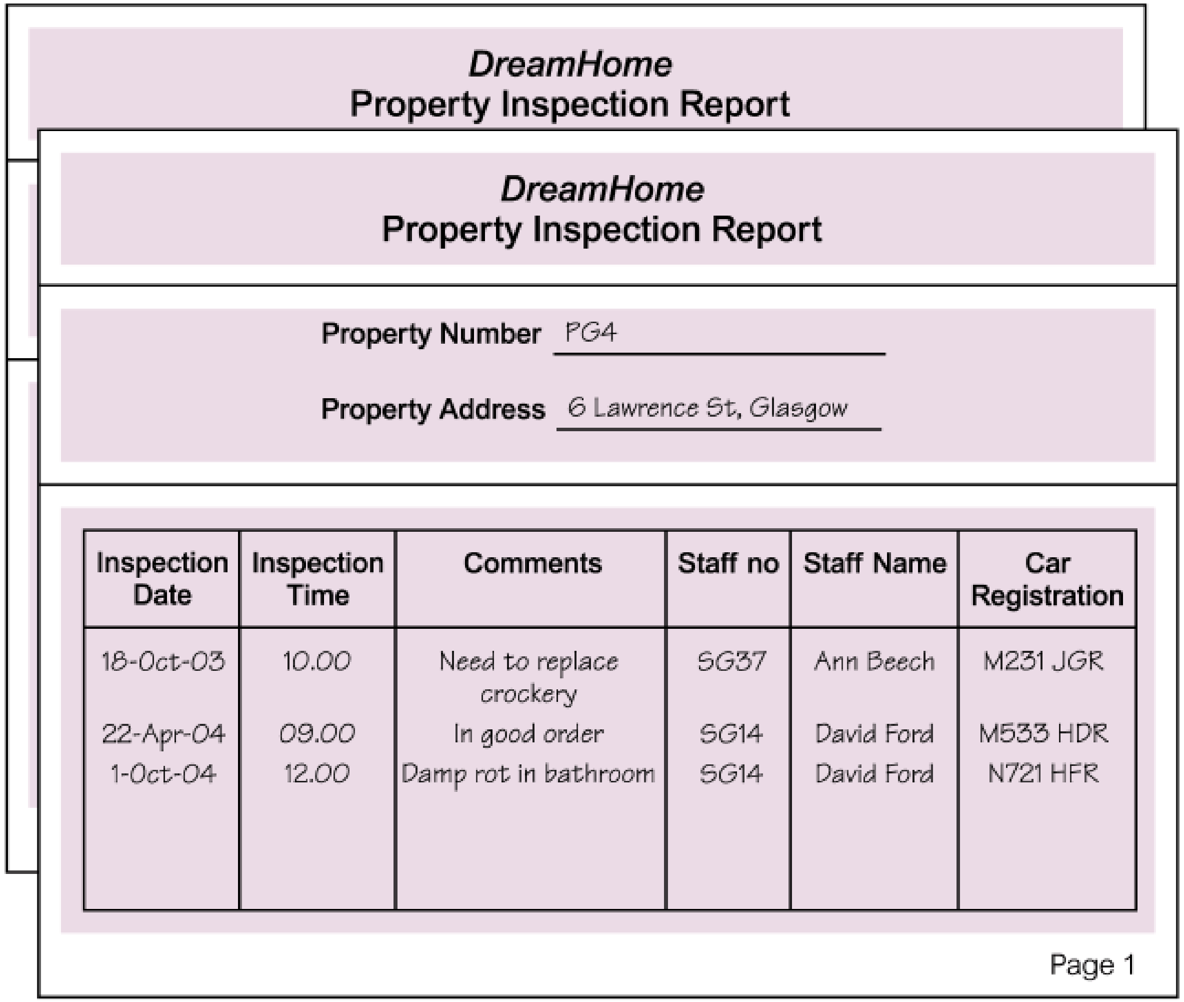
Boyce–Codd Normal Form (BCNF)

- Difference between 3NF and BCNF is that for a functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key.
- Whereas, BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
- Every relation in BCNF is also in 3NF. However, a relation in 3NF is not necessarily in BCNF.

Boyce–Codd Normal Form (BCNF)

- Violation of BCNF is quite rare.
- The potential to violate BCNF may occur in a relation that:
  - contains two (or more) composite candidate keys;
  - the candidate keys overlap, that is have at least one attribute in common.

Review of Normalization (UNF to BCNF)



DreamHome
Property Inspection Report

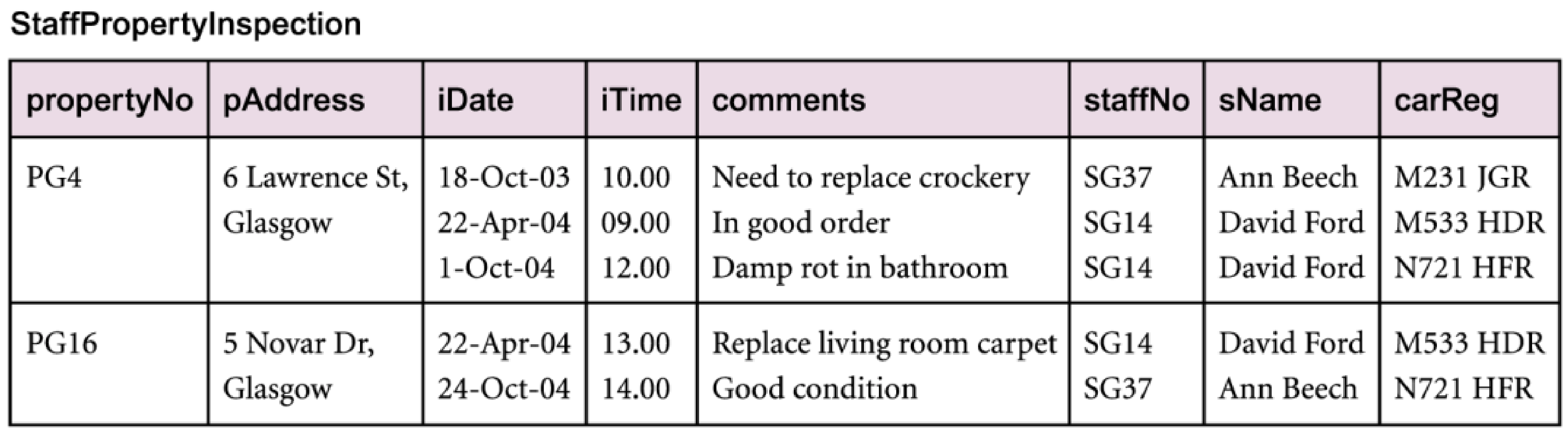
DreamHome
Property Inspection Report

Property Number	PG4
Property Address	6 Lawrence St, Glasgow

Inspection Date	Inspection Time	Comments	Staff no	Staff Name	Car Registration
18-Oct-03	10.00	Need to replace crockery	SG37	Ann Beech	M231 JGR
22-Apr-04	09.00	In good order	SG14	David Ford	M533 HDR
1-Oct-04	12.00	Damp rot in bathroom	SG14	David Ford	N721 HFR

Page 1

Review of Normalization (UNF to BCNF)



StaffPropertyInspection

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG4	6 Lawrence St, Glasgow	18-Oct-03 22-Apr-04 1-Oct-04	10.00 09.00 12.00	Need to replace crockery In good order Damp rot in bathroom	SG37 SG14 SG14	Ann Beech David Ford David Ford	M231 JGR M533 HDR N721 HFR
PG16	5 Novar Dr, Glasgow	22-Apr-04 24-Oct-04	13.00 14.00	Replace living room carpet Good condition	SG14 SG37	David Ford Ann Beech	M533 HDR N721 HFR

StaffPropertyInspection unnormalized table

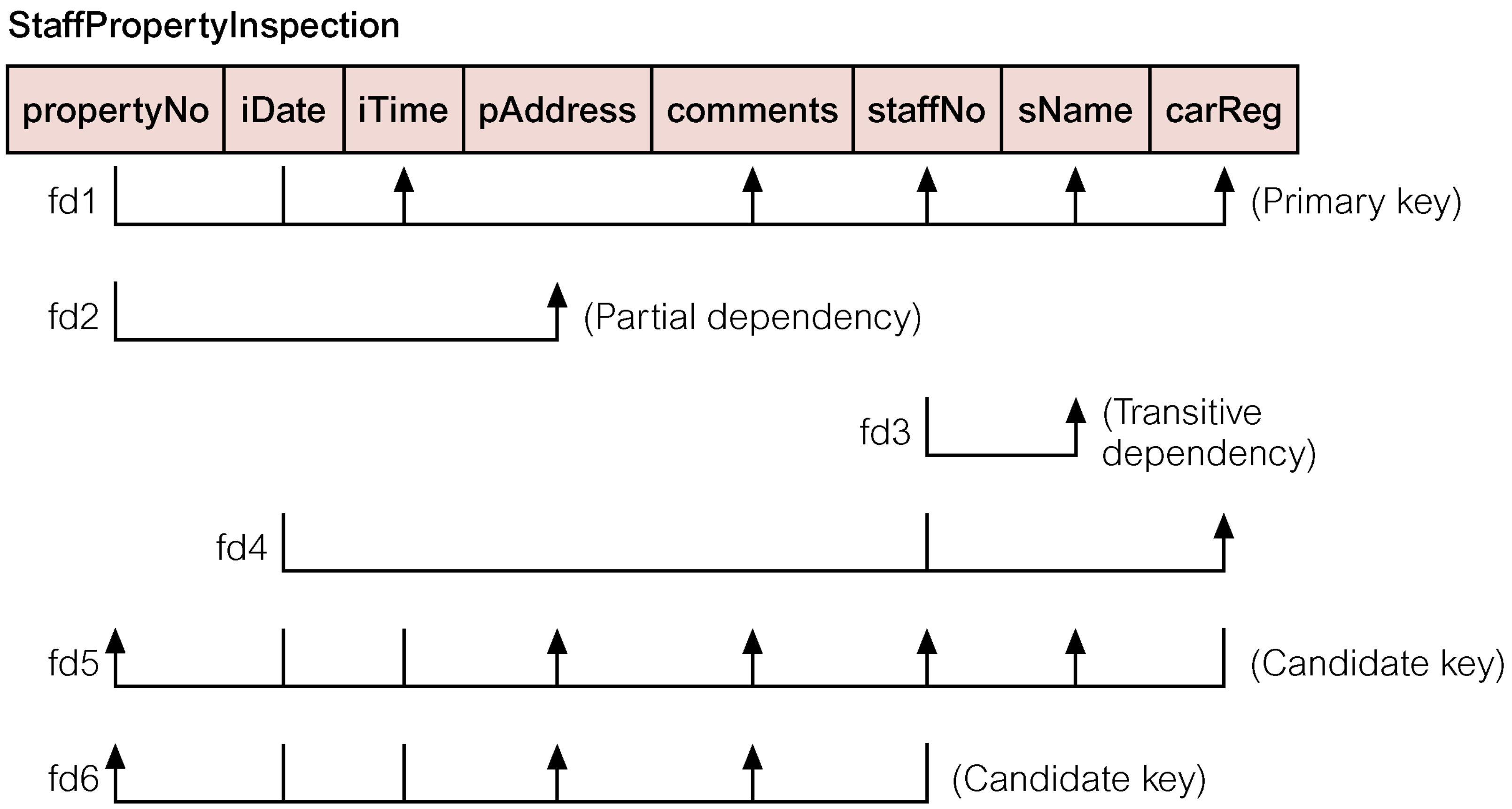
Review of Normalization (UNF to BCNF)

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg
PG4	18-Oct-03	10.00	6 Lawrence St, Glasgow	Need to replace crockery	SG37	Ann Beech	M231 JGR
PG4	22-Apr-04	09.00	6 Lawrence St, Glasgow	In good order	SG14	David Ford	M533 HDR
PG4	1-Oct-04	12.00	6 Lawrence St, Glasgow	Damp rot in bathroom	SG14	David Ford	N721 HFR
PG16	22-Apr-04	13.00	5 Novar Dr, Glasgow	Replace living room carpet	SG14	David Ford	M533 HDR
PG16	24-Oct-04	14.00	5 Novar Dr, Glasgow	Good condition	SG37	Ann Beech	N721 HFR

The First Normal Form (1NF) StaffPropertyInspection relation.

Review of Normalization (UNF to BCNF)



Review of Normalization (UNF to BCNF)

- Second Normal Form (2NF)
  - Property(propertyNo, pAddress)
  - PropertyInspection (propertyNo, iDate, iTime, comments, staffNo, sName, carReg)
- Third Normal Form (3NF)
  - Property (propertyNo, pAddress)
  - Staff (staffNo, sName)
  - PropertyInspect (propertyNo, iDate, iTime, comments, staffNo, carReg)

Review of Normalization (UNF to BCNF)

- Boyce–Codd Normal Form (BCNF)
  - a relation is in BCNF if every determinant of a relation is candidate key
  - the Property and Staff relations are already in BCNF as the determinant in each of these relations is also the candidate key.
  - The only 3NF relation that is not in BCNF is PropertyInspect because of the presence of the determinant (staffNo, iDate), which is not a candidate key (represented as fd4)

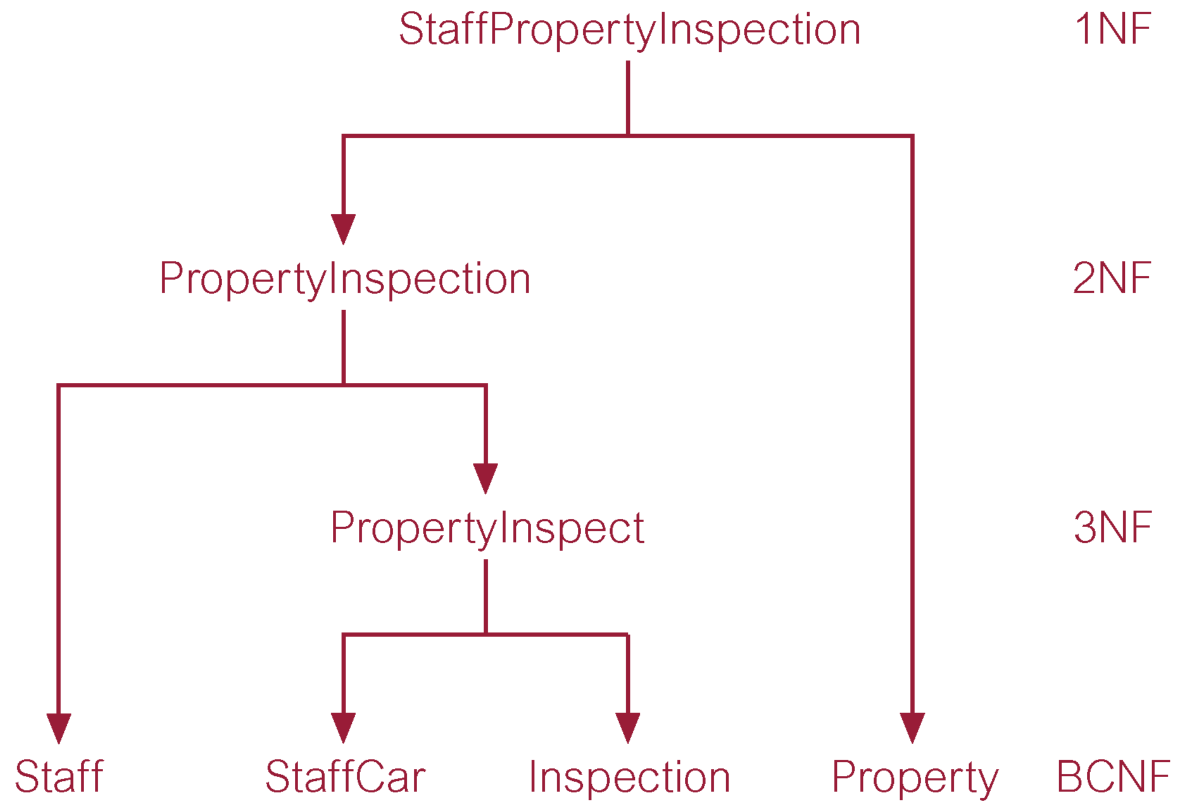
Review of Normalization (UNF to BCNF)

- To transform the PropertyInspect relation into BCNF, we must remove the dependency that violates BCNF by creating two new relations called StaffCar and Inspection with the form:
  - StaffCar(staffNo, iDate, carReg)
  - Inspection (propertyNo, iDate, iTime, comments, staffNo)

Review of Normalization (UNF to BCNF)

- The resulting BCNF relations have the following form:
  - Property (propertyNo, pAddress)
  - Staff (staffNo, sName)
  - Inspection (propertyNo, iDate, iTime, comments, staffNo)
  - StaffCar (staffNo, iDate, carReg)

Review of Normalization (UNF to BCNF)



Decomposition Properties

- Normalisation to 3NF is always lossless and dependency preserving
- Normalisation to BCNF is lossless, but may not preserve all dependencies

Normalisation and Design

- Normalisation is related to DB design
  - A database should normally be in 3NF at least
  - If your design leads to a non-3NF DB, then you might want to revise it
- When you find you have a non-3NF DB
  - Identify the FDs that are causing a problem
  - Think whether they will lead to any insert, update, or delete anomalies
  - Try to remove them

Normalisation Summary

- First normal form
  - All data values are atomic
- Second normal form
  - In 1NF plus no non-key attribute is partially dependent on a candidate key
- Third normal form
  - In 2NF plus no non-key attribute depends transitively on a candidate key
- Boyce-Codd Normal Form
  - In 2NF plus no attribute depends on a non super key

Denormalisation

- Normalisation
  - Removes data redundancy
  - Solves INSERT, UPDATE, and DELETE anomalies
  - This makes it easier to maintain the information in the database in a consistent state

Denormalisation

- However
  - It leads to more tables in the database
  - Often these need to be joined back together, which is expensive to do
  - So sometimes (not often) it is worth ‘denormalising’

Denormalisation

- You *might* want to denormalise if
  - Database speeds are unacceptable (not just a bit slow)
  - There are going to be very few INSERTs, UPDATEs, or DELETEs
  - There are going to be lots of SELECTs that involve the joining of tables