CPS 510 - Assignment 8

Using Bernstein's algorithm to achieve 3NF

We want to find third normal form decomposition of the the following relation with FDs shown in step 1 which is lossless join and dependency preserving.

User Location(locationID, country, city)
User Location = { {userID} → locationID, country, city }

Step 1: Determine functional dependencies

FD= {locationID → country, country → city}

Step 2a: Break RHS and find redundancies

Left	Middle	Right
locationID	country	city

Closure:
$locationID^{+} = \{locationID,\ country,\ city\}$
$country^{+} = \{country,\ city\}$
$city^{+} = \{\ city\}$

Step 2b: Find and remove partial dependencies (or minimizing LHS)

If there is a partial dependency the algorithm would make us remove the attribute from the LHS. In the case above, there is no compound key so we do not need to to minimize the LHS.

Therefore the table is currently in 2NF.

Step 3: Find keys

We know that *locationID* does not appear on the RHS, and only on the LHS, therefore is part of the key. *City* appears only on the RHS so it should not be part of the key. *Country* is not part of the key as it appears on the RHS and the LHS.

Therefore {locationID} should be part of the keys of the User Location relation.
The closure of *locationID* tells us that locationID can get all the other attributes:
$locationID^{+} = \{locationID,\ country,\ city\}$

Step 4: Make tables

We have the FD= {locationID → country, country → city}, so we make a relation for each functional dependency:

locationID → country: **countries**(locationID, country)
country → city: **cities**(country, city)

To make it lossless join and dependency preserving if there is no key in the countries and cities relation, we need to add at least one key.

Relation 1: **countries**(locationID, country) with FD: locationID → country
Relation 2: **cities**(country, city) with FD: country → city
Relation 3: **locations**(locationID) with no FD

Explanation: Here, no non-candidate key is transitively dependent on any candidate key. Therefore, this is in 3NF.

Using BCNF Algorithm

UnitsLeased(unitID, renterID, rentFrom, rentTo, pricePerUnit, totalPrice)
UnitsLeased = { {unitID, renterID} → rentFrom, rentTo, pricePerUnit, totalPrice }

Step 1: Determine functional dependencies

FD= { {unitID, renterID} → rentFrom, {unitID, renterID} → rentTo, {unitID, renterID} → totalPrice unitID → pricePerUnit}

The functional dependencies above show that our compound primary key is {unitID, renterID}. T

The attribute pricePerUnit is a partial dependency as it is a property of unitID only.

{unitID, renterID} → {pricePerUnit}
{unitID} → {pricePerUnit}

A relation is in BCNF if and only if every nontrivial, left irreducible FD has a candidate key as its determinant.

Step 2: Test if the left hand sides are the keys or not.

$unitID, renterID^{+} = \{unitID,\ renterID, rentFrom,\ rentTo,\ totalPrice\}$ is a key

$unitID^{+} = \{unitID,\ pricePerUnit\}$ is not a key

The FD unitID → pricePerUnit is non-trivial and its LHS is not a superkey. It violates BCNF.

Step 3: Derive BCNF Schema from UnitsLeased

Since UnitsLeased is not BCNF with respect to unitID → pricePerUnit and $unitID^{+} = \{unitID,\ pricePerUnit\}$. We must decompose UnitsLeased to UnitsLeased11 and UnitsLease12.

UnitsLeased11 = $\{unitID,\ renterID, rentFrom,\ rentTo,\ totalPrice\}$
UnitsLeased12 = $\{unitID,\ pricePerUnit\}$ which is BCNF

Step 4: Check if UnitsLeased11 in BCNF with FD11

FD11 =
{unitID, renterID} → rentFrom
{unitID, renterID} → rentTo
{unitID, renterID} → totalPrice

We find the closure of $unitID, renterID$:

$unitID, renterID^{+} = \{unitID,\ renterID, rentFrom,\ rentTo,\ totalPrice\}$ is a key

Therefore, UnitsLeased11 is in BCNF

Final BCNF schema for R is:

UnitsLeased11$(unitID,\ renterID, rentFrom,\ rentTo,\ totalPrice)$ with FD:
{unitID, renterID} → rentFrom
{unitID, renterID} → rentTo
{unitID, renterID} → totalPrice

UnitsLeased12$(unitID,\ pricePerUnit)$
{unitID} → {pricePerUnit}

Explanation:
The above table is in BCNF because every functional dependency is the super key of the table.

For every FD, LHS is a super key. The superkey for UnitsLeased11 is {unitID, renterID} and for UnitsLease12 {unitID} → {pricePerUnit}. This property of the superkeys can be seen above.

OTHER TABLES AND FDS

Users(<u>userID</u>, email, phone, password, username, userTypeID)
Users =
{ {userID} → email, phone, password, username, userTypeID, locationID, f_name, l_name }
{ {email} → userID, phone, password, username, userTypeID, locationID, f_name, l_name }
{ {username} → userID, email, phone, password, userTypeID , locationID, f_name, l_name }

UserTypes(userTypeID, userTypeName)
User Types = { {userTypeID } → userTypeName }
The Grade table is in 3NF. Because userTypeName, the only non-key attribute, is non-transitively dependent on userTypeID.

Grade(<u>gradeID</u>, gradeName)
Grade = { {gradeID} → gradeName }
The Grade table is in 3NF. Because gradeName, the only non-key attribute, is non-transitively dependent on gradeID.

Units(<u>unitID</u>, grade, lastRented, availability)
Units = { {unitID} → grade, lastRented, availability }
The Units table is in 3NF. Because non-key attributes are non-transitively dependent on unitID.

Products(<u>productID</u>, productName, productLocation, productCategory, ownerID, quantityRented, quantityTotal, pricePerUnit)
Products = { {productID} → productName, productLocation, productCategory, ownerID, quantityRented, quantityTotal, pricePerUnit }

The Products table is not in 3NF. Because there are partial dependencies between some non-key attributes. For instance, productLocation is partially dependent on productName.

ProductCategory(<u>productCategoryID</u>, productCategoryName)
Product Category = { {productCategoryID} -> productCategoryName }
The ProductCategory table is in 3NF. Because productCategoryName, the only non-key attribute, is non-transitively dependent on productCategoryID.