Corinna Pilcher, Tom Zhang, Kim Merchant
Dr. Harcourt
CS 345 Final Project
Functional Requirements, Dependencies & 4NF Analyses

Document History

- 4/16 - Final edits
- 4/15 - B-Tree writing, organize past notes, final edits, last queries
- 4/14 - Organized information for presentation time, wrote a number of queries, and continued filling database
- 4/13 - Updated ER diagram and produced DDL code for database - preliminary database population
- 4/12 - Tom worked on decompositions and 4NF analyses and modified ER diagram accordingly
- 4/7/21 - Reviewed with Ed, solidified elements of ER diagram and Kim began decomposition proofs
- 4/5/21 - Corinna drafted ER diagram
- 4/2/21 - Reviewed with Ed, edited functional dependencies and entity review
- 3/29 - Wrote preliminary functional dependencies
- 3/27 - Functional requirements draft

Misc

Users: administrators of a singular theater company - producer, director, stage manager ONLY

Functional Requirements

[Note: many trivial requirements, such as adding data to the database, or deleting data, have been excluded for brevity and clarity. More interesting queries are bolded.]

1. The user should be able to look up the job title of individual people in the company.
2. The user should be able to look up the people who work in each job title (stage manager, actor, musician, administrator, director, trustee, etc)
3. **The user should be able to look up the crew at each venue for each show.**
4. **The user should be able to look up the current and historical hourly rate of each company member and the show they were paid that rate.**
5. The user should be able to look up the book assigned to each musician for a show.
6. The user should be able to look up the instruments assigned to each musician in a show.
7. **The user should be able to look up how many musicians are playing from each book.**

8. The user should be able to find out the instrumentation of a show.
9. The user should be able to see the costs of different types of transportation.
10. The user should be able to see the capacity of types of transportation (volume and seating).
11. The user should be able to look up the cast members for any show.
12. The user should be able to look up donors and their contact information.
13. The user should be able to look up the services provided by a given company (catering, audio/video recording) and their costs.
14. **The user should be able to find the donation tier of a donor or sponsor.**
15. The user should be able to look up the venue in a specific zip/state
    a. The company only visits one venue per zip code on a long tour
16. The user should be able to view the square footage of a venue's stage.
17. The user should be able to look up what flooring sections can be removed from the stage (for trapdoors).
18. The user should be able to look up the seating capacity of a venue.
19. The user should be able to look up if any seats can be removed from the front rows of a venue (in the situation where seats might need to be removed to fit an orchestra).
20. **The user should be able to look up the orchestra capacity of a pit (or, lacking a pit, the space taken by removable seats) to verify if the current members of the orchestra will fit in a venue.**
21. The user should be able to look up the price tickets are sold for to adults, children, and seniors.
22. **The user should be able to look up the cost per square foot of a venue's stage based on rental fee.**
23. **The user should be able to look up the price per seat and price per cubic foot of any given transportation.**
24. **The user should be able to look up the tour dates and locations planned.**
25. The user should be able to look up accreditation for any show performed (lyric writer, music writer, etc.)
26. **The user should be able to look up how many scripts/orchestra books the company currently is renting.**
27. **The user should be able to calculate the total costs of the company for a given time span (like the "first quarter", which could be January-March for a company).**
28. **The user should be able to look up the number of hours each member has worked for a particular time period or show.**
29. **The user should be able to look up the number of hours a subsection of the company has worked (actors, managers, musicians, etc.) and the amount the group was collectively paid per time period or show.**
30. **For each individual person, the user should be able to look up how many shows they have been involved in.**
31. The user should be able to look up the span of dates in which the company is working on a particular show.
32. The user should be able to list all actors and their roles for a given show.

33. **The user should be able to compare the potential ticket sale revenue to the venue rental fee.**

Functional & Multivalued Dependencies[1]

[Note: dependencies edited to match the final database relational design. Originally they were not written with specific relations in mind.]

1. By Func. Req. 1: name, birthdate → dept, title
2. By Func. Req. 2:
    a. title ↠ name, birthdate
    b. dept ↠ name, birthdate
3. By Func. Req. 3: ShowVenue.name, ShowDates.date → size of crew*
4. By Func. Req. 4: name, birthdate, year → wage, ShowNames.name
5. By Func. Req. 5: name, birthdate, dept, date →book
    a. The dept will be “Orchestra”
6. By Func. Req. 6: name, birthdate, dept, date ↠ Instrument
    a. The dept will be “Orchestra”
7. By Func. Req. 7: BookAssignment.book ↠ name, show_date, birthdate
8. By Func. Req. 8: ShowName ↠ instrument
9. By Func. Req. 9: Transportation.type, date → cost
10. By Func. Req. 10: Transportation.type → seats, cubic_ft
11. By Func. Req. 11: Role.show_date ↠ name, birthdate
12. By Func. Req. 12: Donor.name, Donor.birthdate → phone, email
13. By Func. Req. 13: Company.name → type, cost
14. By Func. Req. 14: name, birthdate, show_date, amount → donation tier*
15. By Func. Req. 15:
    a. Venue.name → zip_code
    b. Zip_code → venue.name
16. By Func. Req. 16: Venue.name → sq_ft
17. By Func. Req. 17: Venue.name → removable_floor
18. By Func. Req. 18: Venue.name → seating capacity
19. By Func. Req. 19: Venue.name → removable_seats
20. By Func. Req. 20: Venue.name → will the orchestra fit at the venue*
21. By Func. Req. 21: Ticket.type → cost
22. By Func. Req. 22: fee, sq_ft → cost per square foot*
23. By Func. Req. 23: Transportation.type, date → price/seat, price/ft^3*
24. By Func. Req. 25: Show_name → lyrics writer, show writer
25. By Func. Req. 26: Date → # books/scripts rented*
26. By Func. Req. 27: date range → total cost*
27. By Func. Req. 28: name, birthdate, show_date → hours*

[1] Any dependencies followed by an asterisk (*) indicate that in the final design decisions, the right hand side of the dependency was ultimately calculated through a query, rather than being stored in the database explicitly.

28. By Func. Req. 29: dept, date → total the department was paid in time period*
29. By Func. Req. 30: name, birthdate → number of productions*
30. By Func. Req. 31: ShowDates.name ↠ date
31. By Func. Req. 32: show_date ↠ name, birthdate, character

ALL RELATIONS SHOWN ON OUR ER DIAGRAM ARE BELOW:

Relations:

- Company(name, type, cost, venue_name)
    a. This refers to catering or audio/video recording companies used at a specific location (they do not travel with the theater company)
- Venue(name, zip_code, state, fee, seats, sq_ft, removable_seats, removable_floor, orchestra_capacity)
- VenueTime(name, time, day)
- TimeSlot(time, day)
    a. This refers to the name of a day of the week and a time to be reserved. The same day-time combinations are used for every venue
- ShowVenue(date, name)
- ShowDates(date, name)
    a. Date refers to the date of a performance; this relation lists every performance ever done of any show by the theater company
- FirstShow(date)
    a. This is a relation listing the dates of the first performances of any production
- ShowNames(name)
- ShowWriters(show_name, music_writer, script_writer)
- Characters(show_name, character)
- Book(show_name, book, instrument)
    a. A separate entry in this relation for each instrument of each instrument book
- Role(show_date, name, birthdate, character)
    a. A person can play multiple roles (for example, in *Into The Woods*, Prince Charming and the Big Bad Wolf are typically played by the same actor)
- Understudy(name, birthdate, show_date, character)
    a. A person can understudy multiple characters at a time
- BookAssignment(name, show_date, birthdate, book)
- InstrumentAssignment(name, birthdate, show_date, instrument)
    a. A musician will usually play more than one instrument in a show
- Plays(name, birthdate, instrument)
    a. For instruments a person knows how to play (not necessarily is assigned to)
- Instruments(instrument)
- People(name, birthdate, dept, title, phone, email)
    a. Phone is a must when travelling, email is supplied by the company
- Wages(name, birthdate, year, wage)
    a. Contracts renewed on yearly basis in terms of pay (in this model)

- Hours(show_date, name, birthdate, hours)
- Transportation(type, date, seats, cubic_ft, cost)
- DonorTier(type, threshold)
- Donations(name, birthdate, show_date, amount)
- Donors(name, birthdate, phone, email, anonymous)
- Ticket(type, cost)
    a. Types include: child, student, adult, senior

[Note: show_date refers to the first date of an individual production's performance, except for in ShowDates relation, which lists every performance of every show. It is assumed all instrumentation, wages, etc. will remain consistent throughout a production.]

4NF Proofs

- Company(name, type, cost, venue_name)

According to our domain expert, the names of the companies will be unique. Note that in this relation, there will be two types of companies: catering and audio/video recording. Two companies of different types will have different names. Also, because our theatre group will be traveling all the time, we will never partner with a particular company more than once (aka at more than one venue). Hence, it is safe to assume that name is a superkey, and the only functional dependency is:

name → type, cost, venue_name

Thus, relation Company is in 4NF.

- Venue(name, zip_code, state, fee, seats, sq_ft, removable_seats, removable_floor, orchestra_capacity)

As we were told by the domain expert (Kim), venues will have unique names, so these can serve as a primary key. The zip code and state are thus determined by the name of the venue. In each venue, the rental prices are set by the venue, and the square footage of the stage, the number of seats, whether any seats can be removed, and the orchestra capacity, all rely on the venue itself. The orchestra capacity is determined without considering removable seating, which is the only place an additional dependency could have cropped up. The zip code, though not chosen as our primary key, is still a superkey of the relation because it determines the name of the venue, which determines every other attribute. Thus, the only dependencies here are as follows and the relation is in 4NF:

name → zip_code, state, fee, seats, sq_ft, removable_seats, removable_floor, orchestra_capacity

zip_code → name, state, fee, seats, sq_ft, removable_seats, removable_floor, orchestra_capacity

- VenueTime(name, time, day)

Name for a venue will be unique; a different venue might have different available time slots selected for different days. Hence, all 3 attributes form the key for this relation. Since the relation is all key, it is in 4NF.

- TimeSlot(<u>time</u>, <u>day</u>)

  Time slots are a combination of day of the week, and time of day (hourly). Thus this relation is all key, and in 4NF. Functional dependency:

  time, day → time, day

- ShowVenue(<u>date</u>, name)

This relation records which venue was used on which show date. Since only one venue can be used on a particular day, date is unique (and a superkey), and we have:

date → name

Further, it is possible to use the same venue more than once (on different dates), so we have this multivalued dependency which is trivial:

name ↠ date

Therefore, this relation is in 4NF.

- ShowDates(<u>date</u>, name)

  A company will only be putting on one show at a time according to the domain expert. This means that the performance date is a valid primary key. The only non-trivial functional dependency here is:

  date → name

Hence, relation Show is in 4NF.

- FirstShow(<u>date</u>)

  Relations of only one attribute are by default in 4NF.

- ShowNames(<u>name</u>)

  Relations of only one attribute are by default in 4NF.

- ShowWriters(<u>show_name</u>, music_writer, script_writer)

  The same show will always have the same writers. Hence,

  show_name → music_writer, script_writer

would be our only functional dependency; thus, the relation is in 4NF.

- Characters(<u>show_name</u>, <u>character</u>)

The relation records a list of characters for each show, a character is guaranteed to be unique by name (otherwise scripts would be very confusing). A show can of course have multiple characters; thus, {show_name, character} form the key, and we have a single multivalued dependency which is trivial:

show_name ↠ character

Therefore, the relation is in 4NF.

- Book(show_name, book, instrument)

Instrumental books would be things like Trumpet 1, which might contain Bb trumpet, piccolo trumpet, and flugelhorn, while Reed 1 might have piccolo, flute, and oboe. This relation breaks down which instrument is in which book, and then what show that book belongs to. Hence, all three attributes are needed to form the key. The only functional dependency is trivial:

show_name, book, instrument → show_name, book, instrument

We also have one trivial multivalued dependency:

show_name ↠ book, instrument

Thus, the relation Book is in 4NF.

- Role(show_date, name, birthdate, character)

Attributes {Name, birthdate} uniquely identify a person, who will participate in several shows; also, it is possible for each person to play more than one character for a particular show. Hence, {name, birthdate, show_date, character} form the key for this relation. We also have one multivalued dependency which is trivial:

name, birthdate, show_date ↠ character

Hence, relation Role is in 4NF.

- Understudy(name, birthdate, show_date, character)

This relation keeps track of all understudies who relate directly to the more "formal" roles in a show. The relation contains exactly the same attributes and dependencies as Role; hence, following the same logic, relation Understudy is also in 4NF.

- BookAssignment(name, show_date, birthdate, book)

A musician will only be assigned one book on a particular show date. Thus, {name, birthdate, show_date} form the key, and we have a single functional dependency:

name, birthdate, show_date → book

Hence, this relation is in 4NF.

- InstrumentAssignment(name, birthdate, show_date, instrument)

It is true that a musician will play more than one instrument in a particular show; for instance, two people may split the instrumentation within a book. Hence, the relation is all key, and we have one multivalued dependency which is trivial:

name, birthdate, show_date ↠ instrument

Therefore, the relation is in 4NF.

- Plays(name, birthdate, instrument)

______A musician is uniquely identified by {name, birthdate} and could play multiple instruments; hence, all 3 attributes are needed to form the key for this relation. Consequently, we also have this multivalued dependency which is trivial:

name, birthdate ↠ instrument

Hence, this relation is in 4NF.

- Instruments(instrument)

Relations of only one attribute are by default in 4NF.

- People(name, birthdate, dept, title, phone, email)

A person can be identified by their name and birthdate in almost 100% of situations, especially in a limited size sample like a production company, so this is the primary key. From this, it can be determined what department a person works in, their title in that department, and their contact information. The company need only require one phone number and one email for contacts. It is assumed the company will provide an email, and the phone number can be expected to a similar capacity (it is a work necessity for a travelling group). Therefore, the functional dependencies are:

name, birthdate → title, department, phone, email

Since {name, birthdate} is a superkey, relation People is in 4NF.

- Wages(name, birthdate, year, wage)

A particular person identified by {name, birthdate} could have a different wage every year. Hence, we have functional dependency:

name, birthdate, year → wage

Note that {name, birthdate, year} is a superkey. Also, the only multivalued dependency would be:

name, birthdate ↠ year, wage

Note that the dependency is trivial; hence, relation Wages is in 4NF.

- Hours(show_date, name, birthdate, hours)

The relation keeps track of the number of hours each person has worked on a particular day. Hence, we have functional dependency:

name, birthdate, show_date → hours
Note that {name, birthdate, show_date} is a key. Also, we have multivalued dependency which is trivial:
name, birthdate ↠ show_date, hours
Hence, relation Hours is in 4NF.

- Transportation(type, date, seats, cubic_ft, cost)

On a particular date, the theatre group might need to utilize more than one method of transportation; hence, {date, type} together form the key for this relation. Once a date and a type are set, we can determine the number of seats, size, and cost. Hence,
date, type → seats, cubic_ft, cost
Note that {date, type} is a superkey. Also, we have a multivalued dependency which is trivial:
date ↠ type, seats, cubic_ft, cost
Therefore, relation Transportation is in 4NF.

- DonorTier(type, threshold)

This is a simple relation that maps donation tiers to thresholds. The only non-trivial functional dependency is:
type → threshold
Since type is a superkey, the relation is in 4NF.

- Donations(name, birthdate, show_date, amount)

This relation keeps track of every donation ever made for each show. Name and birthdate uniquely identify a donor, who may donate at many shows. Since there will be exactly one show on a given date, a donor will donate only once for that date. Hence, attributes {name, birthdate, show_date} combined make a superkey. Thus, this is the only functional dependency:
name, birthdate, show_date → amount
Now, since a donor could make several donations for more than one show, we could have the following multivalued dependency as well:
name, birthdate ↠ show_date, amount
Notice that this multivalued dependency is trivial. Therefore, the relation Donations is in 4NF.

- Donors(name, birthdate, phone, email, anonymous)

To simplify matters, administrators of the company have decided to require donors to have exactly one phone number and one email. Also note that {name, birthdate} uniquely identify a donor, making it a superkey. Therefore, the only functional dependency in this relation would be:

name, birthdate → phone, email, anonymous

Hence, relation Donors is in 4NF.

- Ticket(<u>type</u>, cost)

  The price of tickets is based solely on the type of the customer (child / adult / senior), so the functional dependency here is:

  type → cost

Because type is also a superkey, relation Ticket is in 4NF.