1. Normalize the following Client Rental data. Assume a client (e.g. John Kay) rents a given property only once and cannot rent more than one property at any one time. Sample data is taken from two leases for two different clients called John Kay and Alice Stewart. Also assume that we do not want to maintain historical information on client rentals. A client appears in the records only when he or she is currently renting a property.

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	Rent	ownerNo	oName
CR76	John Kay	PG4	6 North St, Victoria	1-Jul-07	30-Jun-08	500	C040	Tina Murphy
		PG16	5 West Ave, Victoria	1-Jul-08	31-Aug-09	600	C093	Tony Shaw
CR56	Alice Stewart	PG4	6 North St, Victoria	1-Jan-06	31-Dec-06	500	C040	Tina Murphy
		PC36	2 Manor Rd, Colwood	1-Jan-07	30-Jun-07	550	C088	Wendy Faber
		PG16	5 West Ave, Victoria	1-Jul-07	31-Dec-07	600	C093	Tony Shaw

What is the problem with this structure?

1. There are repeating attributes (propertyNo, pAddress, etc). This is bad because the structure has redundancy issues with repeating attributes. A repeating attribute means there is more than one value at the intersection of any row and column in the table.

2. There is redundancy. The pAddress attribute is duplicated anytime a property is rented more than once. Also, the owner's name is duplicated anytime his or her property is rented.

3. There is an update anomaly if the address for a property changes and not all records are corrected.

4. There is a deletion anomaly if the only record containing information about an owner is deleted (e.g. property PC36) – you would lose information about that owner number and the owner's property.

This is an unnormalized structure.

The process of normalization begins by identifying and removing repeating groups in the table. The result will be a table in First Normal Form (1NF).

First Normal Form

There are two approaches to removing repeating groups from unnormalized tables:

a. Remove the repeating groups by entering appropriate values in the empty columns containing the repeating data. This is "fill-in-the-blanks" by duplicating the non-repeating data where required. This approach is also called "flattening" the table. The resulting table (now referred to as a relation) contains only atomic (single) values at the intersection of each row and column. Redundancy is introduced in this approach but subsequent normalization will remove it.

b. Or, remove the repeating group by placing the repeating data along with a copy of the original key attribute(s), in a separate relation. A primary key is identified for the new relation.

The primary key is identified in the ClientRental table as clientNo.

The repeating group in the unnormalized table is identified as details pertaining to the rental.

Repeating group: propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName

With the first approach, the repeating group is removed by entering the appropriate client data into each row.

ClientRental

Client No*	Property No*	cName	pAddress	rentStart	rentFinish	Rent	ownerNo	oName
CR76	PG4	John Kay	6 North St, Victoria	1-Jul-07	30-Jun-08	500	C040	Tina Murphy
CR76	PG16	John Kay	5 West Ave, Victoria	1-Jul-08	31-Aug-09	600	C093	Tony Shaw
CR56	PG4	Alice Stewart	6 North St, Victoria	1-Jan-06	31-Dec-06	500	C040	Tina Murphy
CR56	PC36	Alice Stewart	2 Manor Rd, Colwood	1-Jan-07	30-Jun-07	550	C088	Wendy Faber
CR56	PG16	Alice Stewart	5 West Ave, Victoria	1-Jul-07	31-Dec-07	600	C093	Tony Shaw

Primary key

The ClientRental relation in 1NF (after removing repeating group)

We identify candidate keys in the ClientRental relation as being these composite keys – (clientNo, propertyNo) and (clientNo, rentStart) and (propertyNo, rentStart)

The primary key is selected to be (clientNo, propertyNo).

The rentFinish attribute not appropriate as part of primary key because it may contain nulls (rent end date not established).

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

With the second approach, the repeating group is removed by placing the repeating data along with a copy of the original key attribute (clientNo) in a separate relation.

Client (clientNo, cName)

PropertyRentalOwner (clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

Both relations Client and PropertyRentalOwner are in 1NF as there is a single value at the intersection of each row and column.

Primary key indicated by the *.

Client

clientNo *	cName
CR76	John Kay
CR56	Alice Stewart

PropertyRentalOwner

client No *	property No *	pAddress	rentStart	rentFinish	Rent	owner No	oName
CR76	PG4	6 North St, Victoria	1-Jul-07	30-Jun-08	500	C040	Tina Murphy
CR76	PG16	5 West Ave, Victoria	1-Jul-08	31-Aug-09	600	C093	Tony Shaw
CR56	PG4	6 North St, Victoria	1-Jan-06	31-Dec-06	500	C040	Tina Murphy
CR56	PC36	2 Manor Rd, Colwood	1-Jan-07	30-Jun-07	550	C088	Wendy Faber
CR56	PG16	5 West Ave, Victoria	1-Jul-07	31-Dec-07	600	C093	Tony Shaw

Second Normal Form

There are still anomalies with the 1NF relations. Any changes made to the property address require all records for that property to be changed or the data will be inconsistent. If the record containing the rental of property PC36 is deleted, then we lose the information that ownerNo C088 is Wendy Faber. We can't add a new owner until that owner has rented a property.

Second normal form is based on the concept of full functional dependency and is of concern if the relation has a composite primary key.

A fully functional dependency is described as follows:

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 subset of A.

A functional dependency is expressed like this:

A → B

B is functionally dependent on A (each value of A is associated with exactly one value of B) if B is determined directly by A. A *determines* B.

A is called the *determinant*. Any entity on the left side of the functional dependency is called a *determinant*.

This does not automatically imply a one-to-one relationship between A and B, though a one-to-one relationship may exist.

Examples of functional dependencies:

studentNumber → studentName, studentAddress

The studentName and studentAddress values are determined by the value of studentNumber

studentNumber, courseNumber, quarter → grade, date

The combination of studentNumber AND courseNumber AND quarter together determines a grade value and its date

lockerID → studentNumber

snack → calorieCount

postalCode → address, city, province

passportNumber → countryOfOrigin, name, birthDate, passportExpiry

boardingPass → seatNumber, travelClass

hostname → IPAddress

NHLplayer, season → NHLteam

courseNumber, quarter → instructor, classroom, textbook

If the primary key is a single attribute, then the relation is automatically in 2NF.

Examples of fully functional dependencies (no partial dependencies):

studentNumber, courseNumber, quarter → grade, date

The values of grade and date together are not determined by any subset combination of studentNumber, courseNumber or quarter. This is making the assumption that many students are taking many courses in different quarters. Also, it is assumed that a student is enrolled at most once into a course each quarter.

author, title → publisher

This will hold true assuming an author may use different publishers for his or her books. If authors must use a single publisher exclusively for all his or her books, then author → publisher and this is not a fully functional dependency.

NHLplayer, season → NHLteam

This is true if any NHLplayer can move to another team in other seasons.

employee_id, job_id → hoursWorked

These are not fully functional dependencies (they are partial dependencies):

studentNumber, courseNumber → instructor

The instructor values is dependent on courseNumber (courseNumber → instructor)

companyName, date → stockSymbol, closingPrice, headquarters

The value for headquarters is dependent on companyName. companyName → headquarters

employee_id, job_id → employeeName

The value of employeeName is dependent on employee_id

Analyze the model for 2NF – it must be in 1NF and every non-primary key attribute is fully functional dependent on the primary key (no partial dependencies).

Analyzing the model's ClientRental relation functional dependencies:

clientNo , propertyNo → rentStart, rentFinish	primary key
clientNo → cName	partial dependency
propertyNo → pAddress, rent, ownerNo, oName	partial dependency
ownerNo → oName	transitive dependency (worry about this in 3NF)
clientNo, rentStart → propertyNo, pAddress, rentFinish, rent, ownerNo, oName	candidate key
propertyNo, rentStart → clientNo, cName, rentFinish	candidate key

Since there are partial dependencies in the ClientRental relation, it is not in 2NF.

To transform the ClientRental relation into 2NF, we create new relations so that the non-primary-key attributes are removed along with a copy of the part of the primary key on which are fully functionally dependent.

The results are two new relations Client and Rental.

Client (<u>clientNo</u>, cName)

PropertyOwner (<u>propertyNo</u>, pAddress, rent, ownerNo, oName)

Rental (<u>clientNo, propertyNo</u>, rentStart, rentFinish)

- note update anomaly here – if the name of the owner changes, you have to update two records in PropertyOwner – this is caused by a transitive dependency

Primary key indicated by the *.

Client

clientNo *	cName
CR76	John Kay
CR56	Alice Stewart

Rental

clientNo *	propertyNo *	rentStart	rentFinish
CR76	PG4	1-Jul-07	30-Jun-08
CR76	PG16	1-Jul-08	31-Aug-09
CR56	PG4	1-Jan-06	31-Dec-06
CR56	PC36	1-Jan-07	30-Jun-07
CR56	PG16	1-Jul-07	31-Dec-07

PropertyOwner

propertyNo *	pAddress	Rent	ownerNo	oName
PG4	6 North St, Victoria	500	C040	Tina Murphy
PG16	5 West Ave, Victoria	600	C093	Tony Shaw
PC36	2 Manor Rd, Colwood	550	C088	Wendy Faber

Third Normal Form

Analyze the model for 3NF – it must be in 2NF and no non-primary key attribute is transitively dependent on the primary key. (Remove any transitive dependencies)

A transitive dependency is a type of functional dependency. It's 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).

Examples of transitive dependencies:

instructor → department
department → buildingLocation

instructor → buildingLocation exists as a transitive dependency via the departmentLocation attribute

employee_id → department_id
department_id → department_manager

employee_id → department_manager exists as a transitive dependency via the department_id

Analyzing the model's relations functional dependencies:

Client
clientNo → cName Primary key

Rental

clientNo, propertyNo → rentStart, rentFinish	Primary key
clientNo, rentStart → propertyNo, rentFinish	Candidate key
propertyNo, rentStart → clientNo, rentFinish	Candidate key

PropertyOwner

propertyNo → pAddress, rent, ownerNo, oName	Primary key
ownerNo → oName	Transitive dependency propertyNo→ownerNo ownerNo → oName creates propertyNo → oName

Since there is a transitive dependency in the PropertyOwner relation, it is not in 3NF.

To transform the PropertyOwner relation into 3NF, we create new relations so that the transitive dependent attributes are removed along with a copy of the determinant (in this case ownerNo is a determinant).

The results are two new relations PropertyForRent and Owner.

PropertyForRent (<u>propertyNo</u>, pAddress, rent, ownerNo)

Owner (<u>ownerNo</u>, oName)

Client (<u>clientNo</u>, cName)

Rental (<u>clientNo, propertyNo</u>, rentStart, rentFinish)

All four relations are now in 3NF.

Client

clientNo *	cName
CR76	John Kay
CR56	Alice Stewart

Rental

clientNo *	propertyNo *	rentStart	rentFinish
CR76	PG4	1-Jul-07	30-Jun-08
CR76	PG16	1-Jul-08	31-Aug-09
CR56	PG4	1-Jan-06	31-Dec-06
CR56	PC36	1-Jan-07	30-Jun-07
CR56	PG16	1-Jul-07	31-Dec-07

PropertyForRent

propertyNo *	pAddress	Rent	ownerNo
PG4	6 North St, Victoria	500	C040
PG16	5 West Ave, Victoria	600	C093
PC36	2 Manor Rd, Colwood	550	C088

Owner

ownerNo *	oName
C040	Tina Murphy
C088	Wendy Faber
C093	Tony Shaw

2. Examine the Patient Medication Form for the Wellmeadows Hospital case
    a. Identify the functional dependencies represented by the data shown in the form.
    b. Normalize the data to first, second, and third normal forms.
    c. Identify the primary and foreign keys in your relations.

Wellmeadows Hospital Patient Medication Form							
Patient Number: P10034							
Full Name: Robert Smith			Ward number: Ward 11				
Bed Number: 84			Ward name: Orthopaedic				
Drug Number	Name	Description	Dosage	Method of Admin.	Units per Day	Start Date	End Date
10223	Morphine	Pain killer	10 mg/ml	Oral	50	24/04/06	25/04/06
10334	Tetracycline	Antibiotic	0.5mg/ml	IV	10	24/04/06	26/04/06
10223	Morphine	Pain Killer	10 mg/ml	Oral	10	25/04/06	26/04/06

(bed numbers are not unique in the hospital – each ward has sets of bed numbers)

patientNo → patientName
wardNo → wardName
patientNo, drugNo, startDate → unitsPerDay, endDate
drugNo → drugName, drugDesc, drugDosage, method

1NF

- flatten the table

Patient (<u>patientNo</u>, <u>drugNo</u>, <u>startDate</u>, patientName, wardNo, wardName,
bedNo, drugName, drugDesc, drugDosage, method, unitsPerDay, endDate)

2NF

- isolate attributes dependent on non-key attributes (drugNo)
- remove partial dependencies (patientNo)

PatientStay (patientNo, drugNo, startDate, wardNo, wardName,
bedNo, unitsPerDay, endDate)

Drug (drugNo, drugName, drugDesc, drugDosage, method)

Patient (patientNo, patientName)

3NF

- remove any transitive dependencies (patientNo → wardNo, wardNo → wardName and so patientNo → wardName)

PatientStay (patientNo, drugNo, startDate, wardNo (FK), bedNo, unitsPerDay, endDate)

Drug (drugNo, drugName, drugDesc, drugDosage, method)

Patient (patientNo, patientName)

Ward (wardNo, wardName)

(FK) indicates a foreign key

3. Create normalized (3NF) tables for the following example database: The president of a book wholesaler has told you that she wants information about publishers, authors and books. Assume an author can publish multiple books but only one edition of that book. An author may publish only under one publisher.

author → book
author → publisher (a publisher is determined by author)
book → publisher

1NF

- flatten the table

books (author, book, publisher)

2NF

- isolate attributes dependent on non-key attributes
- remove partial dependencies (author → publisher)

authors (author, book)

publishers (author, publisher)

3NF

- remove transitive dependencies within relations

authors (author, book)
publishers (author, publisher)

Database Systems 3rd Edition by Connolly, Begg; Addison Wesley
Database Modeling and Design by Teorey, Morgan Kaufmann