DATABASE DESIGN: NORMALIZATION NOTE & EXERCISES (Up to 3NF)

- Tables that contain redundant data can suffer from update anomalies, which can introduce inconsistencies into a database.
- The rules associated with the most commonly used normal forms, namely first (1NF), second (2NF), and third (3NF).
- The identification of various types of update anomalies such as insertion, deletion, and modification anomalies can be found when tables that break the rules of 1NF, 2NF, and 3NF and they are likely to contain redundant data and suffer from update anomalies.
- Normalization is a technique for producing a set of tables with desirable properties that support the requirements of a user or company.
- Major aim of relational database design is to group columns into tables to minimize data redundancy and reduce file storage space required by base tables.
- Take a look at the following example:

StdSSN	StdCity	StdClass	OfferNo	OffTerm	OffYear	EnrGrade	CourseNo	CrsDesc
S1	SEATTLE	JUN	O1	FALL	2006	3.5	C1	DB
S1	SEATTLE	JUN	O2	FALL	2006	3.3	C2	VB
S2	BOTHELL	JUN	O3	SPRING	2007	3.1	C3	OO
S2	BOTHELL	JUN	O2	FALL	2006	3.4	C2	VB

- The **insertion anomaly**: Occurs when extra data beyond the desired data must be added to the database. For example, to insert a course (`CourseNo`), it is necessary to know a student (`StdSSN`) and offering (`OfferNo`) because the combination of `StdSSN` and `OfferNo` is the primary key. Remember that a row cannot exist with `NULL` values for part of its primary key.
- The **update anomaly**: Occurs when it is necessary to change multiple rows to modify **ONLY** a single fact. For example, if we change the `StdClass` of student `S1` (`JUN`), two rows, row 1 and 2 must be changed. If `S1` was enrolled in 10 classes, 10 rows must be changed.
- The **deletion anomaly**: Occurs whenever deleting a row inadvertently causes other data to be deleted. For example, if we delete the enrollment (`EnrGrade`) of `S2` in `O3` (third row), we lose the information about offering `O3` and course `C3` because these values are unique to the table (cell). Furthermore `O3` is a primary key.

RECAP

Problems associated with data redundancy are illustrated by comparing the `Staff` and `Branch` tables with the `StaffBranch` table. Tables that have redundant data may have problems called update anomalies, which are classified as insertion, deletion, or modification anomalies. See the following Figure for an example of a table with redundant data called `StaffBranch`. There are two main types of insertion anomalies, which we illustrate using this table.

Insertion anomalies

1. To insert the details of a new member of staff (staffNo, name, position and salary) located at a given branch into the StaffBranch table, we must also enter the correct details for that branch (branchNo, branchAddress and telNo). For example, to insert the details of a new member of staff at branch B002, we must enter the correct details of branch B002 so that the branch details are consistent with values for branch B002 in other records of the StaffBranch table. The data shown in the StaffBranch table is also shown in the Staff and Branch tables. These tables do have redundant data and do not suffer from this potential inconsistency, because for each staff member we only enter the appropriate branch number into the Staff table. In addition, the details of branch B002 are recorded only once in the database as a single record in the Branch table.

2. To insert details of a new branch that currently has no members of staff into the StaffBranch table, it's necessary to enter NULLs into the staff-related columns, such as staffNo. However, as staffNo is the primary key for the StaffBranch table, attempting to enter nulls for staffNo violates entity integrity, and is not allowed. The design of the tables shown in Staff and Branch avoids this problem because new branch details are entered into the Branch table separately from the staff details. The details of staff ultimately located at a new branch can be entered into the Staff table at a later date.

Deletion anomalies

If we delete a record from the StaffBranch table that represents the last member of staff located at a branch, the details about that branch are also lost from the database. For example, if we delete the record for staff Art Peters (S0415) from the StaffBranch table, the details relating to branch B003 are lost from the database. The design of the tables that separate the Staff and Branch table avoids this problem because branch records are stored separately from staff records and only the column branchNo relates the two tables. If we delete the record for staff Art Peters (S0415) from the Staff table, the details on branch B003 in the Branch table remain unaffected.

Modification anomalies

If we want to change the value of one of the columns of a particular branch in the StaffBranch table, for example the telephone number for branch B001, we must update the records of all staff located at that branch (row 1 and 2). If this modification is not carried out on all the appropriate records of the StaffBranch table, the database will become inconsistent. In this example, branch B001 would have different telephone numbers in different staff records.
The above examples illustrate that the Staff and Branch tables have more desirable properties than the StaffBranch table.

StaffBranch Table

staffNo	name	position	salary	branchNo	branchAddress	telNo
S1500	Tom Daniels	Manager	46000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0003	Sally Adams	Assistant	30000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0010	Mary Martinez	Manager	50000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S3250	Robert Chin	Supervisor	32000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S2250	Sally Stern	Manager	48000	B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131
S0415	Art Peters	Manager	41000	B003	14 – 8th Avenue, New York, NY 10012	212-371-3000

Staff and Branch Tables

Staff

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Manager	41000	B003

Branch

branchNo	branchAddress	telNo
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
B002	City Center Plaza, Seattle, WA 98122	206-555-6756
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131

- `StaffBranch` table has redundant data; **the details of a branch** are repeated for every member of staff for example row 1 and 2, row 3 and 4 on `branchNo`, `branchAddress` and `telNo`.
- In contrast, the branch information appears only once for each branch in the `Branch` table and only the branch number (`branchNo`) is repeated in the `Staff` table, to represent where each member of staff is located.
- Let take a look at another example.

AIRCRAFT_1 Table

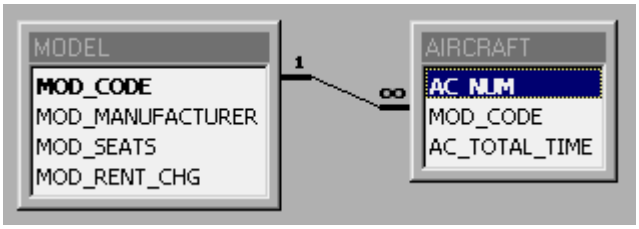
AC_NUM	AC_MODEL	AC_RENT_CHG	AC_SEATS	AC_TOTAL_TIME
1178R	Cessna C-172 Shyhawk	$58.50	4	4512.6
205IY	Cessna C-172 Shyhawk	$58.50	4	5325.5
2087V	Cessna C-152 Commuter	$51.75	2	4889.8
226BR	Cessna C-172 Shyhawk	$58.50	4	4299.6
2867W	Piper PA28-181 Archer II	$64.00	4	3267.4
3213R	Piper PA28-181 Archer II	$64.00	4	2517.9
4112E	Piper PA28-181 Archer II	$64.00	4	5211.3
45ZU	Cessna C-152 Commuter	$51.75	2	7003.1
5725Y	Cessna C-172 Shyhawk	$58.50	4	3968.2

- If we use the `AIRCRAFT_1` table as shown in the above Figure, a change in hourly rental rates (`AC_RENT_CHG`) for the `Cessna 172 Skyhawk` must be made four times; if we forget to change just one of those rates, we have a data integrity problem.
- How much better it would be to have critical data in only one place. Then, if a change must be made, it need be made only once.
- In contrast, table structures are good when they preclude the possibility of producing uncontrolled data redundancies.
- We can produce such a happy circumstance by splitting the `AIRCRAFT_1` table as shown in the following two Figures, connecting the two resulting tables through the `AIRCRAFT_1` table's foreign key `MOD_CODE`.

AC_NUM	MOD_CODE	AC_TOTAL_TIME
1178R	C-172	4512.6
205IY	C-172	5325.5
2087V	C-152	4889.8
226BR	C-172	4299.6
2867W	PA28-181	3267.4
3213R	PA28-181	2517.9
4112E	PA28-181	5211.3
45ZU	C-152	7003.1
5725Y	C-172	3968.2

MOD_CODE	MOD_MANUFACTURER	MOD_SEATS	MOD_RENT_CHG
C-152	Cessna	2	$51.75
C-172	Cessna	4	$58.50
PA28-181	Piper	4	$64.00

- Note that a rental rate change need be made in only one place, a description is given in only one place, and so on. No more data update and delete anomalies and no more data integrity problems. The relational schema in the following Figure shows how the two tables are related.



The First normal form (1NF)

- A table in which the intersection of every column and record contains only one value. It prohibits nesting or repeating groups in table. The intersection must be atomic.
- For example the `telNos` column contains multiple values.

branchNo	branchAddress	telNos
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112

Primary key

More than one value, so *not* in 1NF

Branch (Not 1NF)

branchNo	branchAddress	telNos
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112

Take copy of branchNo column to new table to become foreign key

Remove telNos column and create new column called telNo in the new table

Branch (1NF)

branchNo	branchAddress
B001	8 Jefferson Way, Portland, OR 97201
B002	City Center Plaza, Seattle, WA 98122
B003	14 – 8th Avenue, New York, NY 10012
B004	16 – 14th Avenue, Seattle, WA 98128

Primary key

BranchTelephone (1NF)

branchNo	telNo
B001	503-555-3618
B001	503-555-2727
B001	503-555-6534
B002	206-555-6756
B002	206-555-8836
B003	212-371-3000
B004	206-555-3131
B004	206-555-4112

Becomes foreign key

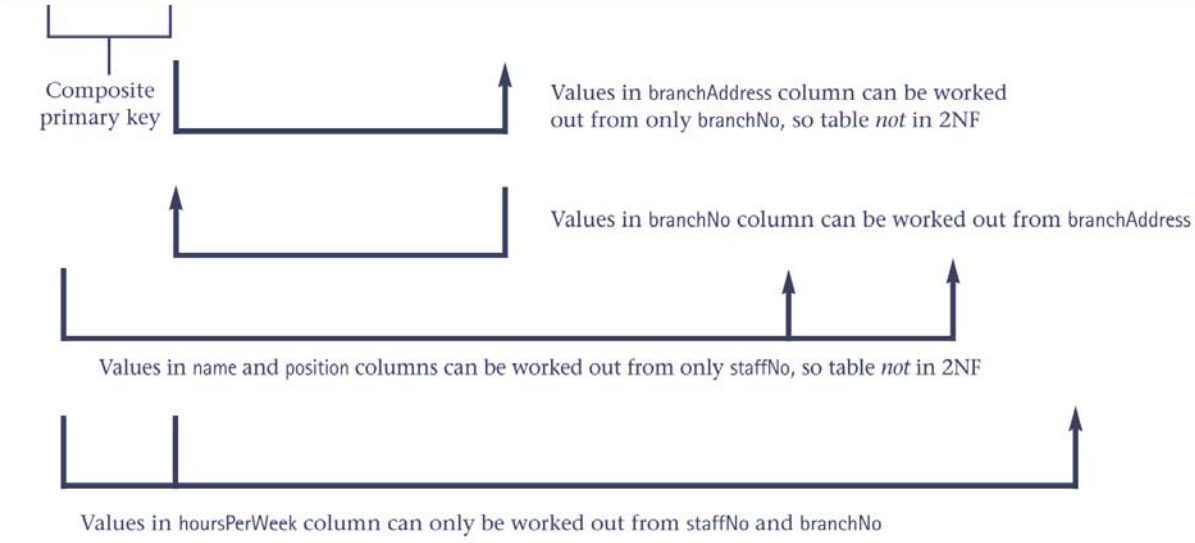
Becomes primary key

The Second normal form (2NF)

- 2NF ONLY applies to tables with **composite primary keys** (more than one primary key).
- A table that is in 1NF and in which the values of each **non-primary-key column** can be worked out from the values in ALL the columns that make up the primary key.

- A table is in 2NF if **each non-key (non primary and/or candidate keys) column depends on ALL candidate keys**, **NOT on a subset of ANY candidate key**.
- The 2NF violation occurs when Functional Dependency (FD) in which part of key (instead of the whole keys) determines a non-key. An FD containing a single column Left Hand Side (LHS) cannot violate 2NF.
- For example, `TempStaffAllocation` table in the following Figure is in 2NF because `branchAddress` can depend on `branchNo` only not both of `staffNo` AND `branchNo` (`staffNo` & `branchNo` are candidate keys and at the same time can be primary keys and at the same time is composite key because more than one primary keys). Another one is the values in `name` and `position` columns can depend on (can stand on it own) the `staffNo` ONLY not both of `staffNo` and `branchNo`. What we want is the `hoursperWeek` column, which depends on both `staffNo` and `branchNo`. In another word we must avoid the partial dependencies on the candidate keys.

staffNo	branchNo	branchAddress	name	position	hoursPerWeek
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10



The Functional dependency

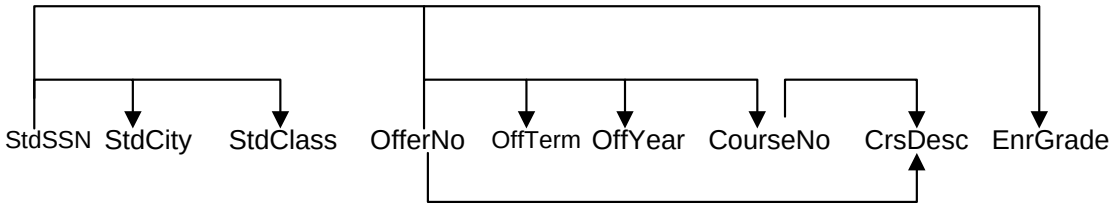
- The particular **relationships that we show between the columns of a table** are more formally referred to as **functional dependencies** (FDs). FD describes the relationship between columns in a table.
- Another example, consider a table with columns A and B, where B is functionally dependent on A (denoted $A \rightarrow B$). If we know the value of A, we find only one value of B in all the records that has this value of A, at any moment in time.

FD Definition

- $X \rightarrow Y$.

- X (functionally) determines Y or Y is functionally dependent on X.
- X: left-hand-side (LHS) or determinant.
- For each X value, there is at most one Y value.
- Similar to candidate keys.

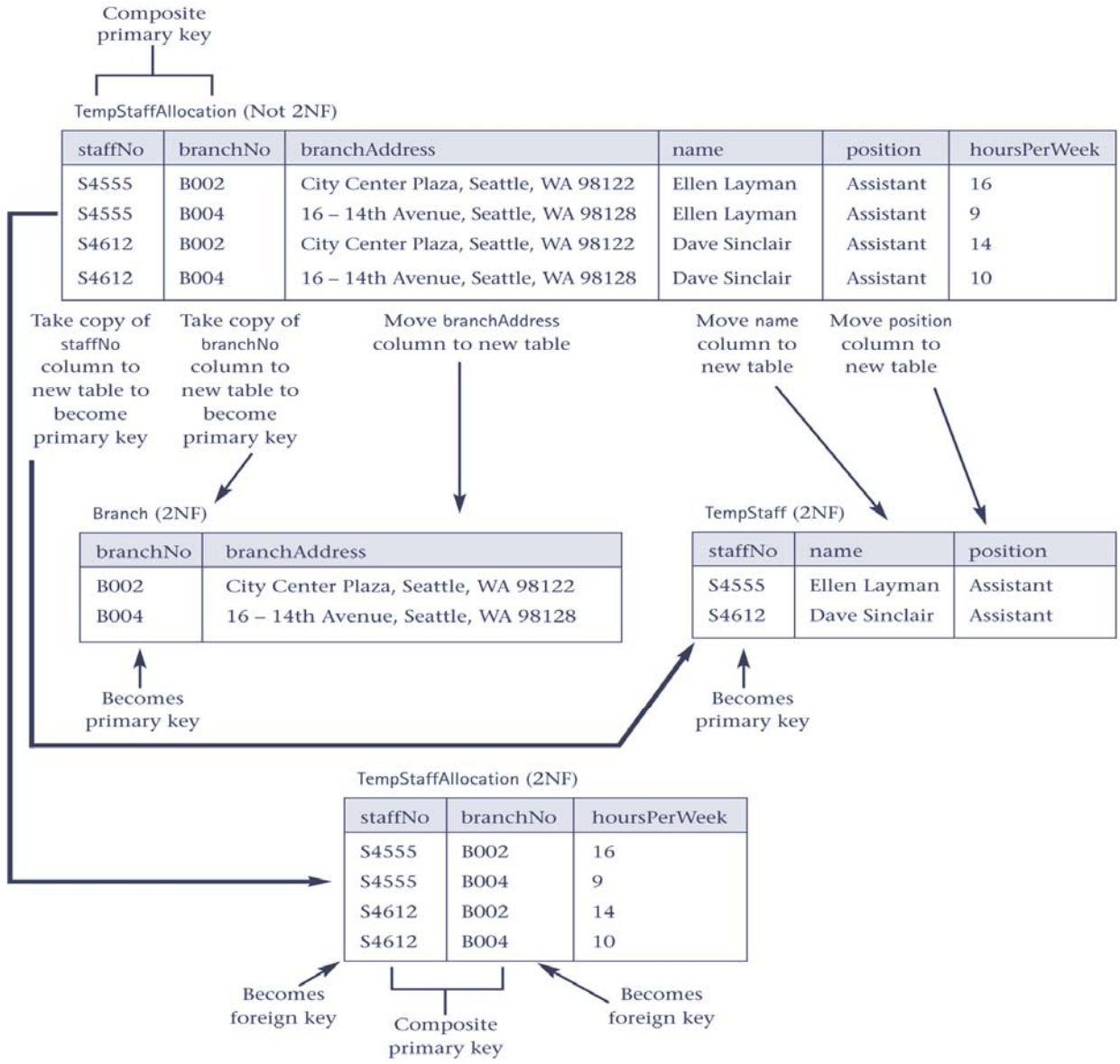
- For example (take note regarding the arrow flow!):



- The FDs are (another notation used to write FDs):

```
StdSSN → StdCity, StdClass
OfferNo → OffTerm, OffYear, CourseNo, CrsDesc
CourseNo → CrsDesc
StdSSN, OfferNo → EnrGrade
```

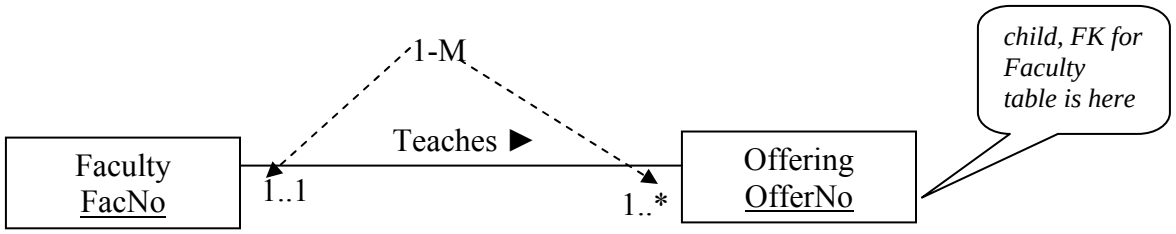
- Formal definition of 2NF is a table that is in 1NF and every non-primary-key column is **fully functional dependent** on the primary key.
- **Full functional dependency** indicates that if A and B are columns of a table, B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.
- Consider the following examples.

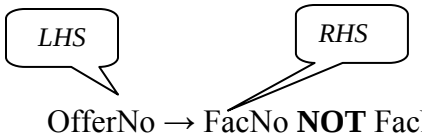


Identifying Functional Dependencies

- Database designers must be able to identify FD when collecting database requirements.
- In problem narratives, some FD can be identified by statements about uniqueness. For example a user may state that each course offering has a unique offering number along with the year and term of the offering. From this statement, the designer should assert that `OfferNo` → `OffYear` and `OffTerm`.
- You can also identify functional dependencies in a table design resulting from the conversion of an ERD. FD would be asserted for each unique column (PK or other candidate key) with the unique column as the LHS and other columns in the table on the RHS.
- Although FD derived from statements about 1-M relationships can be identify, FD derived from statements about 1-M relationship can be confusing to identify.
- When you see a statement about a 1-M relationship, the FD is derived from the child-to-parent direction, not the parent-to-child direction.

- For example, the statement "A faculty teaches many offerings but an offering is taught by one faculty" defines a FD from a unique column of offering to a unique column of faculty such as `OfferNo` → `FacNo`.
- Novice designers sometimes incorrectly assert that `FacNo` determines a collection of `OfferNo` values. This statement is not correct because a **FD must allow at most one associated value**, not a collection of values.





OfferNo → FacNo **NOT** FacNo → OfferNo

- FD in which the LHS is not a PK or candidate key can also be difficult to identify.
- These FDs are especially important to identify after converting an ERD to a table design. You should carefully look for FDs in which the LHS is not a candidate key or primary key.
- You should also consider FDs in tables with a combined PK or candidate key in which the LHS is part of a key, but not the entire key. The NFs these kinds of FDs can lead to modification anomalies.
- Another important consideration in asserting FD is the minimalism of LHS. It is important to distinguish when one column alone is the determinant versus a combination of column.
- An FD in which the LHS contains more than one column usually represents an M-N relationship. For example, the statement "The order quantity is collected for each product purchased in an order" translates to the FD `OrdNo`, `ProdNo` → `OrdQty`.
- Order quantity depends on the combination of order number and product number, not just one of these columns.
- Part of the confusion about the minimalism of the LHS is due to the meaning of columns in the left-hand versus right-hand side of a dependency. The minimal determinant means the determinant (column(s) appearing on the LHS of FD) must not contain extra columns and this minimalism requirement is similar to the minimalism requirement for candidate keys.
- For example, to record that student Social Security Number determines city and class, you can write either:

  `StdSSN` → `StdCity, StdClass` (more compact) or

  `StdSSN` → `StdCity` and `StdSSN` → `StdClass` (less compact)

- If you assume that the email address is also unique for each student, then you can write:

  `Email` → `StdCity, StdClass`

- You should not write:

```
StdSSN, Email → StdCity, StdClass
```

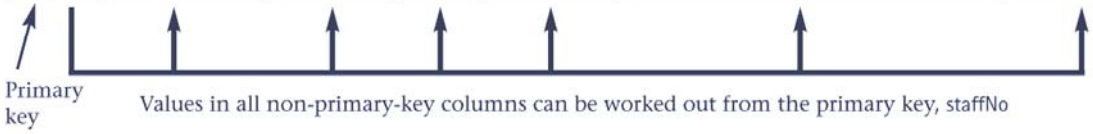
- This is because these FDs imply that the combination of `StdSSN` and `Email` is the determinant. Thus you should write FDs so that the LHS does not contain unneeded columns.
- The prohibition against unneeded columns for determinants is the same as the prohibition against unneeded columns in candidate keys. Both determinants and candidate keys must be minimal.
- A FD cannot be proven to exist by examining the rows of a table. However you can falsify a FD (i.e. prove that a FD does not exist) by examining the contents of a table.
- For example, in the university database, we can conclude that `StdClass` does not determine `StdCity` because there are two rows with the same value for `StdClass` but a different value for `StdCity`.
- Thus, it is sometimes helpful to examine sample rows in a table to eliminate potential functional dependencies.
- There are several commercial database design tools that automate the process of eliminating dependencies through examination of sample rows. Ultimately, the database designer must make the final decision about FDs that exist in a table.

Third normal form (3NF)

- A table that is in 1NF and 2NF and in which **all non-primary-key column can be worked out from only the primary key column(s) and no other columns**.
- At this level, the combined definition of 2NF and 3NF is a table is in 3NF if each non-key column depends on all candidate keys, whole candidate keys and nothing but candidate keys.
- For 2NF we should remove **partial dependency** and for 3NF we should remove **transitive dependency**.
- For example the `StaffBranch` table is not in 3NF.

StaffBranch (Not 3NF)

staffNo	name	position	salary	branchNo	branchAddress	telNo
S1500	Tom Daniels	Manager	46000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0003	Sally Adams	Assistant	30000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0010	Mary Martinez	Manager	50000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S3250	Robert Chin	Supervisor	32000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S2250	Sally Stern	Manager	48000	B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131
S0415	Art Peters	Manager	41000	B003	14 – 8th Avenue, New York, NY 10012	212-371-3000



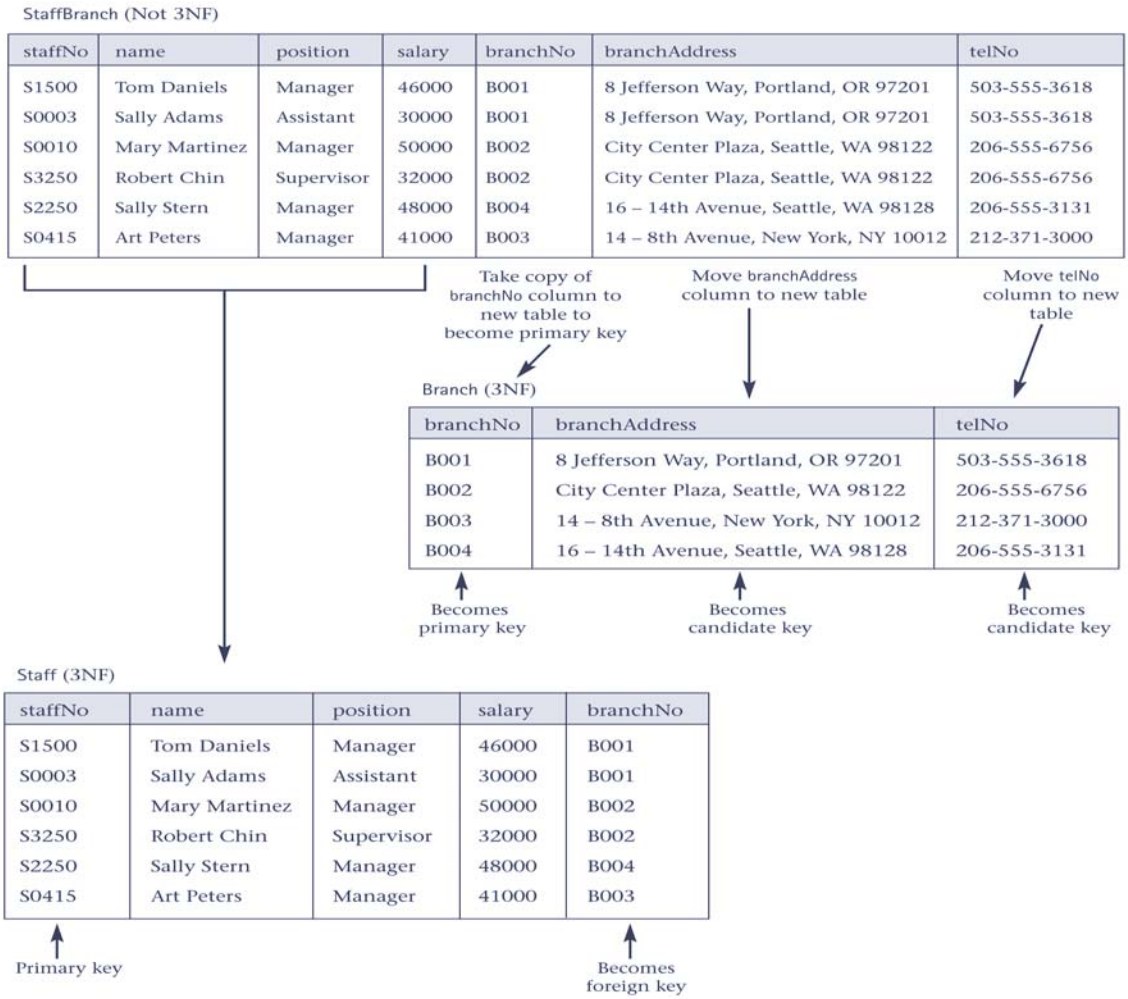
Values in all non-primary-key columns can be worked out from the primary key, staffNo

Values in branchAddress and telNo columns can be worked out from branchNo, so table *not* in 3NF

Values in branchNo and telNo columns can be worked out from branchAddress, so table *not* in 3NF

Values in branchNo and branchAddress columns can be worked out from telNo, so table *not* in 3NF

- The formal definition of 3NF is a table that is in 1NF and 2NF and in which no non-primary-key column is transitively dependent on the primary key.
- For example, consider a table with A, B, and C. If B is functional dependent on A ($A \rightarrow B$) and C is functional dependent on B ($B \rightarrow C$), then C is **transitively dependent** on A via B (provided that A is not functionally dependent on B or C).
- If a transitive dependency exists on the primary key, the table is not in 3NF.



Reviews Questions & Answers

1. Describe the characteristics of a table that violates first normal form (1NF) and then describe how such a table is converted to 1NF.

   The rule for first normal form (1NF) is a table in which the intersection of every column and record contains only one value. In other words a table that contains more than one atomic value in the intersection of one or more column for one or more records is not in 1NF. The non 1NF table can be converted to 1NF by restructuring original table by removing the column with the multi-values along with a copy of the primary key to create a new table. The advantage of this approach is that the resultant tables may be in normal forms later that 1NF.

2. What is the minimal normal form that a relation must satisfy? Provide a definition for this normal form.

   Only first normal form (1NF) is critical in creating appropriate tables for relational databases. All the subsequent normal forms are optional. However, to avoid the update anomalies, it's normally recommended that you proceed to third normal form (3NF).

First normal form (1NF) is a table in which the intersection of every column and record contains only one value.

3. Describe an approach to converting a first normal form (1NF) table to second normal form (2NF) table(s).

   Second normal form applies only to tables with composite primary keys, that is, tables with a primary key composed of two or more columns. A 1NF table with a single column primary key is automatically in at least 2NF.
   A second normal form (2NF) is a table that is already in 1NF and in which the values in each non-primary-key column can be worked out from the values in all the columns that makes up the primary key.
   A table in 1NF can be converted into 2NF by removing the columns that can be worked out from only part of the primary key. These columns are placed in a new table along with a copy of the part of the primary key that they can be worked out from.

4. Describe the characteristics of a table in second normal form (2NF).

   Second normal form (2NF) is a table that is already in 1NF and in which the values in each non-primary-key column can only be worked out from the values in all the columns that make up the primary key.

5. Describe what is meant by full functional dependency and describe how this type of dependency relates to 2NF. Provide an example to illustrate your answer.

   The formal definition of second normal form (2NF) is a table that is in first normal form and every non-primary-key column is fully functionally dependent on the primary key. Full functional dependency indicates that if A and B are columns of a table, B is fully functionally dependent on A, if B is not dependent on any subset of A. If B is dependent on a subset of A, this is referred to as a partial dependency. If a partial dependency exists on the primary key, the table is not in 2NF. The partial dependency must be removed for a table to achieve 2NF.

6. Describe the characteristics of a table in third normal form (3NF).

   Third normal form (3NF) is a table that is already in 1NF and 2NF, and in which the values in all non-primary-key columns can be worked out from only the primary key (or candidate key) column(s) and no other columns.

7. Describe what is meant by transitive dependency and describe how this type of dependency relates to 3NF. Provide an example to illustrate your answer.

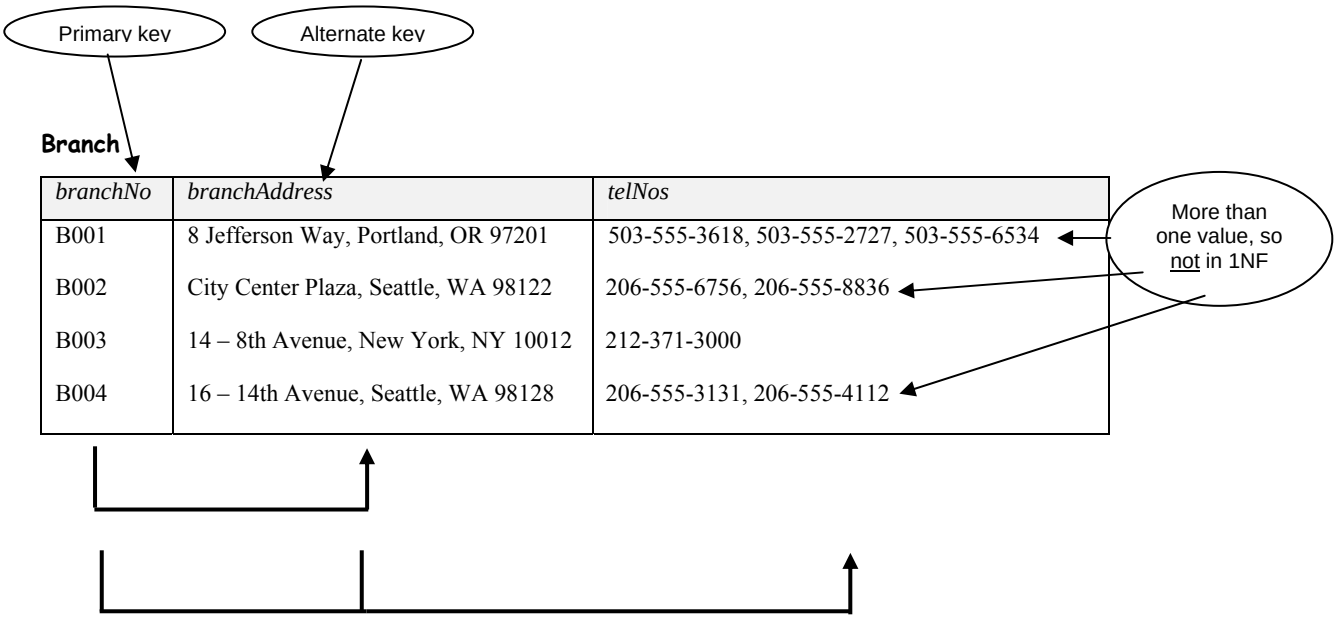
   The formal definition for third normal form (3NF) is a table that is in first and second normal forms and in which no non-primary-key column is transitively dependent on the primary key. Transitive dependency is a type of functional dependency that occurs when a particular type of relationship holds between columns of a table. For example, consider a table with columns A, B, and C. If B is functionally dependent on A ($A \rightarrow B$) and C is functionally dependent on B ($B \rightarrow C$), then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C). If a transitive dependency exists on the primary key, the table is not in 3NF. The transitive dependency must be removed for a table to achieve 3NF.

8. Examine the table shown below.

branchNo	*branchAddress*	*telNos*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112

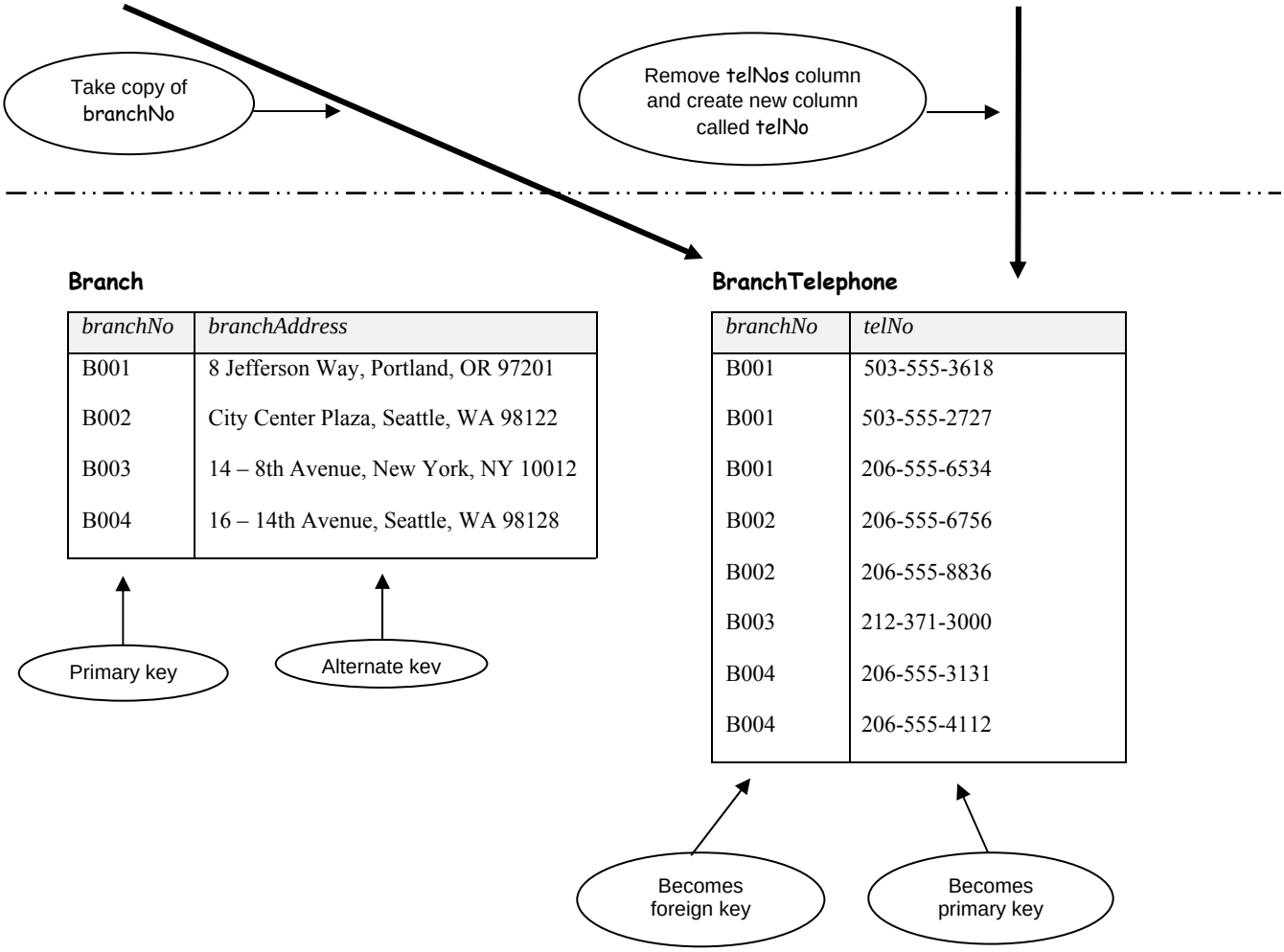
(a) Why is this table not in 1NF?
(b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).
(c) Identify the primary, alternate and foreign keys in your 3NF relations.

Answer:



Branch

branchNo	*branchAddress*	*telNos*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112



Branch

branchNo	*branchAddress*
B001	8 Jefferson Way, Portland, OR 97201
B002	City Center Plaza, Seattle, WA 98122
B003	14 – 8th Avenue, New York, NY 10012
B004	16 – 14th Avenue, Seattle, WA 98128

BranchTelephone

branchNo	*telNo*
B001	503-555-3618
B001	503-555-2727
B001	206-555-6534
B002	206-555-6756
B002	206-555-8836
B003	212-371-3000
B004	206-555-3131
B004	206-555-4112

(Note: There is an alternative approach to altering the original `Branch` table – columns can be added to the original `Branch` table to hold the individual values for each telephone number, e.g. `telNo1`, `telNo2` and `telNo3`.)

9. Examine the table shown below.

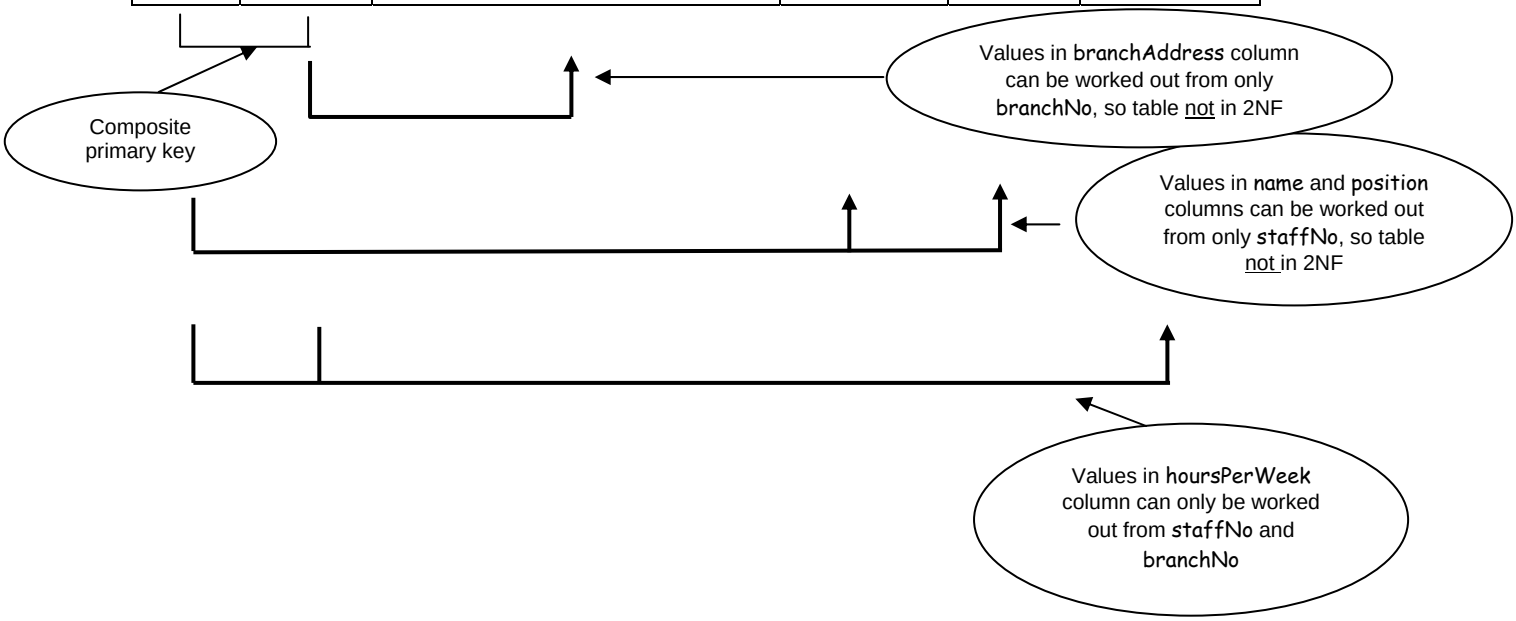
staffNo	*branchNo*	*branchAddress*	*name*	*position*	*hoursPerWeek*
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10

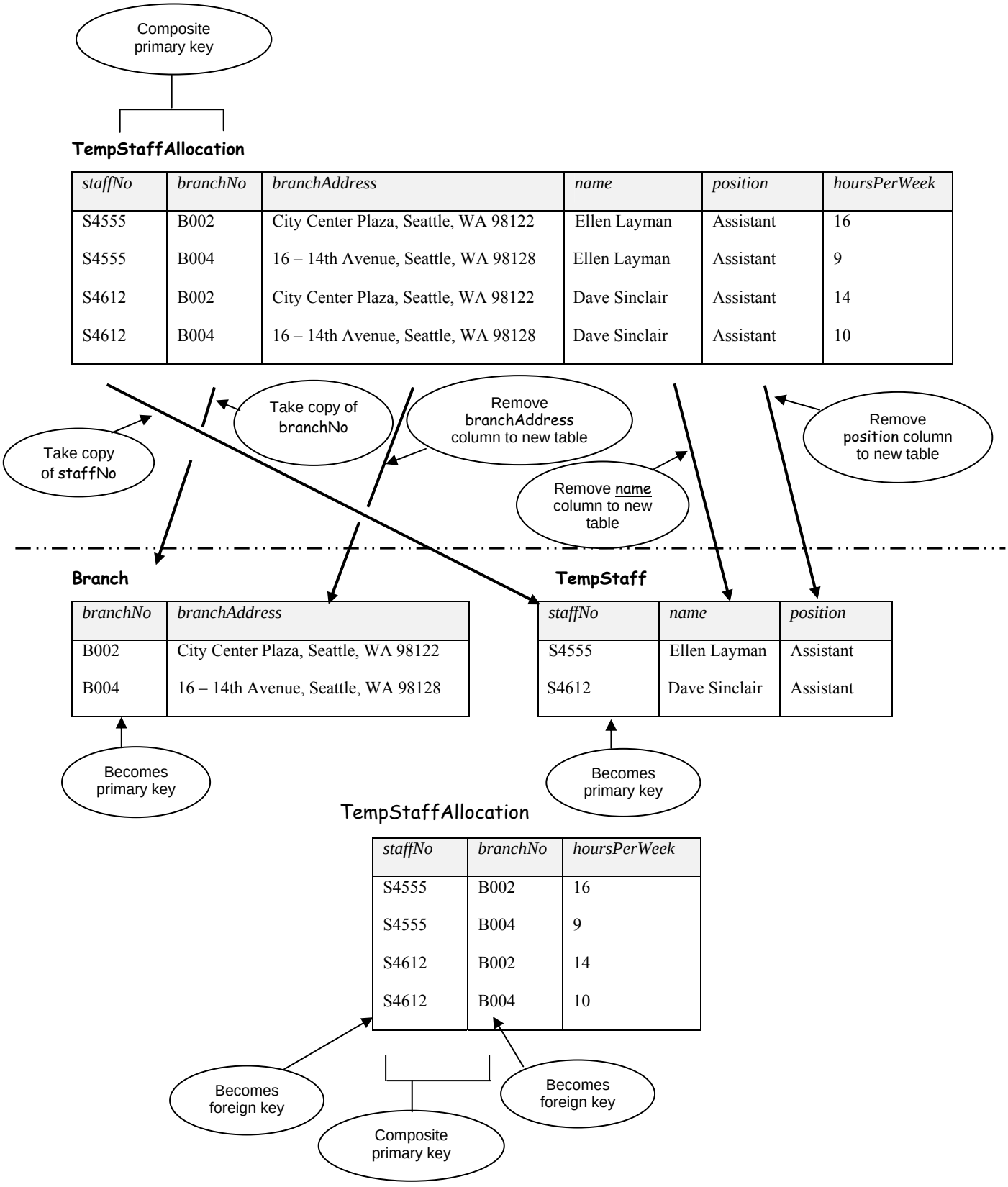
(a) Why is this table not in 2NF?
(b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).
(c) Identify the primary, (alternate) and foreign keys in your 3NF relations.

Answer:

TempStaffAllocation

staffNo	*branchNo*	*branchAddress*	*name*	*position*	*hoursPerWeek*
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10




Composite primary key
TempStaffAllocation
staffNo
branchNo
branchAddress
name
position
hoursPerWeek
S4555
B002
City Center Plaza, Seattle, WA 98122
Ellen Layman
Assistant
16
S4555
B004
16 – 14th Avenue, Seattle, WA 98128
Ellen Layman
Assistant
9
S4612
B002
City Center Plaza, Seattle, WA 98122
Dave Sinclair
Assistant
14
S4612
B004
16 – 14th Avenue, Seattle, WA 98128
Dave Sinclair
Assistant
10
Take copy of staffNo
Take copy of branchNo
Remove branchAddress column to new table
Remove name column to new table
Remove position column to new table
Branch
branchNo
branchAddress
B002
City Center Plaza, Seattle, WA 98122
B004
16 – 14th Avenue, Seattle, WA 98128
TempStaff
staffNo
name
position
S4555
Ellen Layman
Assistant
S4612
Dave Sinclair
Assistant
Becomes primary key
Becomes primary key
TempStaffAllocation
staffNo
branchNo
hoursPerWeek
S4555
B002
16
S4555
B004
9
S4612
B002
14
S4612
B004
10
Becomes foreign key
Becomes foreign key
Composite primary key

10. Examine the table shown below.

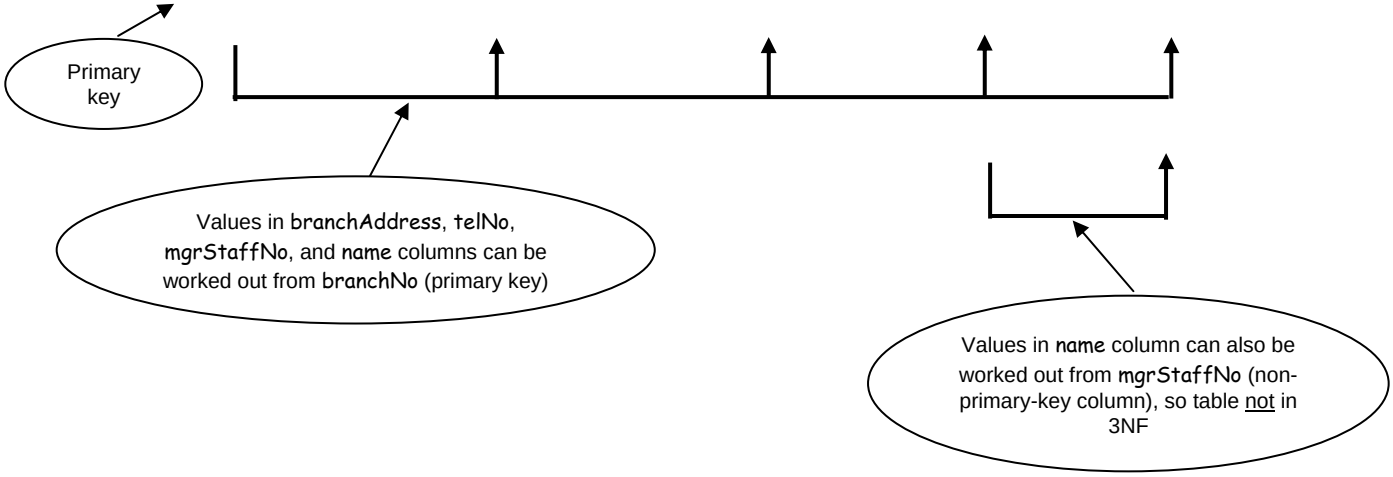
branchNo	*branchAddress*	*telNo*	*mgrStaffNo*	*name*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500	Tom Daniels
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010	Mary Martinez
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0145	Art Peters
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250	Sally Stern

(a) Why is this table not in 3NF?
(b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).
(c) Identify the primary, (alternate) and foreign keys in your 3NF relations.

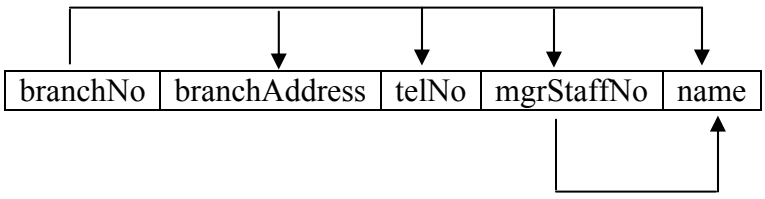
Answer:

BranchManager

branchNo	*branchAddress*	*telNo*	*mgrStaffNo*	*name*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500	Tom Daniels
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010	Mary Martinez
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0145	Art Peters
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250	Sally Stern

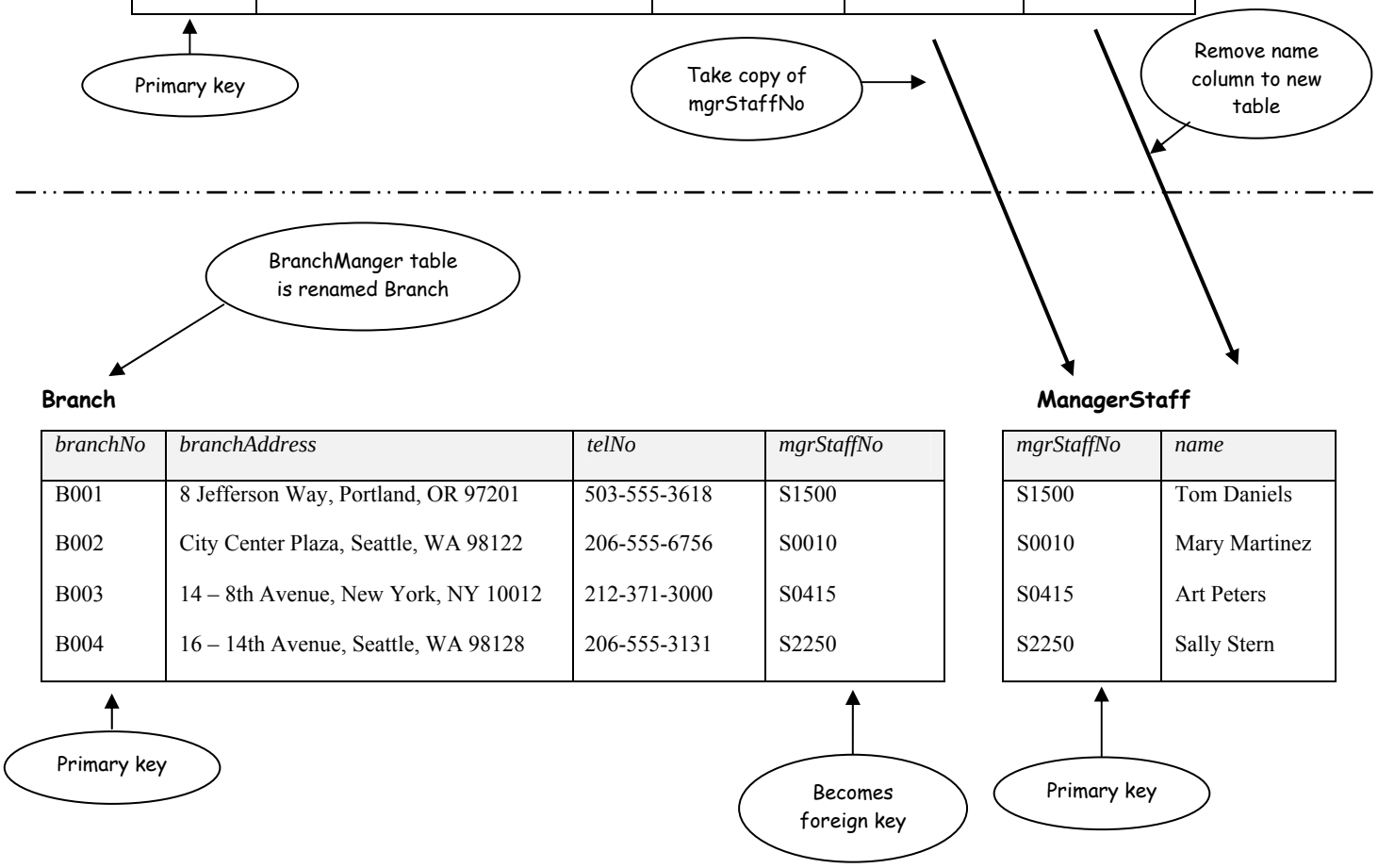


Using Dependency Diagram, it can be represented as shown in the following Figure:



BranchManager

branchNo	*branchAddress*	*telNo*	*mgrStaffNo*	*name*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500	Tom Daniels
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010	Mary Martinez
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0415	Art Peters
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250	Sally Stern



Branch

branchNo	*branchAddress*	*telNo*	*mgrStaffNo*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0415
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250

ManagerStaff

mgrStaffNo	*name*
S1500	Tom Daniels
S0010	Mary Martinez
S0415	Art Peters
S2250	Sally Stern

More Questions and Answers

1. What is normalization?

   Normalization is the process for assigning attributes to entities. Properly executed, the normalization process eliminates uncontrolled data redundancies, thus eliminating the data anomalies and the data integrity problems that are produced by such redundancies. Normalization does not eliminate data redundancy; instead, it produces the carefully controlled redundancy that lets us properly link database tables.

2. When is a table in 1NF?

A table is in 1NF when all the key attributes are defined (no repeating groups in the table) and when all remaining attributes are dependent on the primary key. However, a table in 1NF still may contain partial dependencies, i.e., dependencies based on only part of the primary key.

3. When is a table in 2NF?

A table is in 2NF when it is in 1NF and it includes no partial dependencies. However, a table in 2NF may still have transitive dependencies, i.e., dependencies based on attributes that are not part of the primary key.

4. When is a table in 3NF?

A table is in 3NF when it is in 2NF and it contains no transitive dependencies.

5. When is a table in BCNF?

A table is in Boyce-Codd Normal Form (BCNF) when it is in 3NF and every determinant in the table is a candidate key. For example, if the table is in 3NF and it contains a nonprime attribute that determines a prime attribute, the BCNF requirements are not met. This description clearly yields the following conclusions:

- If a table is in 3NF and it contains only one candidate key, 3NF and BCNF are equivalent.
- BCNF can be violated only if the table contains more than one candidate key. Putting it another way, there is no way that the BCNF requirement can be violated if there is only one candidate key.

6. Given the dependency diagram shown in the following Figure, answer the following questions.

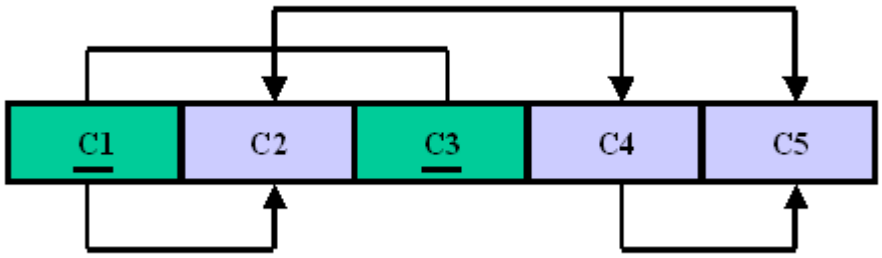


Figure: The Dependency Diagram

a. Identify and discuss each of the indicated dependencies.

- C1 → C2 represents a partial dependency, because C2 depends only on C1, rather than on the entire primary key composed of C1 and C3.
- C4 → C5 represents a transitive dependency, because C5 depends on an attribute (C4) that is not part of a primary key.
- C1, C3 → C2, C4, C5 represents a functional dependency, because C2, C4, and C5 depend on the primary key composed of C1 and C3.

DATABASE DESIGN:
Normalization – Exercises & Answers

staffNo	dentistName	patientNo	patientName	appointment		surgeryNo
				date	time	
S1011	Tony Smith	P100	Gillian White	12-Aug-03	10.00	S10
S1011	Tony Smith	P105	Jill Bell	13-Aug-03	12.00	S15
S1024	Helen Pearson	P108	Ian MacKay	12-Sept-03	10.00	S10
S1024	Helen Pearson	P108	Ian MacKay	14-Sept-03	10.00	S10
S1032	Robin Plevin	P105	Jill Bell	14-Oct-03	16.30	S15
S1032	Robin Plevin	P110	John Walker	15-Oct-03	18.00	S13

Figure 1: Details of patient dental appointments.

(a) The table shown in Figure 1 is susceptible to update anomalies. Provide examples of insertion, deletion, and modification anomalies.

Answers:

This table is not well structured, un-normalized containing redundant data. By using a **bottom-up** approach we analyzing the given table for anomalies. First observation, we see multiple values in an appointment column and this of course violate the 1NF. By assuming the staffNo and patientNo as candidate keys, there are many anomalies exist.

Insertion anomalies:

To insert a new patient particular that makes an appointment with the designated Doctor, we need to enter the correct detail for the staff. For example, to insert the details of new patient in patientNo, patientName and an appointment, we must enter the correct details of the doctor (staffNo, dentistName) so that the patient details are consistent with values for the designated Doctor for example, S1011.
To enter new patient data that doesn't have Doctor to be assigned we can't insert `NULL` values for the primary key.

Deletion anomalies:

If we want to delete a patient named Ian MacKay for example, two records need to be deleted as in row 3 and 4. This anomaly also obvious when we want to delete the dentistName, multiple records needs to be deleted to maintain the data integrity.
When we delete a Dentist record, for example Tony Smith, the details about his patients also lost from the database.

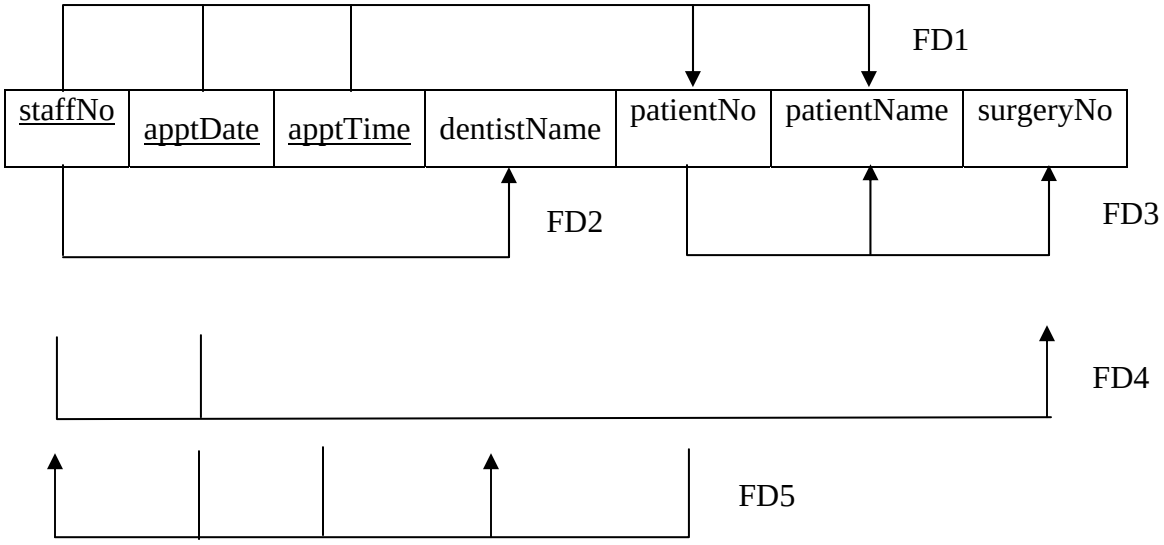
Modification anomalies:

With redundant data, when we want to change the value of one columns of a particular Dentist, for example the dentistName, we must update all the Dentist records that assigned to the particular patient otherwise the database will become inconsistent. We also need to modify the appointment schedules because different Dentist has different schedules.

(b) Describe and illustrate the process of normalizing the table shown in Figure 1 to 3NF. State any assumptions you make about the data shown in this table.

Assumptions made include that a patient is registered at only one surgery and he/she may have more than one appointment on a given day. All the schedules have been fixed for the whole days and week.

In the 1NF we remove all the repeating groups (appointment), assigning new column (apptDate and apptTime) and assigned primary keys (candidate keys). Then we figure out the functional dependencies (FDs). By using dependency diagram we represent the table as shown below. (NF – stand for Normal Form)
Note: How to find the FDs is subjective!!! However, the rule is, it must reflect the real word situation.



FD1 is already in 2NF. In this case, we can see that FD2 (just depend on staffNo) and FD4 (just depend on staffNo and apptDate) violate the 2NF. These two NFs are partially dependent on the candidate keys not the whole keys. FD2 can stand on its own by depending on the staffNo and meanwhile FD4 also can stand on its own by depending on the staffNo.
The FD3 violates the 3NF showing the transitive dependency where surgeryNo and patientName depend on patientNo while patientNo depend on the staffNo that is the non-key is depending on another non-key.

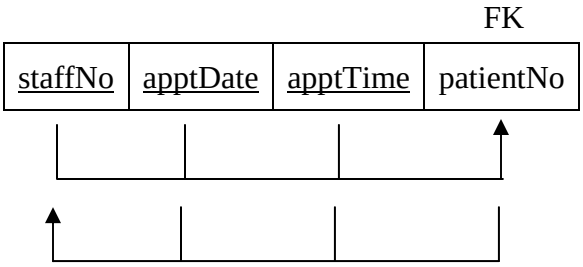
The 2NF, it is already in 1NF and there is no **partial dependency**. So we need to remove the FD2 and FD4 by splitting into new tables and at the same time creating foreign keys. The new tables that are in 2NF are shown below.

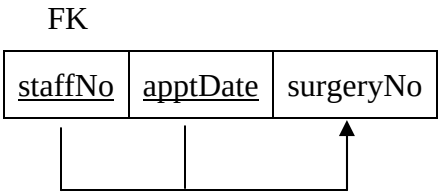
staffNo	apptDate	apptTime	patientNo	patientName

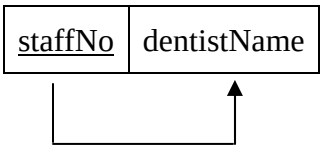
staffNo	apptDate	surgeryNo

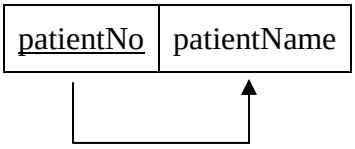
staffNo	dentistName

Finally in 3NF we must remove the transitive dependency. In this case we remove the FD3 by splitting into a new table. The transitive dependency left is the patientName that depend on the patientNo, so we split this into new table while creating a foreign key.









Let re-arrange and giving names to the tables.

staffNo	dentistName

Dentist(staffNo, dentistName)

FK

staffNo	apptDate	surgeryNo

Surgery(staffNo, apptDate, surgeryNo)

patientNo	patientName

Patient(patientNo, patientName)

FK

staffNo	apptDate	apptTime	patientNo

Appointment(staffNo, apptDate, apptTime, patientNo)

An agency called *InstantCover* supplies part-time/temporary staff to hotels throughout Scotland. The table shown in Figure 2 lists the time spent by agency staff working at two hotels. The National Insurance Number (NIN) is unique for employee.

NIN	contractNo	hoursPerWeek	eName	hotelNo	hotelLocation
113567WD	C1024	16	John Smith	H25	Edinburgh
234111XA	C1024	24	Diane Hocine	H25	Edinburgh
712670YD	C1025	28	Sarah White	H4	Glasgow
113567WD	C1025	16	John Smith	H4	Glasgow

Figure 2: Employees of *InstantCover* and their contracts to work at hotels.

(1) The table shown in Figure 2 is susceptible to update anomalies. Provide examples of insertion, deletion, and modification anomalies.

Answers:

It is obvious that the NIN and contractNo can be candidate keys. There are data redundancies in term of repeating groups.

Insertion anomalies:

To insert the details of new hotel we must know the NIN of the staff because this is primary key and we cannot assign NULL value for primary key. For example if we want to insert the details of new hotel, we must also insert the NIN and contractNo.
To recruit a new staff we need to know the details of hotel. For example if we want to recruit and assign new staff at H4 we need to enter the correct details of H4 so that the hotel details are consistent with values for hotel H4.

Deletion anomalies:

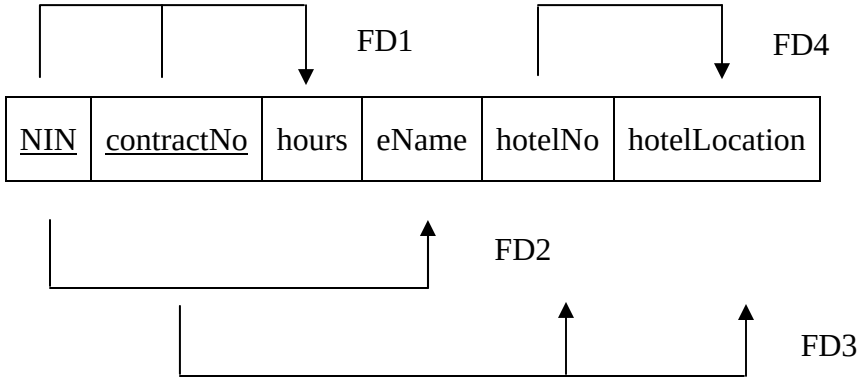
If we delete a record from the table that represent the staff name for example, John Smith, the details about the contract also will be lost from the database and will affect other record as seen in the second row of the contractNo (C1024).
Oppositely, when we delete a contractNo, other staff details will also be lost. For example if we delete C1025, both Sarah White and John Smith details will be lost.

Modification anomalies:

If we change the value of the contractNo, several records need to changed, for example row 1 and 2, also row 3 and 4 if we edit either contractNo.
If we edit the staff details, for example updating John Smith, several records need to be updated. This also applies to the hotelNo column.

(2) Describe and illustrate the process of normalizing the table shown in Figure 2 to 3NF. State any assumptions you make about the data shown in this table.

In the 1NF we remove all the repeating groups if any and assigned primary keys (candidate keys). Then we figure out the functional dependencies (FDs). By using dependency diagram we represent the table as shown below. (NF –stand for Normal Form)



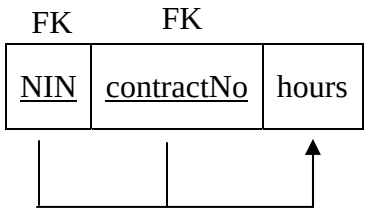
Obviously FD2 and FD3 violate the 2NF (partial dependency) and FD4 violate 3NF (transitive dependency). In the 2NF we remove the partial dependency of the FD2 and FD3 by splitting it into new tables as shown below.

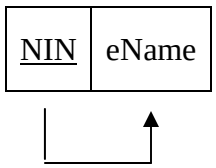
NIN	contractNo	hours

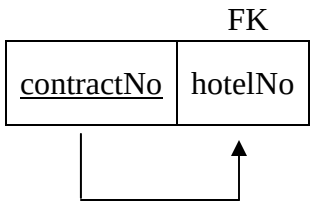
NIN	eName

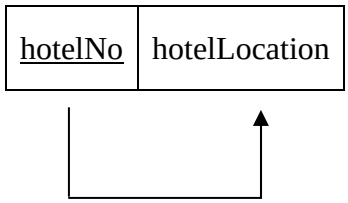
contractNo	hotelNo	hotelLocation

In 3NF we remove the transitive dependency of the FD4 by splitting it into new table as shown below.

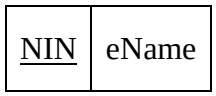




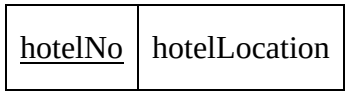




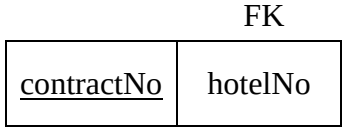
Finally, let re-arrange the table and assign names.



Staff(NIN, eName)



Hotel(hotelNo, hotelLocation)



Contract(contractNo, hotelNo)

FK	FK	
NIN	contractNo	hours

WorkHours(NIN, contractNo, hours)