CPS 510 Assignment 7 - Normalization/3NF

Arshpreet Singh (501030338)
James Tan (501042067)
Sania Syed (501017695)

user_acc(<u>Email</u>, FirstName, LastName, DateOfBirth, Password)
{ Email } → FirstName, LastName, DateOfBirth, Password

This relation is in 3NF because all non-primary keys (FirstName, LastName, DateofBirth, and Password) are fully functionally and non-transitively dependent on the primary key Email.

payment_method(<u>Email, PaymentDetails,</u> MethodType)
{ Email, PaymentDetails } → MethodType

This relation is not in 3NF because the attribute MethodType is partially dependent on the compound primary key i.e, {PaymentDetails} -> MethodType.

We fix this be decomposing the relation into two relations:

uses(<u>Email</u>, PaymentDetails)
{Email} -> PaymentDetails
{Payment} -> Email

payment_method(<u>PaymentDetails</u>, MethodType)
{PaymentDetails} -> MethodType

movie(<u>MovieID</u>, MovieName, Genre, Director, MovieRating, ReleaseYear, Resolution, Runtime, Synopsis, MovieCast)
{ MovieID } → MovieName, Genre, Director, MovieRating, ReleaseYear, Resolution, Runtime, Synopsis, MovieCast

This relation is in 3NF because all non-primary keys (MovieName, Genre, Director, MovieRating, ReleaseYear, Resolution, Runtime, Synopsis and MovieCast) are fully functionally and non-transitively dependent on the primary key MovieID.

orders(<u>Email, MovieID, PaymentDetails</u>)

This is a many-to-many relationship, hence, it has no functional dependencies.

studio(StudioName, ProducedMovies)
{ StudioName } → ProducedMovies

This relation is in 3NF because the non-primary key ProducedMovies is fully functionally and non-transitively dependent on the primary key StudioName.

produces(MovieID, StudioName)

This is a many-to-many relationship, hence, it has no functional dependencies.

rental(MovieID, RentalPrice, RentalDate, RentalExpiry)
{ MovieID } → RentalPrice, RentalDate, RentalExpiry

This relation is in 3NF because all non-primary keys (RentalPrice, RentalDate and Rental Expiry) are fully functionally and non-transitively dependent on the primary key MovieID.

purchase(MovieID, BuyingPrice, PurchaseDate)
{ MovieID } → BuyingPrice, PurchaseDate

This relation is in 3NF because all non-primary keys (BuyingPrice and PurchaseDate) are fully functionally and non-transitively dependent on the primary key MovieID.

movie_review(RevID, WrittenRev, NumericRating, Email, FirstName, LastName)
{ RevID } → WrittenRev, NumericRating, Email, FirstName, LastName
{ Email } -> FirstName, LastName

This relation is not in 3NF because the attributes (FirstName and LastName) are transitively dependent on Email.

To fix this we decompose the relation into two relations:

movie_review(RevID, WrittenRev, NumericRating, Email)
{RevID} -> WrittenRev, NumericRating, Email

user_details(Email, FirstName, LastName)
{Email} -> FirstName, LastName

links_to(RevID, MovieID, MoviePageUrl)
{RevID} -> MovieID

{MovieID} ->RevID
{ RevID, MovieID } → MoviePageUrl

This relation is not in 3NF because the non-key attribute (MoviePageUrl) is partially dependent on the compound primary key i.e, MovieID -> MoviePageUrl.

We fix this by decomposing the relation into two relations:

links_to(RevID,MovieID)
{ RevID } -> MovieID
{ MovieID } -> RevID

movie_url(MovieID, MoviePageUrl)
{MovieID} -> MoviePageUrl

fanclub_membership(MembershipID, RentalDiscount, PurchaseDiscount, RentalSpecialOffers, PurchaseSpecialOffers, OffersRentalStartDate, OffersRentalEndDate, OffersPurchaseStartDate, OffersPurchaseEndDate)
{ MembershipID } → RentalDiscount, PurchaseDiscount, RentalSpecialOffers, PurchaseSpecialOffers, OffersRentalStartDate, OffersRentalEndDate, OffersPurchaseStartDate, OffersPurchaseEndDate

This relation is in 3NF because all non-primary keys (RentalDiscount, PurchaseDiscount, RentalSpecialOffers, PurchaseSpecialOffers, OffersRentalStartDate, OffersRentalEndDate, OffersPurchaseStartDate and OffersPurchaseEndDate) are fully functionally and non-transitively dependent on the primary key MembershipID.

has(MembershipID, StudioName)

This is a many-to-many relationship, hence, it has no functional dependencies.