Tony Nguyen

Dr. Shawn Bowers

CPSC 321 01

28 November 2023

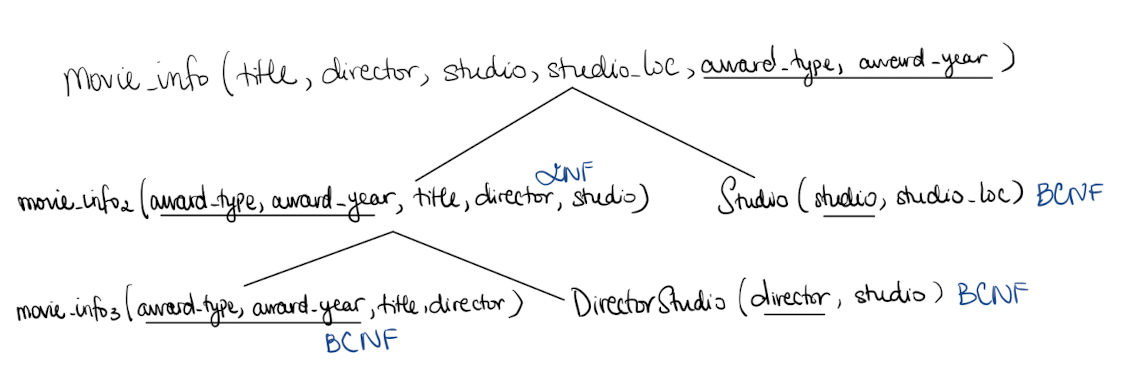
Homework 7

**Part 1**

1. Given the original schema and the functional dependencies (FDs), the primary key for the original schema is (award\_type, award\_year). From there, we can have the following information about the decomposed tables:

* movie\_info1(director, studio, studio\_loc)
  + Candidate key: director.
  + Highest normal form: 2NF since we have a transitive dependency of studio 🡪 studio\_loc.
* movie\_info2(title, director, studio)
  + Candidate key: (title, director).
  + Highest normal form: 1NF since we have a partial key dependency of director 🡪 studio.
* movie\_info3(award\_type, award\_year, title)
  + Candidate key: (award\_type, award\_year).
  + Highest normal form: BCNF since there is no redundancy caused by FDs.
* movie\_info4(award\_type, award\_year, studio\_loc)
  + Candidate key: (award\_type, award\_year)
  + Highest normal form: BCNF, as there is no bad FD, thanks to Transitivity.

1. Schema decomposition using the BCNF algorithm.



**Part 2**

1. Draw an ERD for the ski resort.

A diagram of a function

Description automatically generated

1. Database schema translation for Question 3.

* SkiResort(name, location, total\_elevation).
* SkiRun(name, level, total\_length).
* SkiLift(name, capacity, chair\_type).
* ResortRun(resort\_name, run\_name).
  + ResortRun.run\_name 🡪 SkiRun.name.
  + ResortRun.resort\_name 🡪 SkiResort.name.
* ResortLift(resort\_name, lift\_name).
  + ResortLift.resort\_name 🡪 SkiResort.name.
  + ResortLift.lift\_name 🡪 SkiLift.name.
* RunLift(run\_name, lift\_name).
  + Note: Only contains SkiRun-SkiLift pairs that are connected.
  + RunLift.lift\_name 🡪 SkiLift.name.
  + RunLift.run\_name 🡪 SkiRun.name.

1. Create an ERD for the micromobility vehicles.

A diagram of a flowchart

Description automatically generated

1. Database schema translation for Question 5.

* VehicleType(id, form, make, trim\_level, weight, max\_range, max\_speed, model).
* Vehicle(code, circulation\_date, initial\_cost, purchase\_condition).
* VehicleStatus(vehicle\_code, time\_stamp, circulation\_status, reservation\_status, fuel\_pctg, disabled\_status, latitude, longitude, range\_left).
  + Note: Only Vehicle that has a VehicleStatus will be on this table.
  + VehicleStatus.vehicle\_code 🡪 Vehicle.code.
* Customer(id, first\_name, last\_name, email, card\_type, card\_num, card\_exp).
* Rental(rental\_num, start\_lat, start\_long, end\_lat, end\_long, amount\_charged).
* PricePlan(code, name, unlock\_price, price\_per\_min, min\_after\_unlock).
* VehicleAndVehicleType(vehicle\_code, vehicle\_type\_id).
  + VehicleAndVehicleType.vehicle\_code 🡪 Vehicle.code.
  + VehicleAndVehicleType.vehicle\_type\_id 🡪 VehicleType.id.
* PreferedVehicle(c\_id, vt\_id, date\_added).
  + PreferedVehicle.c\_id 🡪 Customer.id.
  + PreferedVehicle.vt\_id 🡪 VehicleType.id.
* CustomerRental(c\_id, rental\_num).
  + Note: Only Customer who rents a vehicle will be on this table.
  + CustomerRental.c\_id 🡪 Customer.id.
  + CustomerRental.rental\_num 🡪 Rental.rental\_num.
* RentVehicle(rental\_num, v\_code).
  + Note: Only rented vehicles will be on this table.
  + RentVehicle.rental\_num 🡪 Rental.rent\_num.
  + RentVehicle.v\_code 🡪 Vehicle.code.
* RentalPrice(rent\_num, p\_code).
  + Note: Only price plans that have been used by a rental will be on the table.
  + RentalPrice.rent\_num 🡪 Rental.rent\_num.
  + RentalPrice.p\_code 🡪 PricePlan.code.

1. Create an ERD for HW2 Question 1.

A diagram of a flowchart

Description automatically generated

1. Database relation translation for Question 7.

* Song(title, track\_id, year).
  + Song.track\_id 🡪 Track.track\_name.
* Artist(name, birth\_year).
* Track(track\_name, year).
* MusicGroup(name, year\_formed).
* Album(title, year, group, record\_label).
  + Album.group 🡪 MusicGroup.name.
  + Album.record\_label 🡪 RecordLabel.name.
* RecordLabel(name).
* MusicGenre(label, description).
* SongAndArtist(song\_title, artist\_name).
  + SongAndArtist.song\_title 🡪 Song.title.
  + SongAndArtist.artist\_name 🡪 Artist.name.
* TrackAndArtist(track\_name, artist\_name).
  + TrackAndArtist.track\_name 🡪 Track.track\_name.
  + TrackAndArtist.artist\_name 🡪 Artist.name.
* AlbumAndTrack(album\_title, track\_name, group).
  + AlbumAndTrack.album\_title 🡪 Album.title.
  + AlbumAndTrack.track\_name 🡪 Track.track\_name.
  + AlbumAndTrack.group 🡪 Music.group.
* GroupAndArtist(group\_name, artist\_name, start\_date, end\_date).
  + GroupAndArtist.group\_name 🡪 MusicGroup.name
  + GroupAndArtist.artist\_name 🡪 Artist.name.
* GroupAndGenre(group\_name, genre).
  + GroupAndGenre.group\_name 🡪 MusicGroup.name.
  + GroupAndGenre.genre 🡪 MusicGenre.label.
* Influence(group\_name, influenced\_by).
  + Influence.group\_name 🡪 MusicGroup.name.
  + Influence.influenced\_by 🡪 MusicGroup.name.
* Compared with HW2, the only change I made was to implement the TrackAndArtist schema. I did so as there is a possibility that the artist(s) who perform a given track is not the one(s) who wrote the associated song.