CPS 510 - A7

Harshita Singh
Dibbyo Saha
Eqan Ahmad

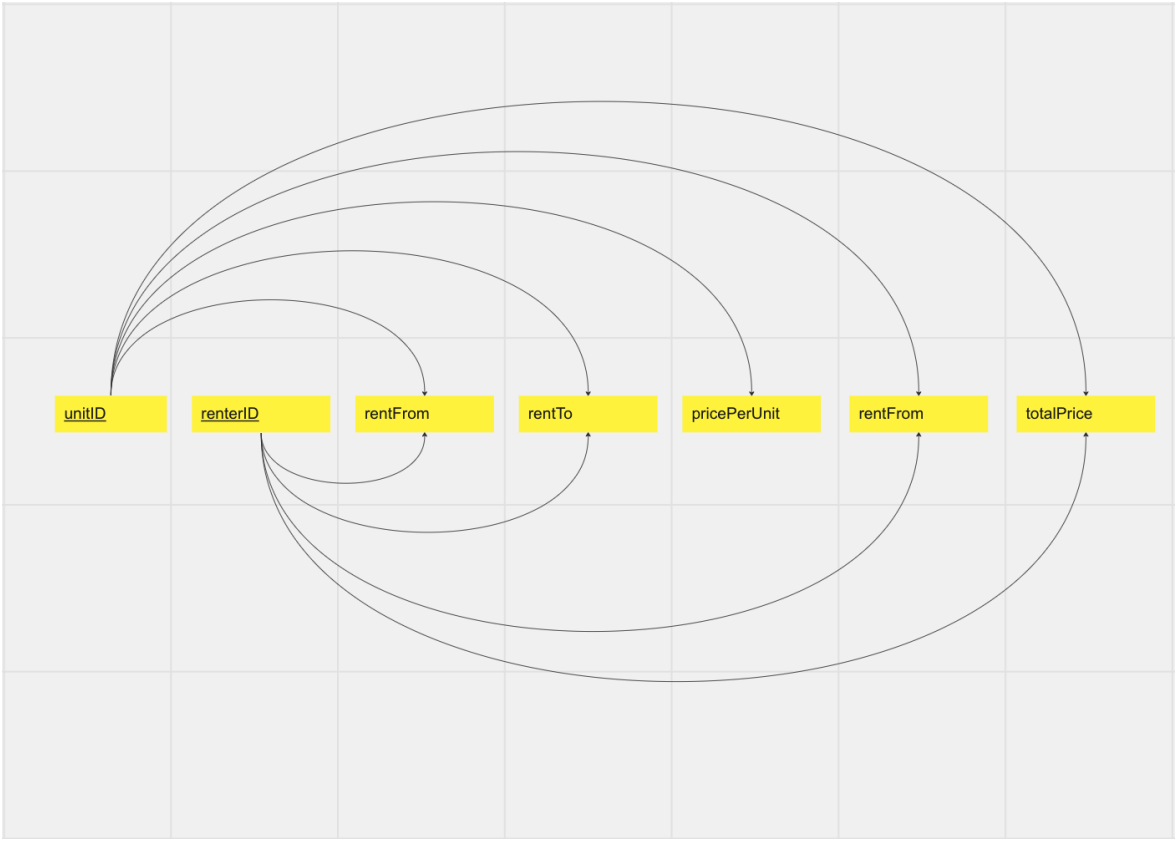
Compound PK

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

Second Normal Form: Each non-key attribute depends on the entire primary key.

In the case above our compound primary key is {unitID, renterID}. The attribute pricePerUnit is a partial dependency as it is a property of unitID only. If we remove the dependency {renterID} from {unitID, renterID} then the dependency will still hold.

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



Transitive Functional Dependency

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

Third Normal Form: in 2nd normal form and every non-key attribute is non transitively dependent on the primary key.

Transitive Dependency:
A→ B→ C and A→C

{locationID} → {country} → {city}

From the above, we can see a transitive dependency. {City} depends on {Country} which depends on {locationID}.

The UserLocation table is not in 3NF. Because there are transitive dependencies between city, locationID, and country.
Countries = { {LocationID} → Country }
Cities = { {Country} → city }

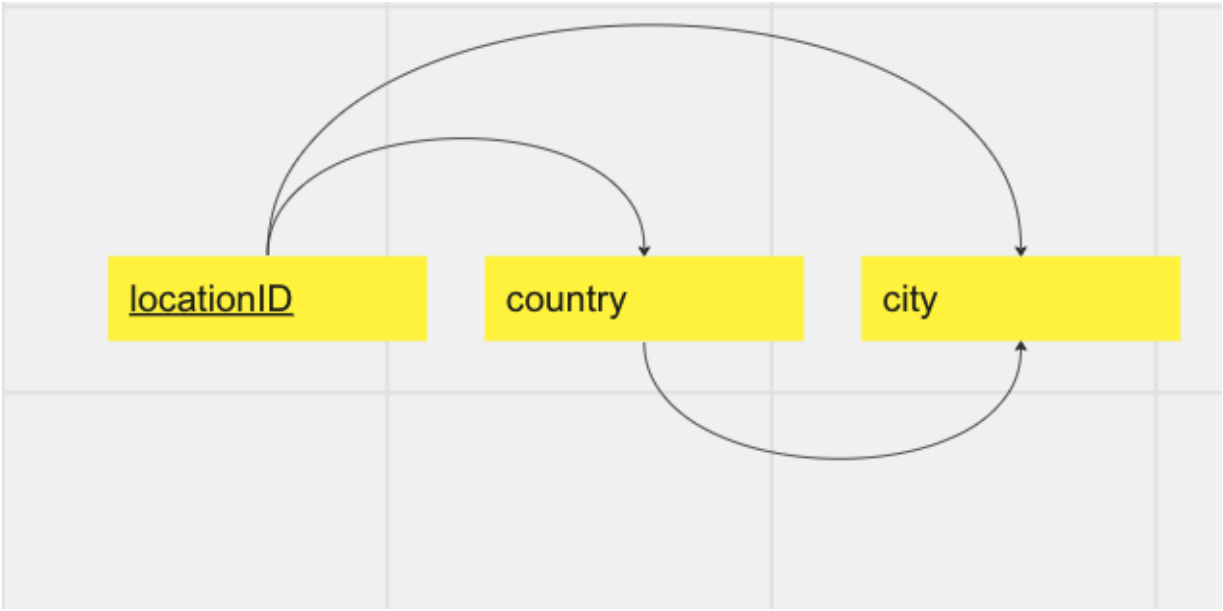


Table Decomposition of Compound PK to 2NF

To decompose the **UnitsLeased** table and remove the partial dependency {unitID} → {pricePerUnit}. We can remove the pricePerUnit attribute from **UnitsLeased** and add the attribute to the **Units** table.

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

New table:

Units(unitID, grade, lastRented, pricePerUnit availability)
Units = { {unitID} → grade, lastRented, pricePerUnit, availability }

UnitsLeased is now in second normal form because each non-key attribute {rentFrom, rentTo, totalPrice } is fully dependent on the primary key {unitID, renterID}. Removing any of the two attributes will lead to the dependency not holding.

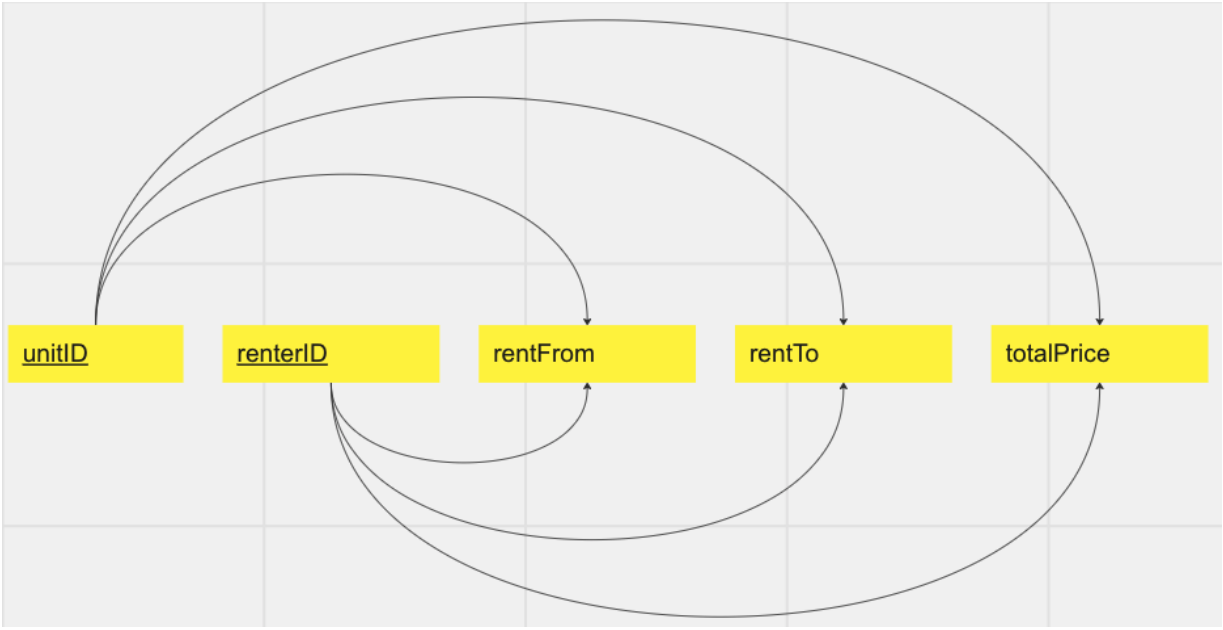


Table Decomposition of Transitive Functional Dependency to 3NF

To decompose the **User Location** table and remove the transitive dependency {locationID} → {country} → {city}, we must create two new tables with the following functional dependencies.

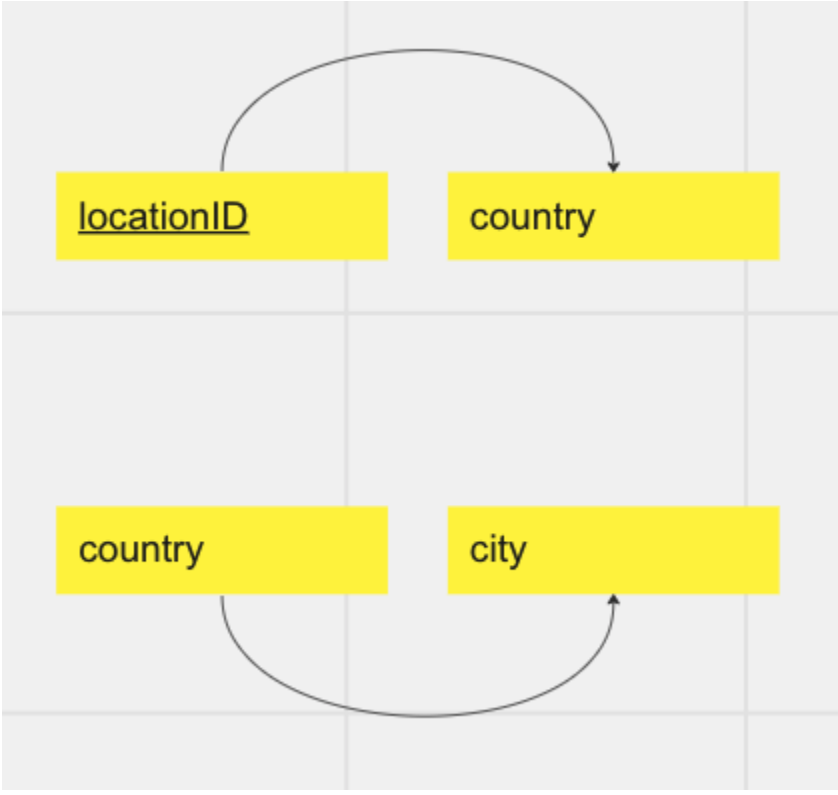
User Location(locationID, country, city)
User Location = { {userID} → locationID, country, city }

New tables:

Countries(LocationID, Country)
Countries = { {LocationID} → Country }

Cities(Country, city)
Cities = { {Country} → city }

The User Location relation is now in 3NF as we have removed the transitive dependency by creating two new relations **Cities** and **Country.**



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.